Learning and Control Applied to Demand Response and Electricity Distribution Networks

by

Gregory Stephen Ledva

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in the University of Michigan
2019

Doctoral Committee:

Assistant Professor Johanna L. Mathieu, Chair
Assistant Professor Laura Balzano
Professor Ian Hiskens
Professor Jerome P. Lynch

Gregory Stephen Ledva
gsledv@umich.edu
ORCID iD: 0000-0003-0155-6943

To my family, who have always supported me throughout this journey
(even if they didn't understand what I was doing)

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

Figure

LIST OF TABLES

<u>Table</u>

ABSTRACT

Balancing the supply and demand of electrical energy in real-time is a core task in power system operation. Traditionally, this balance has been achieved by controlling power plants, but increasing amounts of renewable energy generation increases the variability in generation and requires additional energy balancing capacity. An alternative to providing this additional capacity via power plants is to provide signals to loads that induce changes in their demand, which is referred to as demand response.

There exists a large potential capacity for demand response using residential loads, but enabling these loads to participate in demand response requires communication and sensing capabilities. Thermostatically controlled loads (TCLs) are ubiquitous in residences and have inherent flexibility as they cycle on and off during normal operation. Coordinating on/off switching of TCL aggregations can provide energy balancing. However, TCLs are a spatially distributed resource that require sensing and communication infrastructure to enable demand response capabilities. A key to realizing cost effective residential demand response is minimizing infrastructure costs while maximizing the accuracy of the provided energy balancing, which results in increased revenue while improving reliability in the power system.

The main contribution of this dissertation is to show that advanced algorithms can leverage existing infrastructure to make energy balancing with loads feasible in the near-term, which improves the reliability, economics, and environmental impact of the power grid. The dissertation first presents control algorithms, estimation algorithms, and models for residential demand response on fast timescales, i.e., on the order of seconds. Following this, the dissertation presents online learning algorithms for real-time feeder-level energy disaggregation within an electricity distribution network, which can be used to estimate the demand-responsive load in real-time. Methods for both topics are developed to operate within the capabilities of existing communication and sensing infrastructure, which reduces the implementation costs of the methods.

Control and estimation algorithms are developed that address communication delays while taking into account realistic measurement availability. Results indicate that incorporating delay information into the algorithms can mitigate the effects of communication delays, allowing demand response providers to reduce infrastructure costs by using less expensive, lower quality communication networks. Additional work adapts three existing residential demand response models for a more detailed simulation environment, modifies each model to be more accurate in this environment, and benchmarks the model variations against each other. Results indicate that the model modifications produce more accurate predictions versus the unmodified models. Improving modeling accuracy can improve the reliability of the system and increase revenues for a demand response provider by improving the performance of model-based control and estimation algorithms.

The energy disaggregation algorithms seek to separate the measured demand of a distribution feeder into components (e.g., the demand-responsive load and the remaining load) as feeder-level measurements become available. An online learning algorithm is adapted to perform real-time energy disaggregation using active power measurements of the total demand on the distribution feeder. Results indicate that the algorithm is able to effectively separate the air conditioning demand from the remaining demand connected to a distribution feeder. This algorithm is then extended to include reactive power, voltage, and smart meter measurements. Results indicate that the availability of additional real-time measurements leads to more accurate disaggregation of the demand components. Additional work in state estimation establishes connections between the online learning methods used and Kalman filtering algorithms.

CHAPTER I

Introduction

An electric power network, or an electricity/power grid, is the network of equipment to produce electrical energy and to transport that energy from the energy producers to the energy consumers within a region. The reliable supply of electricity to consumers, or reliability, is central to economic well-being, and estimates of the annual cost of power outages in the US range from $20B to over $100B [1]. The main challenge in operating the power network to ensure reliability is that the production and consumption of electricity within a power network must be balanced at all times [2]. This balance has traditionally been achieved by controlling the bulk electricity generation resources, i.e., the bulk energy producers such as power plants [3]. Recently, electricity generation from renewable energy sources has been increasing, which can require additional flexibility in other controllable resources within the power network to counter the intermittent power generation from these renewable resources. Regulatory changes and the push to incorporate modern technology into the power grid (i.e., to create a "smart" grid) have enabled non-traditional resources such as demand response to participate in the operation of the power network and help balance the electricity production and consumption [4]. As a result, there is increased interest in expanding the usage of demand response to provide this additional flexibility [5]. Demand response is defined as the following:

> "Demand response is a tariff or program established to motivate changes in electric use by end-use customers in response to changes in the price of electricity over time, or to give incentive payments designed to induce lower electricity use at times of high market prices or when grid reliability is jeopardized [6]."

In this work, we use the term demand response to refer changes in the electricity demand caused by signals provided to the demand that help to ensure reliable power network operation.

The work presented in this dissertation develops control algorithms and learning algorithms that are applied to demand response and to electricity distribution networks. The control algorithms include three inter-related tasks: modeling, state estimation, and controller development. Modeling develops a mathematical description of a system of interest, where the state of the system describes the present “status” of the the system, where the input of the system is a signal that influences the system state, and where the output of the system is a signal that consists of the measurements available from the system, which are derived from the system state. State estimation seeks to estimate the state of the system given the outputs of the system and the inputs to the system, where the inputs are usually known as they are computed by a controller. Controller development seeks to design an algorithm that determines an input to the system given an estimate of the system state or given the output of the system, where the inputs try to achieve some desired behavior within the system.

These control algorithms are applied to demand response scenarios, where the algorithms enable automated changes to the aggregate, or total, electricity demand of collections, or aggregations, of residential appliances on fast time-scales (i.e., on the order of seconds) within power network operation. Specific attention is given to practical issues and limitations within the communication and sensing infrastructure needed to enable automated demand response on these time-scales, and these issues and limitations are included within the estimation and control algorithms. By accounting for these issues and limitations, the algorithms are able to produce accurate demand changes or accurate estimates of power system quantities while avoiding prohibitory costs of extensive upgrades to this infrastructure. Incorporating models into estimation and control algorithms can improve their performance. Additional work presented in this dissertation develops aggregate models of the residential appliances (or residential loads) used for demand response, where the aggregate models capture the total behavior of the residential appliance population. These models are based on previously developed models, and they are modified to make them more applicable to real-world scenarios.

The work presented in this dissertation also develops learning algorithms that are within the online learning framework of supervised machine learning. Supervised machine learning uses historical data to estimate a predictor, where the predictor computes an output given input data, and where the historical data contains both the input and output information. In supervised learning, offline or batch learning uses a set of historical training data to compute the predictor, and then applies this predictor as new input data becomes available. In online learning, data becomes available sequentially over the course of time, and in contrast with offline learning, the predictor is updated as each new piece of input and output data becomes available.

The learning algorithms are applied to electricity distribution networks, where the goal is to use measurements from within the distribution network to estimate components of the load, or electricity usage/demand. Electricity distribution networks are part of the infrastructure used to transport electricity to consumers, and the buildings connected to these networks are usually residences and small commercial buildings. This goal has a number of uses, but within the demand response problem, it can be used to obtain estimates of the aggregate electrical demand of the appliances that are available for demand response in real-time. These aggregate demand estimates can then be used within demand response control and state estimation algorithms. As with the control algorithms, these learning algorithms take into account the capabilities of measurement infrastructure that are commonly available in distribution networks, enabling their implementation with minimal infrastructure upgrades.

In the remainder of this chapter, Section 1.3 presents the general problem formulation considered within this dissertation in more detail, and it then summarizes the research contributions of the remaining chapters within this dissertation. In order to provide additional background to the material in Section 1.3, Section 1.1 provides background on electric power networks and their operation, and Section 1.2 provides background on demand response using residential loads, referred to as residential demand response in this work.

1.1 Background on Electric Power Networks

Electric power networks are considered one of the largest and most complex systems ever constructed. Generators (traditionally, large power plants) produce electrical energy. Electrical energy is consumed by loads, or the devices and processes within end-user facilities (such as factories, commercial buildings, and residences) that use the electric energy to drive other processes. The energy is transported from the generators to the loads through the transmission and distribution systems, which contain equipment such as conductors (the power lines) and transformers. Some electrical energy is consumed by the transmission and distribution systems, which is referred to as losses. The total consumption of the losses and loads is referred to as the electricity demand.

A system operator is responsible for operating the power network. Power network operation consists of coordinating the generators, power network equipment, and, increasingly, the demand to ensure the reliable supply of the desired power to consumers while ensuring equipment operates within its engineering limits. Examples of these engineering limits include the thermal limits of conductors, limits on the generation capacity of power plants, limits on voltages within the network, and limits on the frequency of the currents within the alternating current (AC) network [2]. The system operator modifies resources within the network (e.g., the generation settings of power plants and the topology of conductor connections) on a variety of time-scales, from fractions of a second through years, to ensure that the balance between energy generation and consumption is maintained [2].

Regulatory changes, i.e., restructuring, have changed the paradigm of system operation from one based on regulated monopolies to one based on markets [2]. Before these regulatory changes, which began in the US in the 1990s, utilities operated as vertically integrated, regulated monopolies that had control of all generation, transmission, and distribution resources within its network [7]. These monopolies were the only electricity provider within their power network; they supplied power to all customers connected to the network, and prices were set by an external regulating agency [7]. Restructuring forced monopolies to split the ownership of their assets into separate entities that owned either generation, transmission, or distribution resources [7]. Owners of transmission and distribution system resources are still operated as regulated monopolies, and owners of generation resources now compete to provide electrical energy and other services in an open market [7]. Restructuring also created the entity called a system operator that is independent from the other entities that own resources within the power network. Previously, the monopoly owned all resources within its network and could operate them as it saw fit to ensure the reliable supply of electricity, but after restructuring, many resources that ensure reliability are obtained through markets [7].

This section presents some background on the basic network structure of AC power networks, their basic operation principles, and an overview of electricity markets. The background on the operation principles and markets pertain specifically to restructured electricity markets within the US. The goal is to provide context for the general demand response problem addressed in this dissertation, and so only the necessary details are presented. References on electric power networks such as [2,8,9] can be consulted for additional background.

1.1.1 Basic Electric Power Network Structure

The structure of the electric power network helps ensure that the transport of electricity from producers to consumers is done in a reliable, efficient, and safe manner [2]. The equipment that transports the electric power is divided into separate levels (or systems or networks) based on the voltage of the network, and transformers link the levels by converting the flow of electricity from one voltage level to another. The two main levels within the power network are the transmission system, which operates at high voltages, and the distribution system, which operates at low voltages, but there may also be intermediate levels with intermediate voltages [2]. Large generators (e.g., power plants) are connected to the transmission system via a transformer, large loads (e.g., industrial and large commercial facilities) are connected to intermediate network levels, and small generators (e.g., residential solar generation) and small loads (e.g., residential appliances) are connected to the distribution system [2]. The transmission system operates at high voltage to reduce losses in transferring large quantities of electrical energy over long distances, and the transmission system contains redundant paths (i.e., it is a meshed network) to help ensure the reliable transport of electrical energy in the event of faults, or malfunctions [2]. The distribution system operates at a low voltage to help ensure the safe interaction of the general population with the power network, and it usually radial, meaning that there is one or few paths from one point within the network to another [2]. The radial structure helps to reduce the equipment costs and to improve the ability to isolate faults [2]. Finally, distribution feeders are the conductors that supply electrical energy from the distribution substation to the buildings connected to the distribution system. The distribution substation contains transformers that convert the voltage from higher levels to that of the distribution system.

1.1.2 Basic Operation Principles

The generation and consumption of two types of power must be balanced within the power network: active power and reactive power. Active power is the flow of electrical energy that can be used to drive other processes, and reactive power is the flow of energy that oscillates within the power network due to coupling between electric and magnetic fields [2]. A mismatch in the production and consumption of active power causes the system frequency to deviate from its desired value. At the transmission system level, a mismatch in the production and consumption of reactive power mainly influences the voltage levels within the power network, which must be maintained within specific ranges to ensure equipment health and network reliability [2]. In a distribution system, the relationship between reactive power and voltage levels is not as strong due to different characteristics of the conductors versus those in the transmission system. In this work, we are mainly interested in maintaining, or regulating, the frequency in real-time about its desired value of 60 Hz in the US.

Real-time frequency regulation occurs through a combination of passive and active methods [2]. Passive methods are actions inherent in the equipment that help to correct the balance between the production and consumption of active power, and active methods are those where some conscious (possibly automated) action is taken within the system. Active frequency regulation methods are needed to maintain the frequency within an acceptable range of its desired value, where compensation is needed due to differences in the predicted or scheduled generation and demand.

The active control methods include governor control, automatic generator control (AGC), and scheduling [2]. Governor control, the fastest of the three methods, operates on the time-scale of fractions of seconds to seconds. When a power plant employs governor control, the plant's power production is automatically adjusted based on the rotational speed of the power plant's generators; the generators speed up due to the conservation of energy if the system's total power generation exceeds the total demand, and the generators slow down if demand exceeds generation [2]. AGC, the medium-speed method, operates on the time-scale of seconds to minutes. In AGC, the system operator coordinates and transmits automatic adjustments to the power production settings of the generators[1] to regulate the system frequency and to ensure that power flows between specified regions of the network are carried out as planned [2]. Scheduling, the slowest of the three methods, operates on time-scales of minutes to days. Scheduling determines which power plants are producing power, and it determines their power production schedule based on the forecast demand [2].

1.1.3 Overview of Electricity Markets

In restructured power systems, electricity markets are used to help organize the operation of the power network. These markets act as a clearinghouse where offers to buy and sell power system products (such as energy production) are entered, ensuring that system operation can be carried out in an economically efficient manner [2]. The system or market operator runs these markets as an independent entity [7], for example, Regional Transmission Operators (RTOs) and Independent System

[1] Demand can participate in AGC via demand response programs by adjusting the aggregate demand of the loads in the opposite direction that a generator would.

Operators (ISOs) in the US. A number of markets exist to coordinate the long-term expansion of the network, schedule short-term energy production, and to schedule ancillary services, which are resources that help ensure the reliability of the power network.

Typical electricity markets include a capacity market, a day-ahead market, a real-time market, and an ancillary services market. The capacity market allows power production capacity to be bought or sold several years in advance, e.g., three years [10], which coordinates the long-term ability to meet the expected, future electricity demand. The day-ahead and real-time markets are energy markets because they coordinate the energy delivery to meet the demand forecast one day in advance and/or minutes in advance, respectively; these markets determine the nominal power production schedule of the generators or the demand adjustment schedules. In the day-ahead market, energy is purchased one day before it must be delivered, e.g., the market on Thursday sells energy that is delivered on Friday [11]. In the real-time market, energy is purchased for immediate delivery over some relatively short time period, e.g., five minutes. The real-time market allows last-minute balancing between the day-ahead generation commitments and the demand that is expected in the short-term, e.g., over the next five minutes [11]. However, imbalances will still occur between generation and consumption because the actual power generation may not match the commitments and because the demand will vary within the time periods of the real-time market, causing the system frequency to vary.

Finally, the ancillary services market secures services or capabilities that help ensure the real-time reliability of the power network [7]. These services include operating reserves and frequency-regulation reserves. Operating reserves are used to respond to major short-term changes in the electricity supply or demand, e.g., if a generator suddenly fails [7]. Frequency-regulation reserves ensure there is capacity to increase or decrease generation or demand in response to AGC signals to regulate the system frequency in real-time [7].

1.2 Background on Residential Demand Response

This section provides background on residential demand response by summarizing the evolution of research trends in residential demand response, from the inception of the power industry through the present day, and by developing the motivation for the research in this dissertation. The research focus in the field of residential demand response has changed over time as the loads connected to the power grid have changed, as technology has advanced, and as governmental regulations have changed. One mechanism to induce changes in residential demand is a market-based approach, which adjusts the electricity price paid by the consumers based on the price established in energy markets to induce manual changes in demand enacted by the residents (e.g., see [12–14]).[2]An alternate mechanism induces demand changes by transmitting signals directly to the loads (e.g., see [15–18]). The signals can be generated by a system operator or an automated algorithm, and actuation infrastructure enacts automated responses to the signals. Note that these signals could include time-varying prices, which an automated algorithm (e.g., within a household energy management system) uses to adjust the household demand [16, 19, 20]. The discussion below focuses on the latter mechanism, as it produces more predictable demand changes than the former mechanism, which is important when supplying reliability services such as frequency regulation [21].

Participants in the electricity industry have been interested in manipulating the demand of residential loads from the industry's inception in the late 19th century. In the 1880s and 1890s, electricity was mainly used at night for lighting purposes, and utilities discussed using time-varying electricity prices to promote more evenly distributed electricity demand [22]. In the 1930s, [15] proposed ripple control, which transmits a control signal through the power network equipment by superimposing a signal on the electrical currents flowing within the network, as a method to remotely switch lighting and electric water heaters on/off. In the 1970s, increased adoption of air conditioners led to extreme, short-term peaks in the electricity demand during hot days, and costly generation resources were often used to meet these peaks in demand [23]. Utilities began using residential demand response, most notably remote on/off switching of water heating, space heating, and space cooling appliances, to reduce this peak demand and/or shift energy usage to off-peak times, which reduced the high operating costs [23].

This constitutes the traditional setting for a residential demand response program: a utility compensates residents that enroll a household device into the demand response program. In exchange, the utility can remotely turn on/off the enrolled residential appliances. Under normal operation, the demand reaches peak values during a portion of the day, and the utility must use costly generators to produce energy for this demand. With demand response, the utility controls the residential appliances

[2] There are a number of works that attempt to develop a taxonomy for labeling demand response programs (e.g., see [14, 21]); the following discussion does not use terminology that strictly adheres to a specific taxonomy.

either to reduce the peak demand or to shift the energy usage to other times; this allows less costly generators to be used and reduces or eliminates the need to use these high-cost generators. After the widespread adoption of air conditioners, research focused on modeling the physical processes driving electric heating and cooling loads in residences and predicting demand changes from switching these loads on/off [24–30].

From the 1990s through the 2010s, restructuring, the California energy crisis, and a number of regulations paved the way for a paradigm shift in the role of residential demand response within a power network. Restructuring introduced electricity markets, and in the late 1990s and early 2000s, California experienced high and volatile electricity prices in its markets. A proposed solution was to use market prices to influence demand [31]. The Energy Policy Act of 2005 identified the integration of demand response into electricity markets as a major objective of US energy policy [6]. The American Recovery and Reinvestment Act of 2009 supplied $4.5B to start modernizing the electric power grid, i.e., the smart grid initiative, which funded projects that included the installation of smart meters at residences and the installation of communication technologies at the distribution system level [32].

In the 2010s, Federal Energy Regulatory Commission (FERC) Orders 719, 745, 755, and 784 also helped to create a favorable environment for demand response participation in power markets. FERC Order 719 allows aggregations of loads to participate in power markets [33]. FERC Orders 745 and 755 allow demand response resources to participate in energy markets [34] and ancillary service markets [35], respectively. FERC Order 755 includes compensation for ancillary services based on the speed and accuracy with which ancillary services are provided [35], and FERC Order 784 provides more compensation to resources providing ancillary services with faster ramping rates, i.e., resources that can rapidly change their generation/demand [36]. These compensation changes benefit residential demand response because aggregations of small loads, e.g., those in residences, are thought to be able to produce accurate and fast demand changes [37].

One major avenue of modern residential demand response research, enabled by advanced metering infrastructure such as smart meters, is the use of load aggregations to provide frequency regulation. This was initially investigated in [38], and much research has developed models, control algorithms, and estimation algorithms to improve residential demand response capabilities in this context (e.g., see [17,18,39–46]). In this setting, an aggregator, which can be a utility or an entity that specializes in demand response, can manipulate the demand of some set of demand-responsive loads, e.g., air conditioners; the demand-responsive loads are loads that have been enrolled in the demand response program by residential customers in exchange for some form of compensation. The aggregator can bid into an ancillary services market to offer frequency regulation to the network operator based on the ability of the demand-responsive loads to change their aggregate demand. If the frequency regulation bid is accepted, the aggregator receives a desired demand set-point from the system operator every few seconds, where the set-point is adjusted over time to provide frequency regulation. Finally, the aggregator is compensated by the system operator based on the capacity that they bid into the market and the accuracy with which the aggregator's total demand-responsive load tracked the system operator's set-points.

Demand response via a load aggregation requires sensing and communication infrastructure to connect the devices because the aggregation consists of a large number of spatially distributed loads. A demand response program that is providing frequency regulation can generate higher revenues with sensing, data communication, and actuation capabilities that enable faster and more accurate demand changes, but this capability may increase the infrastructure costs of the program. There is an increasing amount of this infrastructure already deployed at residences, e.g., smart meters [47] and home automation systems [48]. However, this existing infrastructure may have limitations in its capabilities. For example, existing smart meters have significant limitations in their communication and data storage capabilities [49]. Operating within the capabilities of existing infrastructure can help reduce costs by avoiding infrastructure upgrades, and control algorithms that take these capabilities into account can provide more accurate demand changes than if the capabilities are not accounted for. This is the motivation for the work within this dissertation.

1.3 Scope of Work

This section defines the system considered within this work and summarizes the contributions of this dissertation. Section 1.3.1 summarizes the general problem framework, i.e., the residential demand response scenario, considered within this dissertation. Each of the following chapters address a portion of this problem directly or develop algorithms or models that can be applied to this scenario. Section 1.3.2 summarizes the contributions of the remaining chapters within this dissertation.

1.3.1 General Problem Framework

This work focuses on the residential demand response scenario summarized in Fig. 1.1. The scenario contains three major components: a portion of a distribution system, three communication networks, and a power system entity (e.g., a utility or demand response provider), hereafter referred to as an aggregator in this chapter. The portion of the distribution network is the plant, or the physical system of interest; it consists of a distribution feeder and the buildings/loads connected to it. We assume that feeder-level measurements (e.g., power flows, voltage magnitudes, voltage angles, and currents) are available from a supervisory control and data acquisition (SCADA) system. We also assume that weather-related measurements of the feeder's geographic area are available. We assume that the plant contains a single distribution feeder for simplicity, but the framework can be readily extended to include multiple distribution feeders. The building-level loads (e.g., residences or commercial buildings) consist of the demand of each of the buildings connected to the distribution feeder. Each of the buildings is equipped with a smart meter, and we assume according to [49] that the smart meter is capable of measuring the total power demand (both active and reactive) of the building, the voltage at the smart meter, and the current flowing into the building. These measurements are stored and can be transmitted to the aggregator at some regular interval, e.g., every hour. Furthermore, we assume that some residences connected to the feeder contain loads, e.g., air conditioners, that are enrolled in a demand response program. Building-level energy disaggregation algorithms are capable of separating the measured demand of a building into the demand of different loads within the building [49], and so we assume that we have access to the demand of the loads participating in demand response on the same time-scales that smart meter measurements are available.

The communication networks enable the transfer of data between the plant and the aggregator. We assume three separate communication networks for generality as different communication infrastructure may be used to link different components within the system. These communication networks could be identical in practice, but modeling them separately allows for a more general problem framework by allowing different characteristics to be modeled for each communication network. The wide-area measurement network communicates the weather-related measurements and feeder-level measurements from the plant to the aggregator. The building-level measurement network communicates measurements of the individual buildings' total power demand to the aggregator. The input network communicates the control inputs from the aggregator to the plant; the control inputs are used to influence the demand-responsive load to produce a desired behavior, e.g., tracking a frequency regulation signal from a system operator. The aggregator computes broadcast inputs, meaning identical control inputs are sent to all demand-responsive loads within the load aggregation.

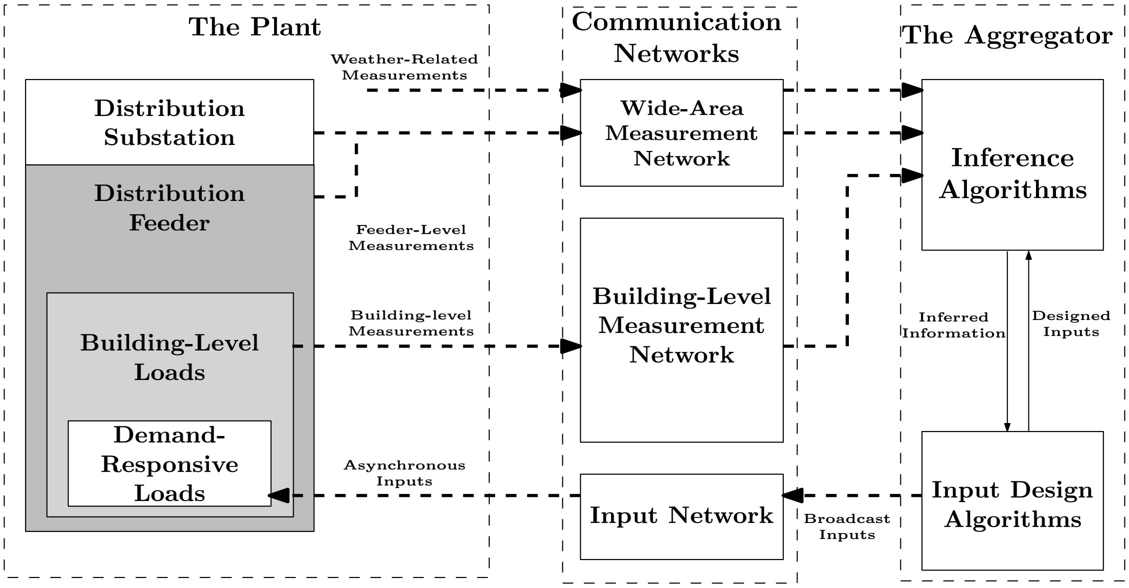


Figure 1.1: The three main components of the residential demand response framework, and the information transfer between those components

The control inputs may arrive asynchronously at the loads due to the effects of the communication network, meaning that the same information within the broadcast input may arrive at each demand-responsive load at different times.

Finally, the aggregator contains inference and input design algorithms that operate on time-scales of seconds to minutes. The inference algorithms use the measurements produced by the plant to infer information about the physical system. The input design algorithms use the inferred information to compute the broadcast inputs. The inference algorithms may incorporate information about the designed inputs, and the input design algorithms incorporate the inferred information.

Each of the following chapters develops algorithms or models that can be used within some portion of this system, and the assumptions regarding the sensing and communication capabilities are different in the various chapters. As a result, details of the assumptions are given in the respective chapters.

1.3.2 Contributions

This dissertation focuses on the development of online learning algorithms, control algorithms, and models for residential demand response and for distribution networks that operate on fast time-scales of seconds to minutes. The algorithms incorporate practical limitations of the communication and sensing infrastructure, and incorporating these limitations allows the algorithms to operate more effectively than algorithms that do not include the limitations. The remainder of the dissertation is organized as follows:

Chapter II presents work that mitigates the effects of communication delays while using residential demand response for frequency regulation on timescales of seconds. Communication delays, or delays in transmitting data through a communication network, can arise due to the communication infrastructure used. In addition, smart meters are not capable of transmitting data to an aggregator at these timescales, and so intermittent, device-level information about the demand-responsive loads is assumed to be available. Accounting for the delays as well as realistic measurement availability from smart meters allows the effects of these limitations on the ability to follow a frequency regulation signal to be reduced.

Chapter III presents work that only considers input delays, i.e., this is a simplified version of the system considered in the previous chapter. The work develops a simplified control algorithm, compared to the algorithm developed in the previous chapter. The goal is to reduce the computational requirement in mitigating input delays and to analyze the reduction in the ability to follow a frequency regulation signal due to the simplified controller.

Chapter IV benchmarks three aggregate models that were developed for residential demand response against on another in a comparable scenario. These models were developed in separate works, under different assumptions, and so evaluating them in a common, realistic simulation scenario allows the evaluation of their comparative accuracy. This chapter compares the accuracy of two Markov-based and one transfer function-based aggregate models of a set of air conditioners against a realistic simulation model with time-varying outdoor air temperature and temperature-dependent air conditioner parameters. The chapter also extends these existing models to cope with the time-varying outdoor air temperature.

Chapter V explores similarities between online learning algorithms and Kalman filtering algorithms, which are two approaches to estimate the state of a system in the presence of inaccurate (e.g., noisy) measurements. The chapter shows that Dynamic Mirror Descent (DMD), an online learning algorithm that incorporates a single model, can be constructed to produce state estimates that are identical to those produced by a discrete-time Kalman filter. Following this, the chapter extends this by exploring connections between a multiple model Kalman filter (MMKF) and Dynamic Fixed Share (DFS), which both incorporate a set of candidate models to address situations in which the underlying model is unknown. The functions/parameters used within DFS are constructed to produce the same estimates as a MMKF. Following this, DFS is modified to include several heuristics that are used to improve the performance of a MMKF in order to assess whether they can also be used to improve the performance of DFS. Finally, we investigate the performance of the algorithms and their variations in a simulation study that seeks to estimate the time-varying power consumption of an aggregation of electric loads, which could be used as the feedback signal within a demand response algorithm.

Chapter VI investigates the ability to disaggregate a distribution feeder's active power demand measurements into: 1) the demand of a population of air conditioners, and 2) the demand of the remaining loads connected to the feeder. It uses an online learning algorithm, DFS, that uses the real-time distribution feeder measurements as well as aggregate models generated from historical building- and device-level data. Two implementations of the algorithm are developed and case studies are conducted using real demand data from households and commercial buildings to investigate the effectiveness of the algorithm. Note that in this chapter, no control inputs are generated, and the loads operate normally.

Chapter VII extends the work of the previous chapter to perform energy disaggregation using real-time reactive power, voltage magnitude, voltage angle, and smart meter measurements in addition to the active power measurement used in the previous chapter. The aggregate models are also modified to incorporate complex current measurements. In contrast with the previous chapter, the distribution network is explicitly modeled in this work. As a result, the distribution feeder demand is separated into three components: 1) the demand of the air conditioners connected to the feeder, 2) the demand of the remaining loads connected to the feeder, and 3) the active and reactive power consumed by the distribution network. A version of the online learning algorithm developed in the previous chapter is modified to incorporate the various real-time measurements via sensor fusion.

Chapter VIII presents the conclusions of the dissertation. It summarizes the content within the dissertation, it summarizes the main contributions of each chapter, and it proposes a number of avenues of future research. These avenues include technical, economic, social science, and policy tasks for advancing residential demand response as well as additional, more general tasks in machine learning and state estimation.

Note that each chapter is self-contained; notation and acronyms are defined in each chapter and for that chapter alone.

CHAPTER II

Managing Communication Delays and Model Error in Demand Response for Frequency Regulation [1]

Incorporating more fluctuating, renewable power generation into the electricity network will usually lead to additional power production variability. To maintain the frequency within an acceptable range, generation resources must supply more reserves, which may require them to operate at inefficient operating points [52]. Alternatively, the manipulation of electric power demand using demand response is also capable of providing frequency regulation.

Common residential demand response methods include price-based demand manipulation and direct control of loads [53], e.g., residential thermostatically controlled loads (TCLs) such as air conditioners, heat pumps, and water heaters. Under normal operation, TCLs cycle on and off to maintain the temperature of an internal medium, e.g., a house's air temperature, around a user-defined set-point. Direct control strategies manipulate a TCL population's total power demand generally by adjusting either the user-defined temperature set-point, e.g., [18, 38, 39], or by requesting additional on/off switching, e.g., [17, 40, 41]. Aggregations of TCLs can be used to provide ancillary services such as frequency regulation to the power system [39]. Recently, researchers have developed non-disruptive load control strategies [37], ensuring TCLs operate within or very close to their normal temperature range [39].

TCLs are a spatially distributed resource, and coordinating the demand of thousands of them to provide frequency regulation requires sensing and communication infrastructure. This infrastructure enables TCLs to receive control inputs and to send

[1] The work presented in this chapter was originally published in [50], and preliminary work was published in [51].

information about their current operating state. However, the cost of this infrastructure can be prohibitive [54]. Using existing infrastructure, such as smart meters, is possible but the frequency of information retrieval is limited [49], and load control input delays can be significant [55]. Developing control algorithms for demand response that are robust to delays and respect the limitations of existing infrastructure may lower the cost of demand response implementations.

Networked control theory addresses imperfect communication within control systems, see e.g., [56]. Ref. [57] investigates the impact of delays in frequency regulation including batteries and develops control algorithms to limit their effects. Within the demand response literature, [17, 40–43] develop control strategies to address infrequent or unavailable state measurements, [58] investigates lost messages in optimal load scheduling, and [59] investigates the impact of, but does not compensate for, communication latencies.

In this chapter, we develop non-disruptive control and estimation algorithms that enable aggregations of residential TCLs to provide ancillary services such as frequency regulation (i.e., secondary frequency control) in the presence of significant communication system limitations, including delays, as well as substantial error within the model used by the algorithms. In practice, we would expect large model mismatch since it is difficult to develop a computationally-tractable and accurate model of the aggregate dynamics of large number of spatially-distributed, heterogeneous TCLs, especially given that many useful TCL parameters and states are not easy to measure.

Our primary contribution is to adapt networked state estimation and control approaches so to that they can be used to solve key practical problems that will be encountered in cost-effectively coordinating large numbers of heterogeneous distributed TCLs for ancillary services. We propose two state estimation strategies, one that synthesizes estimates obtained from a bank of Kalman filters acting on non-synchronous state measurements and another that uses individual TCL state predictions obtained from identified TCL models as pseudo-measurements within a single Kalman filter. We also propose a model predictive control (MPC) algorithm that uses a probabilistic estimate of the control input.

This work builds upon and extends our preliminary work in [51]. The additional contributions of this work are as follows: 1) we make modifications to one of the control algorithms first proposed in [51]; 2) we track real PJM frequency regulation signals (rather than simple sinusoidal signals as in [51]) to evaluate the impact of delays on the adequacy of the frequency regulation from demand response; 3) we evaluate the control and estimation algorithms in tandem (rather than individually

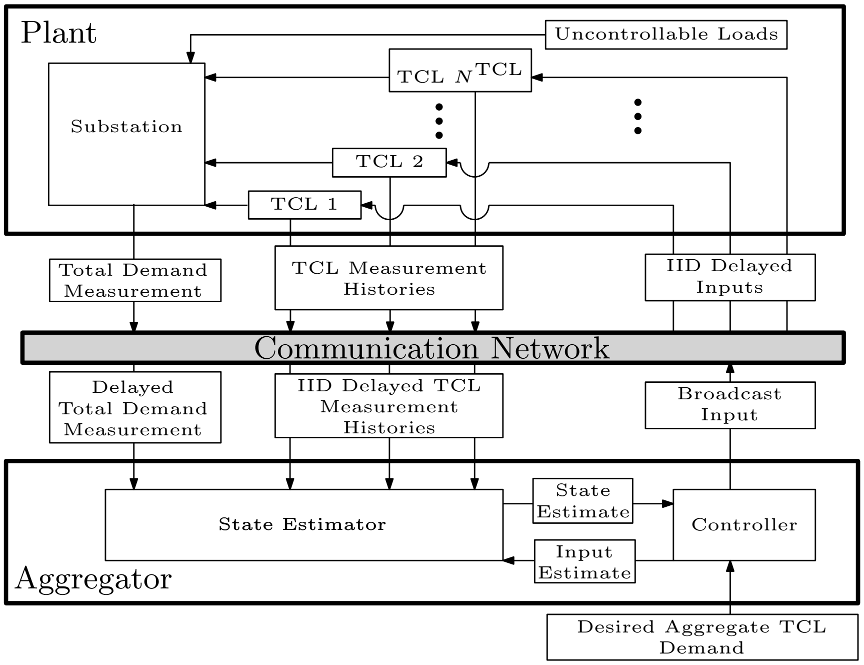


Figure 2.1: An overview of the problem framework of Chapter II.

as in [51]); 4) we evaluate the impact of modeling error by testing the algorithms on a more realistic simulated plant, and compare the results to those generated with the simpler plant used in [51]; 5) we find that both estimator-controller combinations can effectively mitigate communication delays; and 6) we find that one estimator is sensitive to the specific model used in the plant whereas the other is not.

The remainder of the chapter is organized as follows: Section 2.1 describes the problem framework. Section 2.2 summarizes individual and aggregate TCL modeling details. Section 2.3 develops two state estimation algorithms that we use in conjunction with a control algorithm that Section 2.4 develops. Section 2.5 formulates a number of case studies and presents their results. Finally, Section 2.6 presents the conclusions of the chapter.

2.1 Problem Framework

As shown in Fig. 2.1, we assume a problem framework that contains a plant, a communication network, and an aggregator. The plant consists of a set of N^{TCL} controllable TCLs, some uncontrollable loads, and a distribution substation that serves the total demand of the plant. We assume a smart meter acts as an interface between each TCL and the communication network, allowing two-way communication between the TCLs and the aggregator through an imperfect communication network as in [41].

We make several assumptions regarding the communication network and smart meters. Due to the capabilities of digital communication networks, we assume that multiple measurements and inputs can be transmitted within one communication packet, i.e., message. We also assume the messages are time-stamped [56] and the clocks are synchronized across the communication network nodes, allowing knowledge of previously realized delays and their resulting statistics. We assume that the communication network imposes independent and identically distributed (IID) delays on each message. We assume the smart meter can use logic to select an applicable input from a set of inputs, as explained in Section 2.4. Finally, we assume each smart meter can collect histories of the TCL's internal air temperature and on/off mode measurements, but the smart meter can only transmit these state measurement histories infrequently, e.g., every fifteen minutes, due to communication limitations as in [41].

We assume the aggregator, which acts as a central controller, uses a state estimator and controller, both of which include a model of the plant. The aggregator induces TCL on/off switching by broadcasting inputs at each time-step (every two seconds). The inputs are designed to produce a desired aggregate TCL demand, and the inputs are detailed in Section 2.2.2. IID delays cause the inputs to arrive asynchronously, and so an estimated input is used by the aggregator. We assume that the desired aggregate demand values are frequency regulation signals, e.g., automatic generation control (AGC) or secondary frequency control signals, provided by the system operator.

The aggregator's state estimation algorithm produces an estimate of the TCL aggregation's state, where the state is described in Section 2.2.2. We assume that the aggregator has access to measurements of the total substation demand and TCL state measurement histories as in [41]. As in [41], substation demand measurements are available at every time-step, and the aggregate TCL demand is estimated from these measurements by subtracting a prediction of the uncontrollable load. While the substation demand measurements may be accurate, errors in predicting the uncontrollable demand result in measurement noise on the aggregate TCL demand. In this work, we add normally-distributed, zero-mean noise to the aggregate TCL demand to approximate the prediction error and any noise in the substation demand measurements. As in [41] the TCL measurement histories are available infrequently (every 15 minutes) due to smart meter limitations.

The infrequent availability of possibly delayed TCL state measurements, means the aggregator relies on output feedback (i.e., the aggregate TCL demand estimates, referred to as "aggregate power measurements") at most time-steps to form the state estimate. The state estimate is then used by the aggregator to generate the control inputs.

2.2 Modeling

We use three previously developed models and describe them here for completeness and to establish the notation used throughout the chapter. Two hybrid, heat-transfer-based models represent individual TCLs, and a Markov chain model represents the TCL population.

The first individual TCL model, developed in [60, 61] and referred to as the three-state model, models household heating and cooling appliances using a mass temperature and an air temperature. The second, simpler TCL model, developed in [25, 27] and referred to as the two-state model, uses only an air temperature and can model all TCLs. We use the three-state model to represent the TCLs within the plant for the case studies described in Section 2.5 and we incorporate identified two-state model into the state estimator developed in Section 2.3.2.2.

The hybrid nature of these models, i.e., their usage of both discrete and continuous states, makes it computationally challenging to incorporate thousands of the models within optimization-based control algorithms. The Markov chain model, developed in [17] and referred to as the aggregate model, is a linear model of the TCL population's power demand dynamics, and it is easily incorporated into control algorithms. The algorithms in Sections 2.3 and 2.4 use this aggregate model.

The following section describes the individual TCL models, and Section 2.2.2 describes the aggregate model. Note that we assume model parameters are time-invariant throughout, but the models can incorporate time-varying parameters. Time-varying parameters within the individual TCL models would result in a time-varying aggregate TCL population model. Assuming that the time-varying aggregate model is known, all of the algorithms are still applicable.

2.2.1 Individual TCL Models

We first present a generic discrete-time state update equation for cooling TCLs below, then detail the difference between the two individual models in the following sections. Table 2.1 summarizes the model parameters where $[\alpha, \beta]$ corresponds to a uniform distribution and $\mathcal{N}(\alpha, \beta)$ corresponds to a normal distribution with mean α and variance β.

Table 2.1: TCL Model Parameters for Chapter II

Parameter	Description	Three-State Value	Two-State Value
θ^{set}	Temperature Set-Point [°C]	[23, 25]	[23, 25]
θ^{db}	Temperature Deadband [°C]	$[0.85, 1.15]$	$[0.85, 1.15]$
θ^{o}	Outdoor Temperature [°C]	32	32
U^{m}	Envelope Conductance $[\frac{\text{kW}}{°\text{C}}]$	[4.35, 5.87]	-
U^{a}	Internal Conductance $[\frac{\text{kW}}{°\text{C}}]$	[0.26, 0.35]	[0.41, 0.56]
Λ^{m}	Mass Heat Capacitance $[\frac{\text{kWh}}{°\text{C}}]$	[1.93, 2.60]	-
Λ^{a}	Air Heat Capacitance $[\frac{\text{kWh}}{°\text{C}}]$	[0.48, 0.64]	[0.51, 0.70]
Q^{m}	TCL Mass Heat Gain [kW]	0	-
Q^{a}	TCL Air Heat Gain [kW]	$\mathcal{N}(0, \epsilon)$	$\mathcal{N}(0, \epsilon)$
ϵ	Variance for Q^{a} [kW^2]	2.5E-7	2.5E-7
Q^{h}	TCL Heat Transfer [kW]	[-16, -12]	[-16, -12]
η	Coefficient of Performance [-]	3	3
Δt	Time-Step Duration [s]	2	2

Denote the set of TCLs $\mathcal{J}^{\text{TCL}} = \{1, 2, \ldots, N^{\text{TCL}}\}$. Each TCL $j \in \mathcal{J}^{\text{TCL}}$ has continuous-time matrix parameters $A^{\text{c},j}$, $B^{\text{c},j}$, and $E^{\text{c},j}$, and the discrete-time matrix parameters A^j, B^j, and E^j are formed using [62, p. 315]. The vector θ_t^j denotes the continuous states, which are the TCL's internal temperature(s) at time-step t. The discrete state corresponds to the scalar on/off mode m_t^j, and d_t^j is a disturbance vector. The discrete-time state-update equations are

$$\theta_{t+1}^j = A^j \theta_t^j + B^j m_t^j + E^j d_t^j \qquad j \in \mathcal{J}^{\text{TCL}} \tag{2.1a}$$

$$m_{t+1}^j = \begin{cases} 0 & \text{if } \theta_{t+1}^{a,j} < \theta^{\text{set},j} - \theta^{\text{db},j}/2 \\ 1 & \text{if } \theta_{t+1}^{a,j} > \theta^{\text{set},j} + \theta^{\text{db},j}/2 \\ m_t^j & \text{otherwise,} \end{cases} \qquad j \in \mathcal{J}^{\text{TCL}} \tag{2.1b}$$

where (2.1a) updates the internal temperatures, (2.1b) updates the on/off mode, and $\theta_t^{a,j}$ is the element of θ_t^j that corresponds to the TCL's air temperature, which is being regulated. The power demand of TCL j is $P_t^j = (|Q^{\text{h},j}|\ m_t^j)/\eta^j$ with $Q^{\text{h},j} < 0$ for cooling TCLs. The output of the TCL model, or the values that can be measured, is $y_t^j = \begin{bmatrix} \theta_t^{a,j} & m_t^j \end{bmatrix}^\top$.

2.2.1.1 Three-State Individual TCL Model

We use this model to represent individual TCLs within the plant during the case studies presented in Section 2.5. In the three-state model $\theta_t^j = \begin{bmatrix} \theta_t^{a,j} & \theta_t^{m,j} \end{bmatrix}^\top$ where $\theta_t^{m,j}$ denotes the TCL's mass temperature. The disturbance vector is $d_t^j =$

$\begin{bmatrix}\theta^{\mathrm{o}} & Q_t^{\mathrm{a},j} & Q^{\mathrm{m},j}\end{bmatrix}^\top$, where the heat injections $Q_t^{\mathrm{a},j}$ and $Q^{\mathrm{m},j}$ arise due to solar irradiance and heat gain within the household due to occupants and additional appliances. The model's continuous-time matrices are

$$\begin{aligned}
A^{\mathrm{c},j} &= \begin{bmatrix} -\left(U^{\mathrm{a},j}+U^{\mathrm{m},j}\right)/\Lambda^{\mathrm{a},j} & U^{\mathrm{m},j}/\Lambda^{\mathrm{a},j} \\ U^{\mathrm{m},j}/\Lambda^{\mathrm{m},j} & -U^{\mathrm{m},j}/\Lambda^{\mathrm{m},j} \end{bmatrix} \\
B^{\mathrm{c},j} &= \begin{bmatrix} Q^{\mathrm{h},j}/\Lambda^{\mathrm{a},j} & 0 \end{bmatrix}^\top \\
E^{\mathrm{c},j} &= \begin{bmatrix} U^{\mathrm{a},j}/\Lambda^{\mathrm{a},j} & 1/\Lambda^{\mathrm{a},j} & 0 \\ 0 & 0 & 1/\Lambda^{\mathrm{m},j} \end{bmatrix}.
\end{aligned}$$

Table 2.1's "Three-State Value" column parameterizes a population of residential air conditioners using nominal parameters from [63]. However, we set the outdoor temperature θ^{o} to simulate a reasonably hot day, we assume $Q_t^{\mathrm{a},j}$ is zero-mean and normally-distributed to include random air temperature disturbances as in [51], and we set $Q^{\mathrm{m},j} = 0$. The results in Section 2.5.2 include a discussion of the algorithms' ability to accommodate positively biased heat injections.

2.2.1.2 Two-State Individual TCL Model

We use these models within the estimator described in Section 2.3.2.2. In the two-state model $\theta_t^j = \theta_t^{\mathrm{a},j}$ and $d_t^j = \begin{bmatrix}\theta^{\mathrm{o}} & Q_t^{\mathrm{a}}\end{bmatrix}^\top$. The resulting continuous-time matrices are $A^{\mathrm{c},j} = -U^{\mathrm{a},j}/\Lambda^{\mathrm{a},j}$, $B^{\mathrm{c},j} = Q^{\mathrm{h},j}/\Lambda^{\mathrm{a},j}$ and $E^{\mathrm{c},j} = \begin{bmatrix}U^{\mathrm{a},j}/\Lambda^{\mathrm{a},j} & 1/\Lambda^{\mathrm{a},j}\end{bmatrix}$. We set the parameters in Table 2.1's "Two-State Value" column equal to the three-state model values where applicable, but we set $U^{\mathrm{a},j}$ and $\Lambda^{\mathrm{a},j}$ such that the nominal cycle time is comparable to that of the three-state model.

2.2.2 Aggregate TCL Population Model

The aggregate model, which is used by the controller and estimator, assumes that the two-state model of Section 2.2.1.2 is the underlying individual TCL model. While an aggregate model exists for the three-state TCL model [39], measurements of $\theta_t^{m,j}$ are not easy to obtain, and practical construction of the state-transition matrix from available measurements is an open question. In Section 2.5, we evaluate the impact of this assumption on the control and estimation algorithms' performance by simulating TCLs within the plant using the three-state model.

The aggregate model uses an aggregate state $x_t \in \mathbb{R}^{N^{\mathrm{x}}}$ where N^{x} is the number of discrete states, and each element of the state vector corresponds to the portion of

TCLs within the discrete state. The discrete states are formed by first defining a normalized temperature deadband and then dividing it into $N^{\mathrm{x}}/2$ temperature intervals. Each interval contains two states – one for TCLs that are drawing power and one for TCLs that are not. The state transition matrix $A \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{x}}}$ is a transposed Markov Transition Matrix that describes the probability of TCLs transitioning between states in a time-step.

The input $u_t \in \mathbb{R}^{N^{\mathrm{x}}/2}$ is the portion of TCLs that we want to force from the "on" bin of a temperature interval into the corresponding "off" bin or vice versa. The matrix $B \in \mathbb{R}^{N^{\mathrm{x}}/2 \times N^{\mathrm{x}}}$ ensures the TCLs are forced into the opposite on/off bin within the same temperature interval. Before transmitting input vectors to the TCLs, this input is converted into a switching probability by normalizing each input element with the corresponding state element. To implement these switching probabilities, each TCL first selects the probability corresponding to its current state. Then, each TCL determines whether it switches by drawing a random number. We assume the local TCL controller disregards switching requests when necessary to maintain the temperature within the normal operating range.

The output of the aggregate model y_t depends on whether both aggregate state and aggregate power measurements are available at time-step t. The set $\mathcal{T}^{\mathrm{S}}$ denotes time-steps where both aggregate state and aggregate power measurements are available, and $y_t \in \mathbb{R}^{N^{\mathrm{x}}+1}$ at these time-steps. Otherwise only aggregate power measurements are available and $y_t \in \mathbb{R}$. The resulting linear system is

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{2.3a}$$

$$y_t = \begin{cases} C^{\mathrm{P}} x_t + v_t^{\mathrm{P}} & t \notin \mathcal{T}^{\mathrm{S}} \\ \begin{bmatrix} C^{\mathrm{S}} \\ C^{\mathrm{P}} \end{bmatrix} x_t + \begin{bmatrix} v_t^{\mathrm{S}} \\ v_t^{\mathrm{P}} \end{bmatrix} & t \in \mathcal{T}^{\mathrm{S}}, \end{cases} \tag{2.3b}$$

where $w_t \in \mathbb{R}^{N^{\mathrm{x}}}$ is process noise including modeling error, $v_t^{\mathrm{S}} \in \mathbb{R}^{N^{\mathrm{x}}}$ is the aggregate state's measurement noise, $v_t^{\mathrm{P}} \in \mathbb{R}$ is the aggregate power's measurement noise, C^{S} is an $N^{\mathrm{x}} \times N^{\mathrm{x}}$ identity matrix, $C^{\mathrm{P}} = \overline{P}^{\mathrm{on}} N^{\mathrm{TCL}} [0 \ \cdots \ 0 \mid 1 \ \cdots \ 1]$, and $\overline{P}^{\mathrm{on}}$ is an approximation for the average power draw of a TCL that is on within the aggregation. We calculate $\overline{P}^{\mathrm{on}}$ using data from a set $\mathcal{T}^{\mathrm{hist}}$ of N^{hist} historical time-steps when the

TCLs cycled without external forcing

$$\overline{P}^{\text{on}} = \frac{1}{N^{\text{hist}}}\left(\sum_{t\in\mathcal{T}^{\text{hist}}}\frac{\sum\limits_{j\in\mathcal{J}^{\text{TCL}}} P_t^j}{\sum\limits_{j\in\mathcal{J}^{\text{TCL}}} m_t^j}\right). \tag{2.4}$$

The quantity $\sum_{j\in\mathcal{J}^{\text{TCL}}} P_t^j$ is the total power draw of TCLs at time-step t, and $\sum_{j\in\mathcal{J}^{\text{TCL}}} m_t^j$ is the number of TCLs that are on at time-step t. Finally, we assume that the aggregate model is known to the aggregator *a priori*. It can be derived or identified using the methods in [17].

2.3 State Estimation Algorithms

Our state estimation algorithms use a networked, time-varying Kalman filter from [64] that incorporates aggregate state and power measurements. The networked Kalman filter excludes measurements that have not arrived from the calculations at each time-step using binary indicator variables. As delayed measurements arrive, the networked Kalman filter fully incorporates the new information by updating a history of estimates. Measurement delays are treated deterministically within the estimator since delays associated with measurements that have arrived are known. Whereas [64] does not include inputs within the estimator's dynamic model, we use estimated inputs that are described in Section 2.4.2.

To incorporate delayed measurements, past estimator values must be stored so that they can be updated. The memory requirement can be reduced by excluding measurements with delays longer than a preset threshold; for generality, we do not set a delay threshold in this work. Setting a delay threshold within the estimator implies that measurements are discarded if they do not arrive by the deadline. As measurements take longer to arrive, their information becomes outdated and they are less useful. The threshold can be set given the known delay statistics such that only a small number of measurements are discarded due to late arrival and the impact on the state estimate is small. Section 2.3.1 presents the networked Kalman filter, and Section 2.3.2 presents two variations of it, which we apply to our problem.

2.3.1 The Networked Kalman Filter

Within this section and Section 2.4, we use the time indexing notation $\psi_{k|t}$ where ψ is an arbitrary value, and t denotes the time of the calculation. In this section, $k \leqslant t$

indexes a historical horizon of time-steps. The horizon length N_t^{kf} is set at each calculation time t as the number of time-steps of the newly-arrived measurements' largest delay, and the set of time-steps within the historical horizon is $\mathcal{K}_t^{\text{kf}} = \{t - N_t^{\text{kf}}, \ldots, t\}$. The set $\mathcal{K}_t^{\text{kf}}$ includes past time-steps requiring an update to incorporate the newly arrived measurements into the state estimate, and the present time-step for which a new state estimate must be generated.

The binary, scalar variables $\gamma_{k|t}^{\text{S}}$ and $\gamma_{k|t}^{\text{P}}$ indicate whether the aggregate state and aggregate power measurements sampled at time-step k have arrived by time-step t. The indicators are 0 if the corresponding measurement has not arrived by time-step t, and 1 otherwise. The indicator $\gamma_{k|t}^{\text{S}}$ is also set to 0 when the aggregate state is not measured. The Kalman filter observations incorporate the aggregate state measurements y_k^{S} and the aggregate power measurements y_k^{P} using $y_{k|t} = C_{k|t}x + v_{k|t}$ where

$$y_{k|t} = \begin{bmatrix} \gamma_{k|t}^{\text{S}}\, y_k^{\text{S}} \\ \gamma_{k|t}^{\text{P}}\, y_k^{\text{P}} \end{bmatrix}, \; C_{k|t} = \begin{bmatrix} \gamma_{k|t}^{\text{S}}\, C^{\text{S}} \\ \gamma_{k|t}^{\text{P}}\, C^{\text{P}} \end{bmatrix}, \text{ and } v_{k|t} = \begin{bmatrix} \gamma_{k|t}^{\text{S}}\, v_k^{\text{S}} \\ \gamma_{k|t}^{\text{P}}\, v_k^{\text{P}} \end{bmatrix}.$$

The measurement noise $v_{k|t}$ has covariance $V_{k|t}$, which is a block diagonal matrix composed of the aggregate state and aggregate power measurement noise covariances $\gamma_{k|t}^{\text{S}} V^{\text{S}}$ and $\gamma_{k|t}^{\text{P}} V^{\text{P}}$. These covariances are assumed to correspond to zero-mean, normal distributions. The indicator values ensure that measurements that have not arrived have no effect on the state estimate, and the resulting zero components of $y_{k|t}$, $C_{k|t}$, $v_{k|t}$, and $V_{k|t}$ can be removed to reduce the dimension of the computations with no effect on the estimate. The quantities $\widetilde{y}_{k|t}$, $\widetilde{C}_{k|t}$, $\widetilde{v}_{k|t}$, and $\widetilde{V}_{k|t}$ correspond to the reduced matrices.

To perform state estimation, we initialize the recalculation horizon using the state estimate and error covariance from the calculation at time $t-1$: $\widehat{x}_{t-N_t^{\text{kf}}-1|t} = \widehat{x}_{t-N_t^{\text{kf}}-1|t-1}$ and $H_{t-N_t^{\text{kf}}-1|t} = H_{t-N_t^{\text{kf}}-1|t-1}$. Previous state estimates are then recalculated to incorporate newly arrived measurements, and a new state estimate is generated for time-step $k = t$. For $k \in \mathcal{K}_t^{\text{kf}}$:

$$\widehat{x}_{k|t}^{-} = A\widehat{x}_{k-1|t} + B\widehat{u}_{k-1} \tag{2.5a}$$

$$H_{k|t}^{-} = AH_{k-1|t}A^{\top} + W \tag{2.5b}$$

$$K_{k|t} = H_{k|t}^{-}\widetilde{C}_{k|t}^{\top}\left(\widetilde{C}_{k|t}H_{k|t}^{-}\widetilde{C}_{k|t}^{\top} + \widetilde{V}_{k|t}\right)^{\dagger} \tag{2.5c}$$

$$\widehat{x}_{k|t} = \widehat{x}_{k|t}^{-} + K_{k|t}\left(\widetilde{y}_{k|t} - \widetilde{C}_{k|t}\widehat{x}_{k|t}^{-}\right) \tag{2.5d}$$

$$H_{k|t} = H_{k|t}^{-} - K_{k|t}\widetilde{C}_{k|t}H_{k|t}^{-}. \tag{2.5e}$$

The aggregate model in (2.5a) generates an *a priori* state prediction $\widehat{x}^{-}_{k|t}$ and error covariance $H^{-}_{k|t}$ in (2.5b), where W is the process noise covariance of a zero-mean, normal distribution. Section 2.4 explains the estimated input $\widehat{u}_k$ in more detail. The Kalman gain $K_{k|t}$ is calculated in (2.5c) based on available observation values with $\dagger$ denoting a pseudo-inverse, which we use for numerical reasons. The *a posteriori* state estimate $\widehat{x}_{k|t}$ and error covariance $H_{k|t}$ are then calculated in (2.5d) and (2.5e). The output of the algorithm is the new state estimate for the current time-step t, i.e., $\widehat{x}_t = \widehat{x}_{t|t}$, which fully incorporates all measurements that have arrived. Note that the process noise is not necessarily zero-mean and normally-distributed, which results in a sub-optimal filter. We assume that the measurement noise and process noise are independent from each other and independent in time.

2.3.2 Variations of the Networked Kalman Filter

Because of IID transmission delays, the state measurement histories from individual TCLs do not arrive synchronously at the aggregator. We develop and investigate the performance of two methods to form aggregate state estimates from asynchronous TCL measurements for use within the networked Kalman filter. The method in Section 2.3.2.1 describes an algorithm that runs N^{TCL} filters from Section 2.3.1, one for each TCL. The method in Section 2.3.2.2 uses a single networked Kalman filter that uses aggregate state *predictions* obtained by applying identified individual TCL models to old state measurements. Both methods use (delayed) aggregate power measurements that are sampled at every time-step and (delayed) TCL state measurements that sent to the aggregator infrequently, e.g., every fifteen minutes.

2.3.2.1 Estimator 1: Parallel Kalman Filter Estimator

An aggregate state measurement can be formed using each TCL's state measurement, i.e., its temperature and on/off mode measurements, and used within a networked Kalman Filter. However, this would require waiting until all TCL state measurements have arrived at the aggregator, and so the aggregate measurement delay would be equal to the worst-case TCL state measurement delay. Instead, we construct a state estimator that runs N^{TCL} networked Kalman filters in parallel, one for each TCL. When a TCL state measurement arrives, we use it within the corresponding Kalman filter and combine all filter estimates into a single aggregate state estimate. By doing this, we can update the aggregate state estimates while measurements are still arriving.

Each of the filters use the aggregate model, delayed aggregate power measurements, and delayed TCL state measurements from the corresponding TCL. While TCLs transmit measurement histories, only the most recent measurement is used in this method. We define the state vector of each filter as $x_t^j \in \mathbb{R}^{N^{\mathrm{x}}}$ for $j \in \mathcal{J}^{\mathrm{TCL}}$; it is defined equivalently to x_t in Section 2.2.2, however x_t^j models a single TCL. To form x_t^j, we convert TCL j's most recent air temperature and on/off value into its corresponding discrete bin value within the aggregate state. We then set the element of x_t^j corresponding to the TCL's discrete bin value to 1 and all other elements to 0. We assume TCL state measurements are accurate for convenience, and so we use a near-zero aggregate state noise covariance, i.e., $V^{\mathrm{S,j}} \approx 0 \, \forall \, j$. We use the the aggregate power measurement noise covariance defined in Section 2.3.1, and we assume that the covariance of this measurement noise is significantly greater than zero due to errors in predicting the uncontrollable demand as described in Section 2.1. The estimates produced by each filter are combined into an overall aggregate state estimate at each time-step as $\hat{x}_t = \left(\sum_{j=1}^{N^{\mathrm{TCL}}} \hat{x}_t^j \right) / N^{\mathrm{TCL}}$. The main disadvantages of this method are the large computational requirement of running N^{TCL} Kalman filters in parallel and the usage of only the most recent TCL measurements.

2.3.2.2 Estimator 2: Single Kalman Filter Using State Predictions

This state estimator consists of three components: an individual TCL parameter identification algorithm, a bank of N^{TCL} identified two-state individual TCL models as described in Section 2.2.1.2, and a single networked Kalman filter as described in Section 2.3.1. The parameter identification algorithm uses the TCL state measurement histories to identify the thermal parameters for each TCL, which we assume are initially unknown to the aggregator. We assume that measurements of thermal mass temperature are unavailable, as they would be difficult to obtain in practice. Therefore, we use the two-state individual TCL models, as it would be difficult or impossible to identify the three-state model parameters.

This estimator uses delayed aggregate power measurements and delayed TCL state measurement histories sampled at every time-step but transmitted infrequently, e.g., every fifteen minutes. As each TCL's measurement history arrives, it is used within a nonlinear least squares algorithm to identify the thermal parameters $\widehat{\Lambda}^{\mathrm{a},j}$ and $\widehat{U}^{\mathrm{a},j}$ corresponding to the two-state individual TCL model, assuming that the set-point, deadband width, and outdoor temperature are known. We use the identified models to predict the TCL states at each time-step (assuming $Q_t^{\mathrm{a,j}} = 0 \, \forall \, j$), and we use the individual predictions to form an aggregate state prediction $x_t^{\star}$.

A single networked Kalman filter treats the predictions $x_t^\star$ as measurements, allowing the individual TCL models to influence the Kalman filter estimate. The measurement noise associated with the aggregate state predictions v_t^{S} is assumed to be zero-mean and normally-distributed, and the aggregate state measurement noise covariance V^{S} is generated using the historical errors. Note that the noise will not be normally-distributed in general, which results in a sub-optimal filter. As in Section 2.3.2.1, the measurement noise associated with the aggregate power measurements is significant, and so the associated covariance is greater than zero.

The method has the disadvantage that it relies on the accuracy of the two-state model; the implications are discussed in Section 2.5. This estimator must keep track of and compute the state of N^{TCL} two-state models and requires one networked Kalman filter. Alternatively, the parallel Kalman filter estimator requires N^{TCL} networked Kalman filters. The amount of information, e.g., matrices and states, that must be stored for each Kalman filter depends on the aggregate model's dimension N^x. Qualitatively, the data requirements and necessary computations, i.e., matrix multiplication, for each estimator are similar.

2.4 Control Algorithm

The aggregator uses the predictive control approach described in [56] to counteract input delays. In this approach, the control algorithm generates an open-loop input sequence $U_t \in \mathbb{R}^{N^{\mathrm{x}}/2 \times N^{\mathrm{mpc}}}$ at each time-step based on the current state estimate. IID delays cause asynchronous input arrival at the TCLs. Using time-stamping, we assume that the smart meter (or TCL) can select the most recently generated input sequence that has arrived. The TCL then selects the input from that sequence that applies to the current time-step, or it uses a zero input. The controller does not know the implemented input at each TCL, and an estimated input is generated from the known delay statistics and is used within the aggregator's algorithms.

We develop an MPC algorithm that considers a horizon of N^{mpc} time-steps ranging from the present time-step t to future time-step $t+N^{\mathrm{mpc}}-1$ to generate U_t. The MPC algorithm uses the aggregate model to design inputs to track the desired aggregate demand $y_t^{P,\mathrm{ref}}$.

Within this section, k is used to indicate the time-step of the MPC horizon. Inputs corresponding to time-step k are produced at time $t = k-N^{\mathrm{mpc}}+1, \dots, k$, resulting in a total of N^{mpc} inputs for each time-step. The matrix $\mathcal{U}_k = \begin{bmatrix} u_{k|k} & \dots & u_{k|k-N^{\mathrm{mpc}}+1} \end{bmatrix}$ denotes the set of input vectors that apply to time-step k.

As in [65], we form an input estimate $\widehat{u}_t$ based on previously transmitted input sequences. Ref. [65] attempts to estimate the single input within an actuator whereas we form the input estimate $\widehat{u}_t$ as the weighted sum of possible inputs and their probability of being implemented. Specifically, we send a finite number of inputs that apply to time-step k to the TCLs. The inputs are known because the controller designs them. Using knowledge of the TCLs' input selection logic and the probability distribution of delays, which is assumed known from historical data, we compute the probability that each of the inputs that applies to time-step k is implemented by the TCLs, where $\mathcal{P}$ is the vector of probabilities. The MPC formulation uses the expected value of the input. This allows us to reformulate the stochastic optimization problem as a deterministic optimization problem [66]. Section 2.4.2 details the construction of $\widehat{u}_t$, $\mathcal{U}_k$, and $\mathcal{P}$. The following section presents the MPC formulation, which is a finite-horizon, linear quadratic output regulator with input and state constraints that is implemented using [67].

2.4.1 MPC Formulation

To set N^{mpc}, we first fix a parameter $p^{\max}$. The value $1 - p^{\max}$ is the probability that no valid input is available at the TCL, and we explain this further in Section 2.4.2. The MPC algorithm is initialized using the current state estimate $x_t = \widehat{x}_t$, the current aggregate demand request $y_t^{P,\text{ref}}$, and any previously transmitted inputs that apply to time-steps within the horizon. The full formulation is

$$\underset{u,\delta}{\text{minimize}} \sum_{k=t}^{t+N^{\text{mpc}}-1} \Big[c^y \left(y_k^{\text{err}}\right)^2 + c^\delta(\delta_k^- + \delta_k^+) + \sum_{m=k-N^{\text{mpc}}+1}^{k} c^u (u_{k|m}^\top \, u_{k|m}) \Big] \tag{2.6}$$

$$\begin{aligned}
\text{s.t.} \qquad x_{k+1} &= A\, x_k + B\, \widehat{u}_k && (2.7)\\
\widehat{u}_k &= \mathcal{U}_k \mathcal{P} && (2.8)\\
y_k^{\text{err}} &= y_k^{P,\text{ref}} - C^{\text{P}} x_k && (2.9)\\
u_{k|m}^i &\leqslant x_k^i \qquad i \in \{1, \ldots, N^{\text{x}}/2\} && (2.10)\\
-u_{k|m}^i &\leqslant x_k^{N^{\text{x}}+1-i} \qquad i \in \{1, \ldots, N^{\text{x}}/2\} && (2.11)\\
0 - \delta_k^- &\leqslant x_k \leqslant 1 + \delta_k^+ && (2.12)\\
0 &\leqslant \delta_k^-, \delta_k^+. && (2.13)
\end{aligned}$$

The objective function (2.6) minimizes the total cost over the horizon where the costs c^y, c^u, and c^δ penalize the tracking error y_k^{err}, control effort, and soft constraint violations δ_k^+ and δ_k^-. The input $u_{k|m}$ in (2.6) is a column of $\mathcal{U}_k$. The dynamic model in (2.7) corresponds to (2.3a) excluding the process noise and using the estimated input from (2.8). The tracking error is calculated in (2.9) using a persistent value of the current aggregate demand request, i.e., $y_k^{P,\text{ref}} = y_t^{P,\text{ref}}$ for all k in the MPC horizon. The input constraints (2.10) and (2.11) limit the fraction of TCLs to switch from a particular bin to be less than the fraction of TCLs within that bin. The soft state constraint (2.12) is satisfied regardless of the initial value provided from the unconstrained state estimator, and (2.13) restricts the soft constraint violations to positive values.

2.4.2 Constructing Input Estimates

The vector $\mathcal{P}$ weights each of the inputs in $\mathcal{U}_k$ based on their probability of being implemented. The probabilities within $\mathcal{P}$ are fixed during an MPC calculation. However, the probabilities could be recomputed between MPC calculations if the delay distribution changes, e.g., due to different traffic levels at various times of the day.

The inputs in $\mathcal{U}_k$ become "older" as we go from left to right in the matrix, and we calculate the elements of $\mathcal{P}$ using the corresponding input vector's location in $\mathcal{U}_k$. Due to its input selection logic, a TCL uses the input corresponding to the leftmost column of $\mathcal{U}_k$ that has arrived. The probability of using a column depends on two events: 1) the column must have arrived, and 2) every column to its left within $\mathcal{U}_k$ must not have arrived. We generate the elements of $\mathcal{P}$ based on these two events, the necessary delays for these events to occur, and the probability of realizing these delays. Note that while assuming IID delays simplifies the following calculations, they are still possible without independence.

To simplify the notation in the following discussion, denote the ith element of $\mathcal{P}$ as p^i, the ith column of $\mathcal{U}_k$ as u^i (which was previously denoted $u_{k|k-i+1}$), the delay in time-steps associated with the arrival of column u^i as τ^i, and the probability of the first and second necessary events as $p^{1,i}$ and $p^{2,i}$. The delays take non-negative, integer values. Using the assumption of IID delays, $p^i = p^{1,i}p^{2,i}$.

We say that the input $u^i = u_{k|k-i+1}$ arrives within time-step k if its delay is less than i

$$p^{1,i} = p(\tau^i < i) \qquad i = 1, \ldots, N^{\text{mpc}}. \tag{2.14}$$

For example, $u^1 = u_{k|k}$ arrives within time-step k if $\tau^1 < 1$, i.e., $\tau^1 = 0$, and $u^2 = u_{k|k-1}$ arrives within time-step k if $\tau^2 < 2$, i.e., $\tau^2 = 0$ or $\tau^2 = 1$.

The probability that u^i has not arrived by time-step k is $p(\tau^i \geqslant i)$. Assuming IID delays, the probability that all columns left of column i have not arrived by time-step

Table 2.2: Simulation Parameters for Chapter II

Parameter	Description	Value
N^{x}	Number of State Bins [-]	100
N^{TCL}	Number of TCLs [-]	10,000
R^{P}	Aggregate Power Noise Covariance [kW^2]	$\mathcal{N}(0, 4\text{E}6)$
P^{avg}	Average Steady-State TCL Demand [kW]	6E3
Δt	Time-Step Duration [s]	2
$\Delta t^{S,s}$	TCL State Measurement Interval [s]	2
$\Delta t^{S,t}$	TCL State History Transmission Interval [s]	900
n^{steps}	Time-Steps in Simulation [-]	1800
p^{max}	MPC Delay Probability Threshold [-]	0.999
c^{y}	MPC Output Cost [-]	1
c^{u}	MPC Input Cost [-]	1
c^{δ}	MPC Soft Constraint Cost [-]	1.01

k is

$$p^{2,i} = \prod_{n=1}^{i-1} p(\tau^n \geqslant n) \qquad i = 1, \ldots, N^{\mathrm{mpc}}. \tag{2.15}$$

For example, $u^3 = u_{k|k-2}$ can only be used if u^1 and u^2 have not arrived by time-step k, i.e., $\tau^1 \geqslant 1$ and $\tau^2 \geqslant 2$.

For an MPC calculation, columns within $\mathcal{U}_k$ whose right time index is less than t correspond to inputs that have already been sent to the TCLs. The remaining columns are inputs that the MPC algorithm chooses (i.e., decision variables within the optimization problem (2.6)–(2.13)), but only a portion of these are included in the input sequence U_t sent to the TCLs after the MPC calculation. Specifically, U_t consists of the one column from each $\mathcal{U}_k$, $k \in \{t, \ldots, t + N^{\mathrm{mpc}} - 1\}$, whose right-hand time index corresponds to the current time t, i.e., every $u_{a|b}$ with $b = t$. The horizon length N^{mpc} is set such that the sum of elements in $\mathcal{P}$ is greater than p^{max} where N^{mpc} is the length of $\mathcal{P}$.

2.5 Case Studies

We summarize a series of simulations investigating 1) the impact of compensating for delays, and 2) the ability of the methods to provide frequency regulation despite communication delays and model error. Section 2.5.1 details the simulation parameters, algorithm combinations, delay distributions, reference signal construction, and quantities used to evaluate the simulations. Section 2.5.2 presents the simulation results.

2.5.1 Case Study Setup

Table 2.2 details the simulation parameters. We simulate a TCL population of 10,000 air conditioners. The average steady-state TCL demand is slightly different between the two- and three-state TCL model populations because of the parameters used, and the value in the table is approximate. We use a zero-mean, normal distribution to generate the aggregate power measurement noise. Similar to [17, 51], we set the noise variance assuming the average steady-state TCL demand is 15% of the demand served by the substation, and the standard deviation of the power measurement noise is set to 5% of the total substation load. We generate the aggregate model's process noise covariance using historical errors. The resulting process noise is neither zero-mean nor normally-distributed.

We conduct a series of case studies, varying: 1) the average delay, 2) the reference signal, 3) the model used to simulate individual TCLs within the plant, and 4) the estimator. The two reference signals, called the Reg-A and Reg-D reference, correspond to segments of published PJM traditional and dynamic frequency regulation signals from [68]. The reference signals are from May 4, 2014. The signals are interpolated to two second time-steps, and each signal is scaled so that the maximum demand change request corresponds to ±20% of the average steady-state aggregate TCL demand. Three delay distributions are used, referred to by their average delay: 0, 10, and 20 seconds. With average delays of 0, we do not impose any measurement or input delays. With average delays of 10 and 20 seconds, IID delays τ are sampled from a discretized log-normal distribution $\tau = \lfloor \exp(\tau^\star) \rfloor$ where $\lfloor \cdot \rfloor$ rounds down and $\tau^\star$ is normally-distributed. In the cases with delays, the variance of the log-normal distribution is 0.25 sec^2. We use the chosen distribution to model delays that do not take negative values and are unbounded above. Recall that the estimator treats the delays deterministically after each measurement has arrived, and so the delay distribution does not impact the estimator performance. The controller uses the statistics of the delay distribution, and so alternative distributions would change the entries in $\mathcal{P}$, but these can be computed regardless of the distribution.

We compare four control setups, which vary both the complexity of delay compensation method and the estimator used.

- Estimator 1-FC, where FC refers to "full compensation," pairs the controller from Section 2.4 with Estimator 1. This setup requires measurement and input time-stamping, aggregator knowledge of the input delay statistics, and that TCLs are capable of input selection.

- Estimator 2-FC, pairs the controller from Section 2.4 with Estimator 2. This setup requires measurement and input time-stamping, aggregator knowledge of the input delay statistics, and that TCLs are capable of input selection.

- Estimator 1-TS, where TS refers to "time stamping," pairs Estimator 1 with a simplified controller that does not require aggregator knowledge of the input delay statistics, but still requires measurement and input time-stamping. The controller assumes that there are no input delays and sets the probability of the input arriving at the TCL within the time-step to 1, i.e., $\mathcal{P} = 1$. Additionally, it uses only one time-step within the MPC algorithm, i.e., $N^{\text{mpc}} = 1$ because this is all that is necessary without delays [44] and the reference signal is unknown in future time-steps. Time-stamping enables measurement delay compensation in the estimator and input selection at the TCLs.

- Estimator 1-NC, where NC refers to "no compensation," pairs Estimator 1 with a simplified controller that does not require aggregator knowledge of the input delay statistics *and* assumes measurements/inputs are not time-stamped. As in Estimator 1-TS, the controller does not account for input delays. Since measurements are not time-stamped, the estimator uses measurements based on their arrival time rather than their sampling time. Since inputs are not time-stamped, TCLs use whichever input arrives first during a time-step or zero if no input has arrived.

We quantify the results using two values 1) the normalized RMS tracking error (RMSE), and 2) the PJM score described below. We run ten instances of each case with different realizations of the random quantities and average the RMSE and PJM score across the instances for each case. The RMSE for a single case instance is

$$P^{\text{RMSE}} = \frac{1}{P^{\text{avg}}} \sqrt{\frac{1}{n^{\text{steps}}} \sum_{t \in \mathcal{T}} \left(y_t^{P,\text{real}} - y_t^{P,\text{ref}} \right)^2} \tag{2.16}$$

where P^{avg} is the average steady-state aggregate TCL demand and $y_t^{P,\text{real}}$ is the achieved aggregate demand. The PJM score is a value between 0 and 1 that is calculated using PJM's standards [69, pp. 52-54]. The score uses the correlation, delay, and difference in energy between the requested and actual signals. A passing score is $\geqslant 0.75$, which could certify a resource to provide frequency regulation when tracking the PJM test signal. A score of $\geqslant 0.50$ would be satisfactory to maintain certification for an arbitrary reference signal.

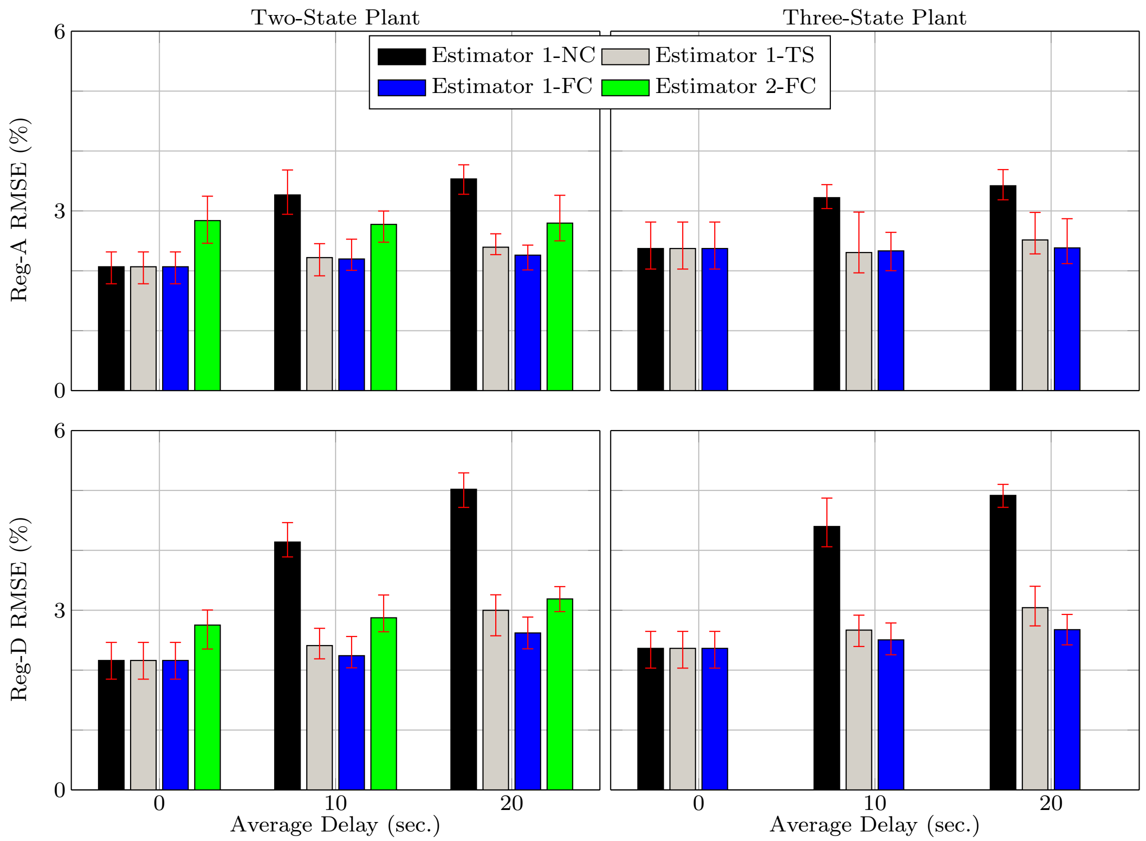
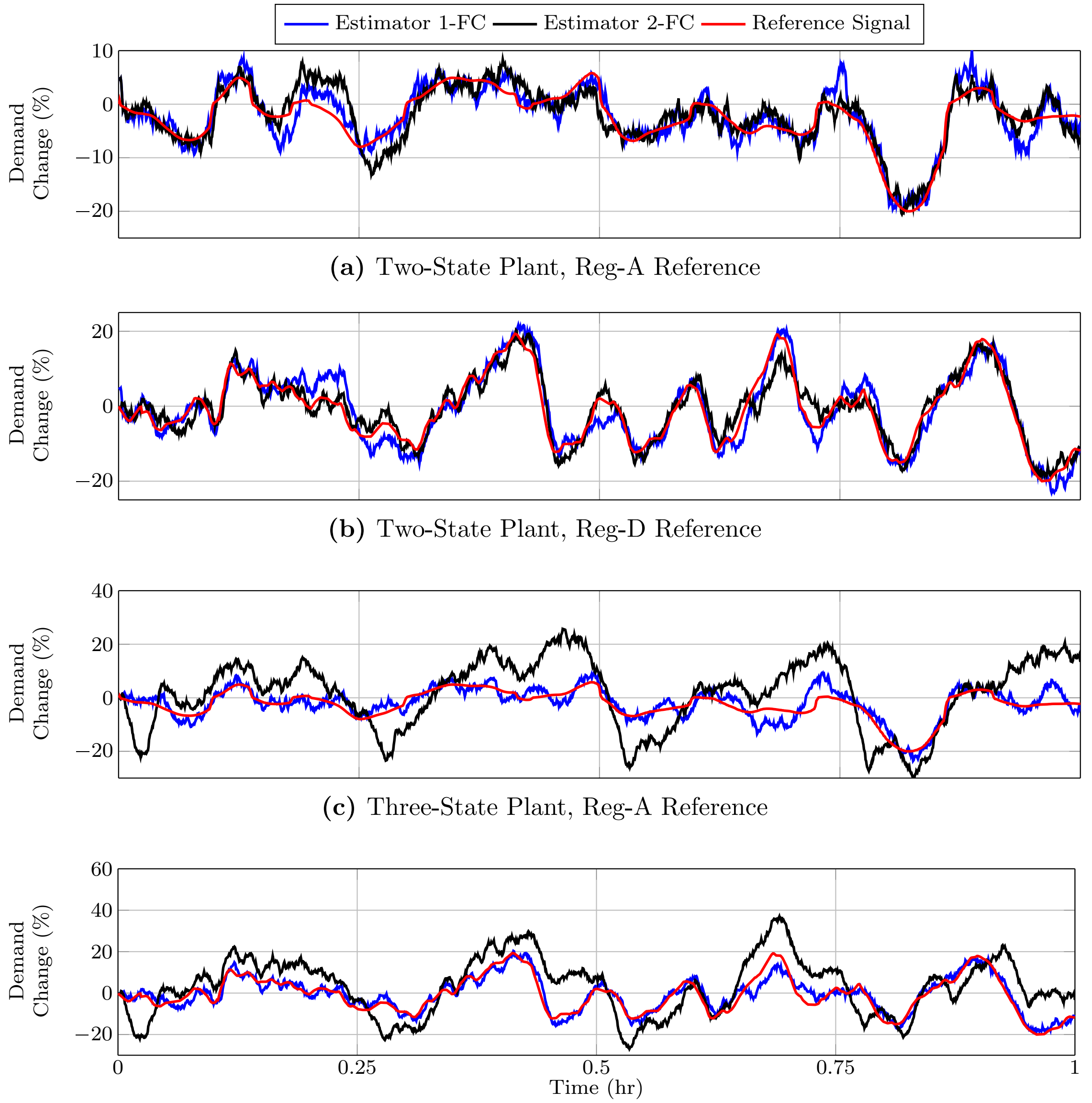


Figure 2.2: The average RMSE across all simulation scenarios varying the plant (two-state and three-state TCL models), control setup, average delay, and reference (Reg-A and Reg-D). Error bars indicate the range of values achieved across the ten instances of each scenario.

2.5.2 Simulation Results

Figure 2.2 summarizes the average RMSE for the four control setups, Table 2.3 summarizes the average PJM scores, and Fig. 2.3 provides time series for Estimators 1-FC and 2-FC assuming an average delay of 20 seconds. Note that Fig. 2.2 does not contain results for cases using Estimator 2-FC and three-state models as the plant. As shown in Fig. 2.3c and 2.3d, the resulting RMSE is high ($\sim 15-20\%$) in these cases, and we explain the cause below.

Note that with no delay, the RMSE and PJM scores of Estimators 1-NC, 1-TS, and 1-FC are all identical in Fig. 2.2 and Table 2.3 since all three methods are equivalent without delay. However, with increasing average delays, the RMSE increases significantly for Estimator 1-NC, increases slowly for Estimator 1-TS, and is roughly constant for Estimator 1-FC. This indicates that i) delay compensation is needed if delays are significant, ii) a simple compensation method reduces the effects of delays, and iii) more complex methods can further mitigate the effects of delays.



(a) Two-State Plant, Reg-A Reference

(b) Two-State Plant, Reg-D Reference

(c) Three-State Plant, Reg-A Reference

(d) Three-State Plant, Reg-D Reference

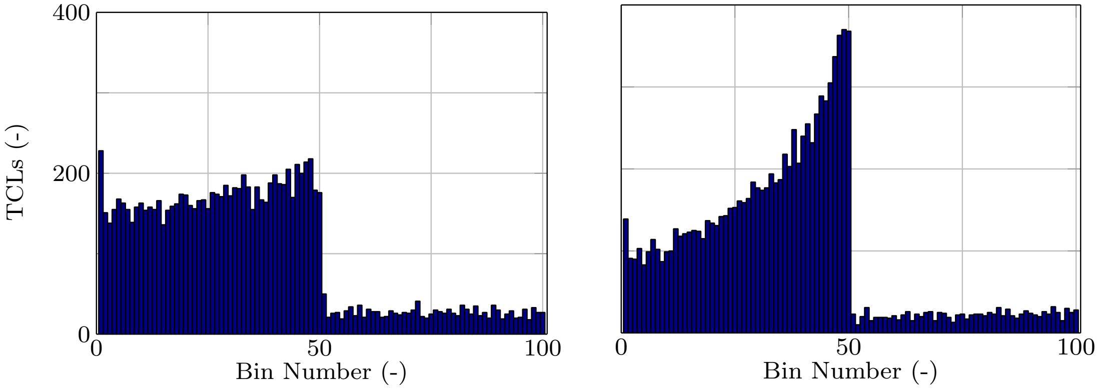
Figure 2.3: Time-series plots comparing the reference tracking of Estimator 1 and Estimator 2 with average delays of 20 seconds.

Estimator 2-FC generally performs worse than Estimator 1-TS, and we discuss this below. Note that all methods produce average PJM scores over the 0.75 threshold, except Estimator 1-NC when used to provide Reg-A with average delays of 10 and 20 seconds.

Focusing on Estimator 1-FC, PJM scores for the Reg-D reference are slightly better than those for the Reg-A reference, while the RMSE is slightly worse for the Reg-D signal. Because the trends are different for each performance metric, it is unclear whether the Reg-A or Reg-D cases are superior. Using the three-state TCL

Table 2.3: Average PJM Scores [-]

		Two-State Plant		Three-State Plant	
Control Setup	Avg. Delay	Reg-A	Reg-D	Reg-A	Reg-D
Estimator 1-NC	0	0.852	0.899	0.810	0.894
	10	0.727	0.821	0.735	0.835
	20	0.707	0.782	0.727	0.805
Estimator 1-TS	0	0.852	0.899	0.810	0.894
	10	0.846	0.908	0.821	0.901
	20	0.836	0.892	0.801	0.896
Estimator 1-FC	0	0.852	0.899	0.810	0.894
	10	0.849	0.908	0.814	0.901
	20	0.849	0.903	0.824	0.903
Estimator 2-FC	0	0.785	0.882	——	——
	10	0.799	0.892	——	——
	20	0.807	0.887	——	——



(a) Two-State Plant

(b) Three-State Plant

Figure 2.4: Comparison of discrete state bin distributions in steady-state for populations of TCLs represented by two-state and three-state models.

models within the plant resulted in slightly worse RMSE and PJM scores, but they are still above PJM's threshold, which is important since three-state models capture the TCL dynamics more accurately. While we used zero-mean heat injections to generate all of the results shown in this paper, simulations results (not shown here due to space limitations) indicate that Estimator 1-FC can adequately handle more realistic, biased heat injections. In contrast, Estimator 2-FC is unable to account for biased heat injections.

Estimator 2-FC's performance is dependent on the TCL model used within the plant. When two-state models are used within the plant, Estimator 2-FC's PJM scores are acceptable in all scenarios; however, it performs worse than Estimators 1-TS and 1-FC. Estimator 2-FC's performance is dependent on the number of discrete states N^{x}, and diminishes if N^{x} is reduced, e.g., to 40, for computational reasons.

Improvements to Estimator 2-FC may be achievable with more advanced parameter estimation methods.

When the three-state models are used within the plant, Estimator 2-FC is not able to provide effective frequency regulation because Estimator 2 uses two-state models to generate aggregate state predictions, which are treated as measurements within the state estimator. Figure 2.4 shows the steady-state distribution of TCLs within 100 discrete state bins using the two- and three-state plants. The distribution in the two-state plant (left) is fairly flat across the first 50 bins, which correspond to TCLs that are off. Alternatively, the distribution in the three-state plant (right) has a large concentration of TCLs around bin 50, which corresponds to the edge of the deadband. This qualitative difference in their distributions means that the two-state models used in Estimator 2 cannot effectively predict the aggregate state when the plant consists of three-state models. Because the thermal mass temperature and environmental heat injections are difficult to measure, identifying the three-state model is difficult.

2.6 Chapter II Conclusions

In this chapter, we developed a predictive controller and two estimators to mitigate the effects of communication delays within a residential demand response scenario. In simulations, we investigated the ability of the algorithms to control an aggregation of TCLs to track frequency regulation signals. Results show that both estimator-controller combinations are able to effectively provide frequency regulation with average delays of up to 20 seconds. The first estimator, which includes only an aggregate model and relies on a number of Kalman filters running in parallel is effective with both structural- and parameter-based modeling error. The second estimator assumes a specific TCL model and identifies parameters for those models. It is effective if the assumed TCL model structure matches the true model structure.

Future work should incorporate and address time-varying outdoor temperatures and time-varying, biased heat injections into the individual TCL models. Investigating a more effective parameter identification algorithm may allow Estimator 2 to be used in a wider range of scenarios. Modifying the controller to account for modeling error may improve tracking performance. Finally, accounting for non-normally-distributed measurement noise should be investigated.

CHAPTER III

A Linear Approach to Manage Input Delays While Supplying Frequency Regulation Using Residential Loads [1]

Demand response refers to the manipulation of the electric power usage, or demand, of devices to provide some operational benefit to an electric power network. These benefits can include both peak load reduction and participation in frequency regulation [37]. Peak load reduction limits the electricity demand during a period of interest with a goal of reducing the system's operating costs or improving the system's reliability [37]. Demand response for frequency regulation manipulates the electric power demand to help balance the supply and demand of electricity, maintaining the power network's frequency near its operating point. Frameworks for implementing demand response include both price-based and direct control schemes [53]. Price-based approaches encourage or discourage additional demand by adjusting the price of power. In contrast, direct control methods manipulate the state of the loads, e.g., switching a device on or off.

In this chapter, we use non-disruptive on/off switching of residential thermostatically controlled loads (TCLs) to provide frequency regulation. Residential TCLs are household loads such as air conditioners, water heaters, and heat pumps that draw electricity to maintain the temperature of an internal medium, e.g., a house's air temperature, about a user-defined set-point. These loads periodically switch between an on mode, where the device draws power, and an off mode, where it does not, to keep the medium's temperature within a dead-band around the set-point. While some residential demand response schemes manipulate user-defined set-points, non-disruptive demand response [37] respects the users' temperature settings while

[1] The work presented in this chapter was originally published in [70].

imposing additional on/off switching on the device.

There exists a large potential capacity of residential TCLs for demand response due to their widespread usage, and smart meters can enable communication between a central controller and TCLs. However, practical issues associated with implementing residential demand response include the high cost of the sensing and communication infrastructure due to the resource's spatially distributed nature. Developing demand response control algorithms that cope with imperfect communication systems may allow lower cost communication networks, e.g., existing legacy equipment, to become viable. Also, reducing the sensing requirements of these algorithms can reduce sensor costs.

Existing smart meters have communication limitations [49], e.g., infrequent data transmission, and demand response state estimation approaches have been developed to cope with unavailable measurements [17, 40, 43, 71]. Networked control is a class of control that addresses imperfect communication between components within the control system, see e.g., [56]. Demand response literature incorporating these concepts include [58], which investigates lost messages in optimal load scheduling, and [51], which adapts networked control algorithms and uses infrequent state measurements. Ref. [59] characterizes the effects of (but does not compensate for) communication latencies. Finally, [72] considers the required rate of communication to enable their demand control mechanism, [73] investigates the cost of generation during economic dispatch where packet loss influences the uncertainty of the demand estimate, and [74, 75] investigate the impact of packet loss on demand response control.

In this chapter, we develop a linear controller that accounts for input delays, and we compare it to the model predictive control (MPC) algorithm from [51], presented in the previous chatper, that also accounts for input delays. We account for input delays within both methods by i) including the delay statistics, which are assumed to be known, within the control algorithms, ii) generating a sequence of inputs at each time-step rather than an individual input, and iii) allowing the TCLs to select inputs based on the realized delays, which are assumed to be known by the TCLs based on the capabilities of digital communication networks. While our controller uses off-the-shelf linear control techniques, the challenge is to model the large-scale hybrid system subject to input delays as a compact linear system amenable these techniques. For this, we extend the linear aggregate TCL modeling approach in [17], detailed in Section 3.2.2. We present a novel model-reduction method that produces a reduced-order model similar to that in [76] but provides additional insights. We then use state augmentation to capture the impact of input delays. We present case studies for both the linear and MPC controllers to compare computation times and tracking performance under several delay scenarios. The benefits of the linear controller over the MPC controller are reduced online computation time and a simple, closed-form control law. Drawbacks of the linear controller compared to the MPC controller include increased tracking error and the inability to enforce constraints that the MPC formulation includes explicitly.

The remainder of the chapter is organized as follows: Section 3.1 describes the problem setting, Section 3.2 details the models used within this work, Section 3.3 develops the control algorithms, Section 3.4 describes the case studies and summarizes the results, and Section 3.5 presents the conclusions of the chapter.

3.1 Problem Setting and Overview

Figure 3.1 provides an overview of the problem setting. We assume two-way communication is possible between a population of residential TCLs and an aggregator, which interfaces between a power system operator and a population of loads. The aggregator can manipulate the total power demand of a TCL population to track a desired aggregate power signal. The desired aggregate power signal is generated by a system operator, and we assume it corresponds to a frequency regulation signal.

To manipulate the aggregate TCL demand, the aggregator broadcasts an input signal to all TCLs within the population. The inputs, detailed in Section 3.2.2, induce on/off switching, and the inputs are updated and broadcast at intervals of seconds. Communication delays cause the inputs to arrive asynchronously at the TCLs, and the implemented input at an individual TCL is not known by the aggregator in real-time. Finally, TCLs transmit their individual on/off modes and internal air temperatures (which comprise the "TCL state measurement") to the aggregator at each time-step, providing aggregate state information to the controller.

This work focuses on comparing the ability of controllers to mitigate input delays, and so we neglect communication limitations and delays associated with the TCL state measurements, which were considered in [51]. As a result, we assume TCL state measurements are available at every time-step and without delay. Realistically, state measurements are available infrequently, e.g., due to smart meter limitations, or not at all, and they may also be delayed if they are available. Similarly, output measurements may be delayed. Therefore, state estimators would be needed. Future work will address this.

Our assumptions regarding the communication network include synchronized clocks

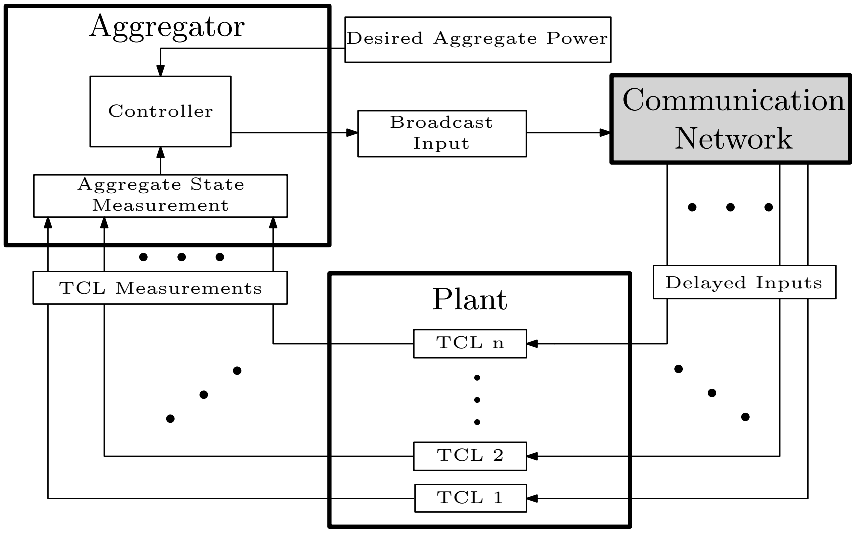


Figure 3.1: An overview of the problem setting for Chapter III.

across the network, the ability to transmit multiple values in a single message [56], the ability to time-stamp messages [56], and input delays that are independent and identically distributed (IID). We assume time-stamping enables the aggregator to know the delay statistics, and so we use these statistics within the controller.

The aggregator's control algorithms incorporate the predictive control approach described in [56] to construct an open-loop input sequence at each time-step. The input sequence is broadcast to the TCLs, and we assume the TCLs use selection logic and time-stamping to implement the most recently generated input vector that applies to a given time-step. The controller uses stochastic programming concepts to account for previously transmitted inputs by combining them into a weighted combination where the weights are generated based on the delay statistics. The following section discusses the models used within the work before incorporating them within the control algorithms detailed in Section 3.3.

3.2 Modeling

Several models are used within this work. The individual TCL model developed in [60] and described in Section 3.2.1 represents the TCLs within the simulated plant. It models the heat transfer driving the duty cycle of each residential TCL using discrete and continuous states. A linear time invariant aggregate model developed in [17] and described in Section 3.2.2 is a probability-based model that captures the behavior of the TCL population with reduced complexity. It is used within the MPC algorithm. Section 3.2.3 derives a reduced-order aggregate model similar to [76]. The individual TCL model is the same as that described in Section 2.2.1.1, and the aggregate model was described in Section 2.2.2; we describe them below for completeness and to introduce notation. It allows usage of linear quadratic regulator (LQR) techniques in designing the linear controller.

3.2.1 Individual TCL Model

In this chapter, the individual TCL model represents each of the N^{TCL} residential air conditioners within the controllable load population. Table 3.1 provides the notation, description, and approximate distributions of the individual model's parameters within this work. The nominal parameters are based on [63] with two exceptions: θ^{o} and $Q_t^{\mathrm{a},i}$. The outdoor temperature θ^{o} is set to be a reasonably hot day, and $Q_t^{\mathrm{a},i}$ includes Gaussian disturbances as in [51]. Values sampled from uniform and normal distributions are denoted $[\alpha, \beta]$ and $\mathcal{N}(\alpha, \beta)$, respectively. In the latter, α and β are the mean and variance of the normal distribution.

The parameter distributions in Table 3.1 are randomly sampled to generate the population of TCLs, and we denote the set of TCLs as $\mathcal{I}^{\mathrm{TCL}} = \{1, 2, \ldots, N^{\mathrm{TCL}}\}$. We use $i \in \mathcal{I}^{\mathrm{TCL}}$ to index an arbitrary TCL from the set. Each TCL contains three states – its internal air temperature $\theta_t^{a,i}$, its internal mass temperature $\theta_t^{m,i}$, and its current on/off mode m_t^i. We define the internal temperature vector as $\theta_t^i = \begin{bmatrix}\theta_t^{a,i} & \theta_t^{m,i}\end{bmatrix}^T$. A disturbance vector $d_t^i = \begin{bmatrix}\theta_t^{\mathrm{o}} & Q_t^{\mathrm{a},i} & Q_t^{\mathrm{m},i}\end{bmatrix}^T$ captures the exogenous, environmental inputs that influence the TCL's on/off cycling where $Q_t^{\mathrm{a},i}$ and $Q_t^{\mathrm{m},i}$ capture heating from loads and occupants within the house as well as solar irradiance. Finally, the TCL state measurement includes the TCL's air temperature and on/off mode $y_t^{\mathrm{TCL},i} = \begin{bmatrix}\theta_t^{a,i} & m_t^i\end{bmatrix}^T$.

The model's discrete-time state-update equations are

$$\theta_{t+1}^i = A^i\theta_t^i + B^i m_t^i + E^i d_t^i \qquad i \in \mathcal{I}^{\mathrm{TCL}} \tag{3.1a}$$

$$m_{t+1}^i = \begin{cases} 0 & \text{if } \theta_{t+1}^{a,i} < \theta^{\mathrm{set},i} - \theta^{\mathrm{db},i}/2 \\ 1 & \text{if } \theta_{t+1}^{a,i} > \theta^{\mathrm{set},i} + \theta^{\mathrm{db},i}/2 \\ m_t^i & \text{otherwise} \end{cases} \qquad i \in \mathcal{I}^{\mathrm{TCL}} \tag{3.1b}$$

where (3.1a) updates the internal temperatures and (3.1b) updates the on/off mode. The power draw is $P_t^i = (|Q^{\mathrm{h},i}|\, m_t^i)/\eta^i$ with $Q^{\mathrm{h},i} < 0$ for cooling loads. The matrices in these equations are discretized using [62, p. 315] where the underlying continuous-

Table 3.1: TCL Model Parameters for Chapter III

Parameter	Description	Value
Δt	Time-Step Duration [s]	2
θ^{set}	Temperature Set-Point [°C]	[24, 26]
θ^{db}	Temperature Dead-band [°C]	$[1.9, 2.2]$
θ^{o}	Outdoor Temperature [°C]	32
U^{m}	Envelope Conductance $[\frac{\mathrm{kW}}{°\mathrm{C}}]$	[0.89, 1.09]
U^{a}	Internal Conductance $[\frac{\mathrm{kW}}{°\mathrm{C}}]$	[0.2, 0.25]
Λ^{m}	Mass Heat Capacitance $[\frac{\mathrm{kWh}}{°\mathrm{C}}]$	[4.75, 5.80]
Λ^{a}	Air Heat Capacitance $[\frac{\mathrm{kWh}}{°\mathrm{C}}]$	[0.16, 0.20]
Q^{m}	TCL Mass Heat Gain [kW]	$\mathcal{N}(\overline{Q}, 0)$
Q^{a}	TCL Air Heat Gain [kW]	$\mathcal{N}(\overline{Q}, 2.5\text{E-}9)$
$\overline{Q}$	Heat Gain Distribution Mean	[0.45, 0.55]
Q^{h}	TCL Heat Transfer [kW]	[-17.0, -13.8]
η	Coefficient of Performance [-]	3

time matrices are

$$A^{\mathrm{c},i} = \begin{bmatrix} -\left(U^{\mathrm{a},i} + U^{\mathrm{m},i}\right)/\Lambda^{\mathrm{a},i} & U^{\mathrm{m},i}/\Lambda^{\mathrm{a},i} \\ U^{\mathrm{m},i}/\Lambda^{\mathrm{m},i} & -U^{\mathrm{m},i}/\Lambda^{\mathrm{m},i} \end{bmatrix}$$

$$B^{\mathrm{c},i} = \begin{bmatrix} Q^{\mathrm{h},i}/\Lambda^{\mathrm{a},i} & 0 \end{bmatrix}^T$$

$$E^{\mathrm{c},i} = \begin{bmatrix} U^{\mathrm{a},i}/\Lambda^{\mathrm{a},i} & 1/\Lambda^{\mathrm{a},i} & 0 \\ 0 & 0 & 1/\Lambda^{\mathrm{m},i} \end{bmatrix}.$$

The average cycle time of the discrete-time model using the parameters in Table 3.1 is 10 minutes with a 20% duty cycle.

3.2.2 Aggregate TCL Population Model

The aggregate model [17] captures the power draw behavior of the TCL population with reduced modeling complexity, and we summarize it here for completeness. The model is

$$x_{t+1} = A\, x_t \; + B\, u_t \tag{3.3}$$

$$y_t = C\, x_t. \tag{3.4}$$

The aggregate state, $x_t \in \mathbb{R}^{N^{\mathrm{x}}}$, is a set of discrete state bins constructed from a normalized temperature dead-band. The entries in x_t correspond to the portion of TCLs in each bin, and each TCL maps to a bin based on its current air temperature and on/off mode. The state transition matrix, $A \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{x}}}$, is a transposed Markov transition matrix describing the probability of transitioning between state bins during a time-step. Elements in the input $u_t \in \mathbb{R}^{N^{\mathrm{x}}/2}$ correspond to the probability mass that should be switched within each temperature interval, and $B \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{x}}/2}$ shifts the probability mass accordingly. The inputs are broadcast to the TCLs as switching probabilities, and TCLs switch their on/off mode with the probability corresponding to their current bin. The output $y_t \in \mathbb{R}$ is the total, or aggregate, power demand of the TCL population. It is formed using $C \in \mathbb{R}^{1 \times N^{\mathrm{x}}}$, which sums the portion of TCLs that are on, and then scales this by $\overline{P}^{\mathrm{on}}$, a historical average power draw of TCLs that are on. This model is observable [17], but it is over-defined, as x_t must sum to one. Given this and our input definition, there is one uncontrollable state. The reduced-order model presented in the following section eliminates this one uncontrollable state.

Note that [39] develops an alternative aggregate model that models the effect of the TCLs' internal mass temperature on the aggregate dynamics. Since we assume a stationary outdoor temperature, stationary disturbance distributions, and on/off control, we do not observe significant changes in the thermal mass temperature, and so we do not use this model.

3.2.3 Reduced-Order Aggregate Model

This section develops the reduced-order aggregate model from the linear, time-invariant aggregate model. Removing the single uncontrollable state results in a controllable reduced-order model that retains the observability of the original system. Eliminating a constant-valued state in the system's modal representation, which corresponds to the steady-state value of the aggregate state, preserves all dynamics of the original system.

Our approach relies on several facts about the aggregate model's modal representation: i) it contains an eigenvalue $\lambda_1 = 1$ and the subsystem corresponding to λ_1 is decoupled from the remaining system, ii) λ_1 is always the uncontrollable mode within the aggregate model in Section 3.2.2, iii) the component of the modal state corresponding to λ_1 has no dynamics and is actually a constant scalar equal to 1, and iv) the output associated with λ_1 is y_{ss}. These four points result in the following structure for the modal system

$$\begin{bmatrix} 1 \\ \widetilde{x}_{t+1} \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & \widetilde{A} \end{bmatrix}}^{A*} \overbrace{\begin{bmatrix} 1 \\ \widetilde{x}_t \end{bmatrix}}^{x_t^*} + \overbrace{\begin{bmatrix} 0 \\ \widetilde{B} \end{bmatrix}}^{B*} u_t \tag{3.5}$$

$$y_t = \underbrace{\begin{bmatrix} y_{\text{ss}} & \widetilde{C} \end{bmatrix}}_{C*} \begin{bmatrix} 1 \\ \widetilde{x}_t \end{bmatrix} \tag{3.6}$$

where stars and tildes denote modal and reduced-order quantities, respectively. Eliminating the constant modal state and defining $\widetilde{y}_t = y_t - y_{\text{ss}}$ forms the reduced-order system

$$\widetilde{x}_{t+1} = \widetilde{A}\,\widetilde{x}_t + \widetilde{B}u_t \tag{3.7}$$

$$\widetilde{y}_t = \widetilde{C}\,\widetilde{x}_t. \tag{3.8}$$

Point (i) follows from the proof in [76] that the state transition matrix of the three-state aggregate model contains an eigenvalue, $\lambda_1 = 1$, with an algebraic and geometric multiplicity of 1. This also holds for the time-invariant aggregate model, and the resulting Jordan block for λ_1 is decoupled from the remainder of the system. The right eigenvector for λ_1 is a unique steady-state value x_{ss} for the unforced, full-order aggregate model with corresponding output $y_{\text{ss}} = C\,x_{\text{ss}}$. Since the columns of A sum to 1, a vector of ones, denoted $\mathbf{1}$, is the left eigenvector of λ_1. Point (ii) follows from applying the PBH eigenvector test to λ_1 [77]. The test fails for λ_1 because the columns of B sum to 0, and so λ_1 is the uncontrollable mode of the full-order aggregate model. The uniqueness of λ_1 requires that the corresponding row of B^* is zeros.

Points (iii) and (iv) rely on the mapping from the original aggregate state to the modal aggregate state $x_t^* = T^{-1}x_t$. The rows of T^{-1} are the left eigenvectors, and the columns of T are the right eigenvectors. Placing the left and right eigenvectors of λ_1 as the first row and first column of T^{-1} and T respectively provides the structure of A^* and B^* in (3.5).

It also ensures the first modal state, noted as $x_t^{*,1}$, is always 1. Recall from Section 3.2.2 that $\mathbf{1}^T\,x_t = 1$. Given the structure of T^{-1}, then $x_t^{*,1} = \mathbf{1}^T x_t$, and (iii) follows for all valid x_t. To show (iv) note that the first column of T is x_{ss}, and the first element of $C^* = CT$ is $Cx_{\text{ss}} = y_{\text{ss}}$. Since $x_t^{*,1} = 1$, eliminating $x_t^{*,1}$ from the model only requires that the output of the reduced-order system is redefined as $\widetilde{y}_t$.

The approach above has similarities to that in [76] to achieve an asymptotically stable system. Both methods eliminate an eigenvalue $\lambda_1 = 1$ that is guaranteed to exist, and both methods redefine the output of the reduced-order model as $\widetilde{y}_t$. However, there are also differences between the methods. Whereas the method detailed here relies on points (i)-(iv), [76] uses projections into subspaces, does not establish decoupling of the subspaces, and does not note that the eliminated modal state is a constant. Also, the method detailed here shows the reduced-order model is controllable whereas [76] does not include inputs within their reduced-order model. The following section incorporates the full-order and reduced-order aggregate models into an MPC algorithm and a linear controller, respectively.

3.3 Control Algorithms

The controllers and the selection of inputs at the TCLs account for input delays by i) including the delay statistics, which are assumed to be known, within the control algorithms, ii) generating a sequence of inputs at each time-step rather than an individual input, and iii) allowing the TCLs to select inputs based on the realized delays, which are assumed to be known by the TCLs based on the capabilities of digital communication networks. The controllers utilize two external values – the desired demand level y_t^{des} and the aggregate state measurement x_t – to generate an input sequence. The input sequence contains inputs that are designed for the current time-step and some set of future time-steps. Each TCL uses selection logic to first find the most recently generated input sequence that has arrived, then it selects the input vector from this sequence that applies to a given time-step. IID input delays cause different TCLs to use different input vectors. The TCL then chooses the element of the input vector that corresponds to its current state, or it disregards the input when necessary to maintain the temperature within its normal operating range.

The linear controller generates the input sequence using a simple control law, calculated offline, that consists of matrix multiplication. Alternatively, the MPC controller solves a quadratic program online to generate an input sequence. Both controllers use delay statistics to compute the probabilities that previously generated inputs are implemented by TCLs, which reduces the effects of input delays.

Within this section, we occasionally use the time indexing notation $u_{k|t}$ where k indicates the time-step that the input applies to and t indicates the time-step during which the input was calculated. For example, an input $u_{t+4|t}$ is generated at time-step t, and it applies four time-steps after t. Using this notation, the input sequence

$u_t^{\mathrm{seq}} \in \mathbb{R}^{N^{\mathrm{x}}/2 \times N^{\mathrm{u}}}$ generated at each time-step t is a set of N^{u} input vectors

$$u_t^{\mathrm{seq}} = \begin{bmatrix} u_{t|t}^T & u_{t+1|t}^T & \cdots & u_{t+N^{\mathrm{u}}-1|t}^T \end{bmatrix}^T . \tag{3.9}$$

The number of time-steps within the input sequence is set so that the probability of a TCL having no valid input is $1 - p^{\max}$ where we choose $p^{\max}$. Section 3.3.1.1 details the process of setting N^{u} from $p^{\max}$.

Section 3.3.1 describes the MPC algorithm originally developed in [51] and presented in the previous chapter. Section 3.3.2 develops the linear controller.

3.3.1 MPC Algorithm

The MPC algorithm is a finite-horizon, quadratic program with equality and inequality constraints. The objective function penalizes desired aggregate power errors and input effort. Equality constraints embed the full-order aggregate model within the optimization problem, and inequality constraints impose the physical limitations on the feasible inputs and states. We define the N^{u} time-step horizon considered within the calculation at time t as $\mathcal{K}_t^{\mathrm{MPC}} = \{t, \ldots, t + N^{\mathrm{u}} - 1\}$. The aggregate state measurement x_t initializes the dynamics. Assume we have no knowledge of y_k^{des} over the horizon, so the desired aggregate power trajectory is assumed to be constant and equal to the current value.

The MPC controller's formulation at time t is

$$\min_u \sum_{k=t}^{t+N^{\mathrm{mpc}}-1} \left[c^y \left(y_k^{\mathrm{err}}\right)^2 + \sum_{j=k}^{k-N^{\mathrm{mpc}}+1} c^u (u_{k|j}^T \; u_{k|j}) \right] \tag{3.10}$$

$$\text{s.t.} \quad x_{k+1} = A\, x_k + B\, \hat{u}_k \tag{3.11}$$

$$\hat{u}_k = \mathcal{U}_k \mathcal{P} \tag{3.12}$$

$$y_k^{\mathrm{err}} = y_k^{P,\mathrm{ref}} - C^{\mathrm{P}} x_k \tag{3.13}$$

$$u_{k|j}^i \leqslant x_k^i \qquad i \in \{1, \ldots, N^{\mathrm{x}}/2\} \tag{3.14}$$

$$-u_{k|j}^i \leqslant x_k^{N^{\mathrm{x}}+1-i} \qquad i \in \{1, \ldots, N^{\mathrm{x}}/2\} \tag{3.15}$$

$$0 \leqslant x_k \leqslant 1 \tag{3.16}$$

with $k \in \mathcal{K}_t^{\mathrm{MPC}}$ and $j = k, \ldots, k - N^{\mathrm{u}} + 1$. The objective function (3.10) minimizes the total cost of output deviations (y_k^{err}, which is defined in (3.13)) and input effort, where c^y and c^u are cost coefficients. The state update (3.11) corresponds to the aggregate model of (3.3) while excluding the noise term, and it uses the estimated input $\widehat{u}_k$ from (3.12), which is calculated as a linear combination of the input matrix $\mathcal{U}_k \in \mathbb{R}^{N^{\mathrm{x}}/2 \times N^{\mathrm{u}}}$ and the weighting vector $\mathcal{P} \in \mathbb{R}^{N^{\mathrm{u}} \times 1}$, where $\mathcal{U}_k$ and $\mathcal{P}$ are detailed below. The input constraints (3.14) and (3.15) limit each input element based on the fraction of TCLs available to be turned on or off. Finally, (3.16) imposes physical limitations on the aggregate state. We implement the algorithm using [67].

3.3.1.1 Constructing Input Estimates

This section explains the construction of $\mathcal{U}_k$ and $\mathcal{P}$. To construct $\mathcal{U}_k$, note that inputs corresponding to a given time-step k appears within N^{u} MPC calculations. After each of these MPC calculations, an input sequence u_t^{seq} containing an input corresponding to time-step k is sent to the TCLs. The columns of the matrix $\mathcal{U}_k = \begin{bmatrix} u_{k|k} & \dots & u_{k|k-N^{\mathrm{u}}+1} \end{bmatrix}$ are the N^{u} separate input vectors that could apply to time-step k where the inputs become "older", i.e., they were generated at earlier calculations, as we go from left to right in the matrix.

The TCLs use the leftmost column of $\mathcal{U}_k$ that has arrived. The probability of using a column depends on two necessary events: 1) the column has arrived, and 2) the columns left of it within $\mathcal{U}_k$ must not have arrived. Denote the ith element of $\mathcal{P}$ as p^i, the corresponding column of $\mathcal{U}_k$ as u^i, and the probability of first and second necessary events as $p^{1,i}$ and $p^{2,i}$. Using the assumption of IID delays, $p^i = p^{1,i}p^{2,i}$. Define the delay τ^i associated with column u^i. The input u^i arrives by k if its delay is less than i

$$p^{1,i} = p(\tau^i < i) \qquad i = 1, \dots, N^{\mathrm{u}}. \tag{3.17}$$

Using IID delays, the probability that all columns left of column i have not arrived by time-step k is

$$p^{2,i} = \prod_{n=1}^{i-1} p(\tau^n \geqslant n) \qquad i = 1, \dots, N^{\mathrm{u}}. \tag{3.18}$$

The first column is used if its delay is less than one. Use of the second column requires that its delay is less than two and the first column's delay is at least one, and so on.

3.3.2 Linear Controller

This section reformulates the MPC algorithm into a linear controller that accounts for input delays through state-space augmentation of the aggregate model. We include an integrator for disturbance rejection, use reference feedforward to achieve tracking, and use an infinite-horizon output-regulating LQR for pole placement. To decouple reference tracking from disturbance rejection, the integrator dynamics are not included within the feedforward gain, which only includes the augmented aggregate model dynamics. Ref. [78] also uses LQR methods within demand response, however, they use a finite-horizon, output-tracking, LQR controller for commercial air conditioning systems. The following subsection develops the augmented system, and Section 3.3.2.2 develops the linear feedback law.

3.3.2.1 Augmenting the Aggregate Model

The augmented state vector, $\overline{x}_k$, includes the original state x_k and previously transmitted inputs that TCLs could still implement

$$\overline{x}_k = \begin{bmatrix} x_k^T & \overline{u}_{k|k}^T & \overline{u}_{k+1|k}^T & \cdots & \overline{u}_{k+N^{\mathrm{u}}-2|k}^T \end{bmatrix}^T . \tag{3.19}$$

The $\overline{u}_{k+a|k}$ values with $a = 0, \dots, N^{\mathrm{u}} - 2$ are constructed

$$\overline{u}_{k+a|k} = \begin{bmatrix} u_{k+a|k-1}^T & \cdots & u_{k+a|k-N^{\mathrm{u}}+1+a}^T \end{bmatrix}^T . \tag{3.20}$$

When updating $\overline{x}_k$ to $\overline{x}_{k+1}$, the $\overline{u}_{a|b}$ values become $\overline{u}_{a+1|b+1}$.

The block matrix form of the augmented system is then

$$\overline{x}_{k+1} = \overbrace{\begin{bmatrix} A & A_{\mathrm{B}} & 0 \\ 0 & 0 & A_{\mathrm{u}} \end{bmatrix}}^{\overline{A}} \overline{x}_k + \overbrace{\begin{bmatrix} p_1 B & 0 \\ 0 & B_{\mathrm{u}} \end{bmatrix}}^{\overline{B}} u_k^{\mathrm{seq}} \tag{3.21}$$

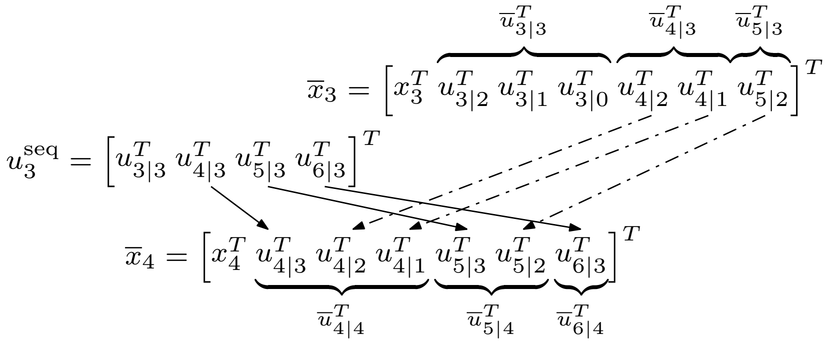
$$y_k = \underbrace{\begin{bmatrix} C & 0 & 0 \end{bmatrix}}_{\overline{C}} \overline{x}_k \tag{3.22}$$

where $\bar{\cdot}$ indicates augmented quantities, u_k^{seq} is defined in (3.9), and p_1 is the first element of $\mathcal{P}$. We explain A_{B}, A_{u}, and B_{u} below. The first block row of the block matrices $\overline{A}$ and $\overline{B}$ updates x_k. The second block row manipulates the inputs within $\overline{x}_k$ as time progresses. The construction of $\overline{x}_k$ multiplies the first column of $\overline{A}$ and $\overline{C}$ with x_k, the second column multiplies with $\overline{u}_{k|k}$, and the last column multiplies with the remaining inputs within $\overline{x}_k$.

Finally, we explain the A_{B}, A_{u}, and B_{u} matrices. To construct A_B, we reorganize $B\hat{u}_k = B\mathcal{U}_k\mathcal{P}$ into

$$B\mathcal{U}_k\mathcal{P} = p_1 B u_{k|k} + \cdots + p_{N^{\mathrm{u}}} B u_{k|k-N^{\mathrm{u}}+1} \tag{3.23}$$

$$= p_1 B u_{k|k} + A_{\mathrm{B}} \overline{u}_{k|k} \tag{3.24}$$



$\overline{u}_{3|3}^T$
$\overline{u}_{4|3}^T$
$\overline{u}_{5|3}^T$
$\overline{x}_3 = \begin{bmatrix} x_3^T\ u_{3|2}^T\ u_{3|1}^T\ u_{3|0}^T\ u_{4|2}^T\ u_{4|1}^T\ u_{5|2}^T \end{bmatrix}^T$
$u_3^{\text{seq}} = \begin{bmatrix} u_{3|3}^T\ u_{4|3}^T\ u_{5|3}^T\ u_{6|3}^T \end{bmatrix}^T$
$\overline{x}_4 = \begin{bmatrix} x_4^T\ u_{4|3}^T\ u_{4|2}^T\ u_{4|1}^T\ u_{5|3}^T\ u_{5|2}^T\ u_{6|3}^T \end{bmatrix}^T$
$\overline{u}_{4|4}^T$
$\overline{u}_{5|4}^T$
$\overline{u}_{6|4}^T$

Figure 3.2: Diagram portraying the effects of A_u and B_u when advancing the augmented state from $k = 3$ to $k = 4$ with $N^{\text{u}} = 4$. The dash-dotted lines correspond to manipulations by A_u, which advances inputs for future time-steps within $\overline{x}_3$. The solid lines corresponds the action of B_u, which places inputs from u_3^{seq} into $\overline{x}_4$.

where $A_{\text{B}} = \begin{bmatrix} p_2 B & \cdots & p_{N^{\text{u}}} B \end{bmatrix}$. The components of A_{u} and B_{u} are identity and zero matrices of various sizes that appear without a simple pattern. Rather than constructing A_{u} and B_{u} explicitly, Fig. 3.2 depicts their effect on the augmented state.

The augmented system above uses the matrices from (3.3)-(3.4) and adds controllable and observable modes at 0 into the system. The model reduction method in Section 3.2.3 can still eliminate the uncontrollable mode within the extended system. We denote the reduced-order, augmented matrices as $\tilde{\overline{A}}$, $\tilde{\overline{B}}$, and $\tilde{\overline{C}}$ respectively. The reduced-order, augmented state is $\tilde{\overline{x}}_k$, and the corresponding output is still $\tilde{y}_k$. These are used below.

3.3.2.2 Control Law Development

We define the linear control law with constant gain matrices K_∞^{x}, K_∞^{w}, and K_∞^{y} as

$$u_t^{\text{seq}} = -K_\infty^{\text{x}}\, \overline{x}_t - K_\infty^{\text{w}}\, w_t + K_\infty^{\text{y}}\, y_t^{\text{des}} \tag{3.25}$$

where w_t is an the integrator state that captures the historical tracking error. An output-regulating LQR formulation with $\tilde{y}_k^{\text{des}} = 0$ generates the feedback terms K_∞^{x} and K_∞^{w}

$$\min_u \sum_{k=t}^{\infty} \begin{bmatrix} \tilde{\overline{x}}_k \\ w_k \end{bmatrix}^T \begin{bmatrix} (\tilde{\overline{C}})^T q^{\text{y}} \tilde{\overline{C}} & 0 \\ 0 & q^{\text{w}} \end{bmatrix} \begin{bmatrix} \tilde{\overline{x}}_k \\ w_k \end{bmatrix} + (u_k^{\text{seq}})^T\, R\, u_k^{\text{seq}} \tag{3.26}$$

$$\text{s.t.} \begin{bmatrix} \tilde{\overline{x}}_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{\overline{A}} & 0 \\ \tilde{\overline{C}} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\overline{x}}_k \\ w_k \end{bmatrix} + \begin{bmatrix} \tilde{\overline{B}} \\ 0 \end{bmatrix} u_k^{\text{seq}} \tag{3.27}$$

where the scalars q^{y} and q^{w} penalize $\widetilde{y}_k$ and w_k respectively. The input penalty is $R = q^{\mathrm{u}} I$ where q^{u} is a scalar. This formulation results in a feedback gain for the reduced-order augmented state, $\widetilde{K}^{\mathrm{x}}_\infty$, that we convert using $K^{\mathrm{x}}_\infty = [0\ \widetilde{K}^{\mathrm{x}}_\infty]\, T$, where T denotes the mapping from modal states.

Output tracking and regulating formulations produce identical feedback gains [79]. However, using the tracking formulation above would result in a feedforward gain incorporating the integrator. We use an alternative feedforward gain that excludes the integrator while attempting to achieve steady-state tracking

$$K^{\mathrm{y}}_\infty = \left(\widetilde{\overline{C}}\{zI - \widetilde{\overline{A}} + \widetilde{\overline{B}}\widetilde{K}^{\mathrm{x}}_\infty\}^{-1}\widetilde{\overline{B}}\right)^{-\dagger} \tag{3.28}$$

where $-\dagger$ is a pseudo-inverse that is needed because there are more inputs than outputs, and we set $z = 1$.

3.4 Case Studies

This section evaluates the MPC controller and linear controller, denoted as LIN, while tracking a desired aggregate demand signal under a variety of scenarios. We present RMS tracking errors and statistics on each controllers' computation time to quantify and compare the scenarios. Section 3.4.1 defines the scenarios used to simulate the system and presents the performance metrics. Section 3.4.2 summarizes the results of the scenarios.

3.4.1 Scenario Definitions

We simulate 12 scenarios using combinations of the two controllers, two reference signals, and three delay scenarios. Fifty instances of each scenario use different realizations of the random quantities. The simulation time is one hour, with 1800 time-steps, and we use the following parameter settings: $N^{\mathrm{TCL}} = 10,000$, $N^{\mathrm{x}} = 30$, $p^{\max} = 0.999$, $c^{\mathrm{y}} = 1$, $c^{\mathrm{u}} = 1$, $q^{\mathrm{y}} = 0.1$, $q^{\mathrm{u}} = 1$, $q^{\mathrm{w}} = 0.01$. In all scenarios, a population of N^{TCL} hybrid models, detailed in Section 3.2.1, are simulated to represent the plant whereas the controllers rely on the aggregate models detailed in Sections 3.2.2 and 3.2.3.

The two reference signals correspond to historical dynamic and traditional PJM frequency regulation signals, denoted as "Reg-D" and "Reg-A" references respectively. Data published by PJM [68] from May 4, 2014 are interpolated to two second time-steps, and each signal is scaled so that the maximum demand change request

Table 3.2: Computation Times for Generating Inputs

Controller	Mean Delay (s)	Mean Time (s)	Max Time (s)
MPC	0	0.187	0.978
	10	0.589	2.185
	20	1.123	3.800
LIN	0	0.001	0.031
	10	0.003	0.055
	20	0.014	0.132

corresponds to ±20% of the average steady-state aggregate TCL demand.

The scenarios include three delay distributions – a delay-free scenario referred to as delay case 0 and two scenarios with delays. In the scenarios with delays, IID delays are sampled by i) sampling values from a log-normal distribution with mean μ and variance σ^2, then ii) rounding down the sampled values. Delay cases 1 and 2 set μ to 10 and 20 seconds respectively, and the variance is 0.25 for both. Note that the choice of distribution and their parameters are examples, not requirements of the algorithm.

We quantify each simulation using the normalized RMS tracking error (RMSE), the average time to compute an input, and the maximum time to compute an input. The tracking error for each time-step is $y_t - y_t^{\text{des}}$, and the standard RMSE is then expressed as a percentage of the average steady-state aggregate TCL demand. The RMSE value associated with each scenario is the average RMSE across all of the scenario's instances. We compute an average computation time for each controller to generate an input during each delay case, meaning we average across instances without differentiating the reference signal used. Finally, the single maximum computation time for generating an input is taken from the set of instances for a controller and delay case.

3.4.2 Results

Table 3.2 summarizes computation times for the controllers, Fig. 3.3 summarizes the RMSE of the scenarios, Fig. 3.4 provides sample time series. The simulations were carried out on a server using Matlab. Note that the controllers were roughly tuned to values that provide good performance across all scenarios. Improved performance for both controllers may be achievable by additional tuning, but we do not believe additional tuning would make LIN outperform MPC.

From Table 3.2, we see that LIN, the linear controller, achieves a significant reduction in computation time needed to generate an input. The average time and maximum time for the linear controller is roughly 100 times faster than the MPC controller. Also, the maximum time needed to generate an input is 0.132 seconds in

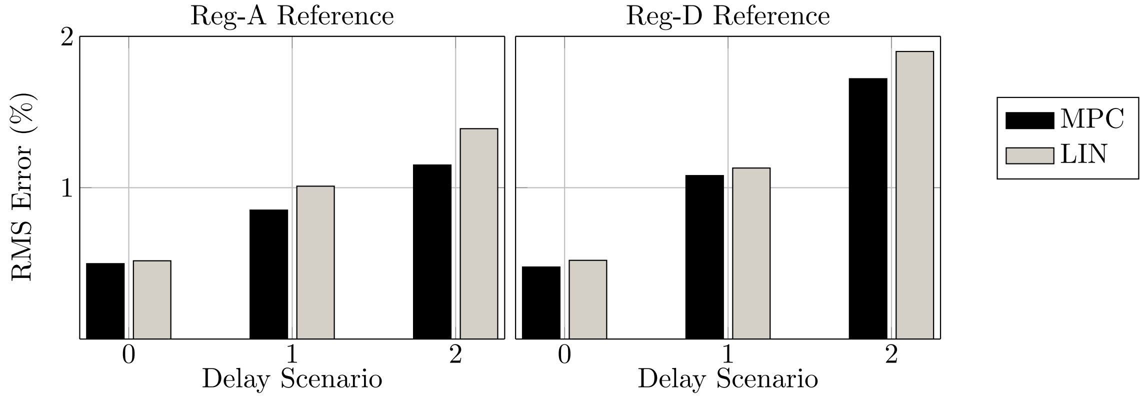


Figure 3.3: RMS errors of the LIN and MPC controllers in delay scenarios 0, 1, and 2 when attempting to track the Reg-A and Reg-D reference signals.

the case with the largest delay. This compares with the MPC controller's maximum time of 3.8 seconds in the same scenario, which exceeds the 2 second time-step and means the MPC approach will not always be able to compute an input within the given time-step. While the computation times are dependent on the machine running the simulations, LIN clearly achieves faster computation times. This is especially important when considering the more realistic case where delays and communication network limitations affect the transmission of state and output measurements, which will require use of an estimator. By reducing the computation time of the controller, we allow more time to compute the state estimates within the given time-step duration.

This reduction in computation time comes at the cost of increased RMSE, as Fig. 3.3 shows. Without delays, the MPC and LIN controllers perform roughly equivalently. As the delays increase, the MPC controller is able to achieve better tracking with slightly reduced RMSE. However, the differences in RMSE are not large, and it may be worth sacrificing some tracking accuracy for simplicity (i.e., a closed-form control law) and reduced computational requirements. In addition, simulations not presented here show that LIN is better able to compensate for errors in the probability vector $\mathcal{P}$ than the MPC approach, likely because of the integrator.

The number of state bins and amplitude of the reference signal also influence the results; when using a more extreme reference signal, input inequalities become more important, and LIN (which does not explicitly include these constraints) performs worse. To explore this scenario, we increase the amplitude of the reference signal to 80% of the mean steady-state TCL demand. When tracking the "Reg-D" Reference without delays, LIN has an RMSE of 0.74% and hundreds of input constraint viola-

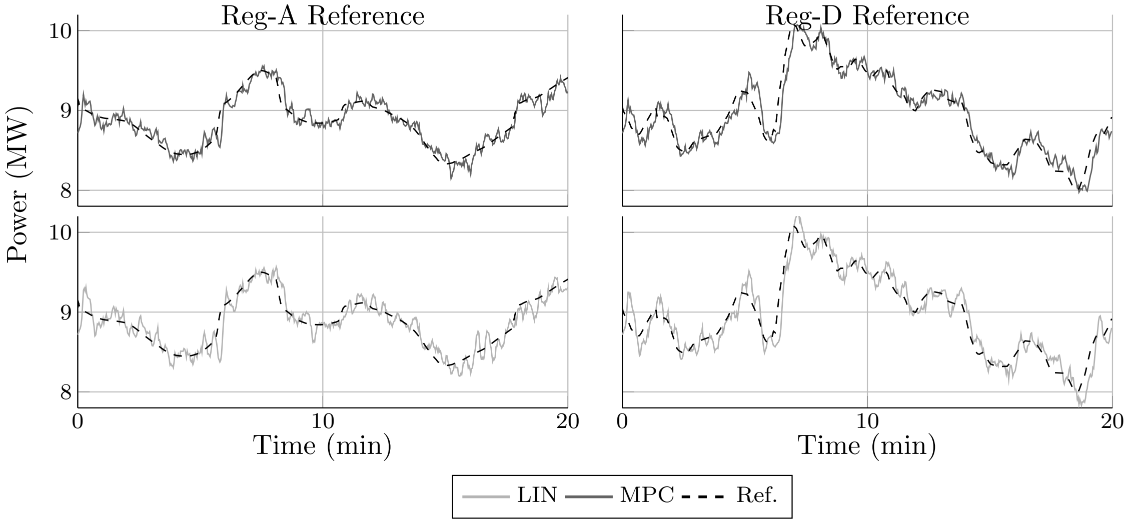


Figure 3.4: Time series showing the tracking of the Reg-A and Reg-D references under delay scenario 2.

tions. Alternatively, MPC achieves an RMSE of 0.59% and no constraint violations.

3.5 Chapter III Conclusions

In this chapter, we developed a linear controller that mitigates the effect of input delays in residential demand response, and we compared it to a previously developed MPC controller through simulations that manipulate the aggregate demand of thousands of air conditioners to track real frequency regulation signals. Both methods counteract input delays by generating an open-loop input sequence at each time-step and by incorporating knowledge about the input delay statistics. The linear controller reduces computation time significantly while losing some tracking performance in the scenarios investigated. This may be a reasonable trade-off since the MPC computation time is sometimes longer than the time-step duration and, in practice, the algorithm will require a state estimator, which will also take time to run. Additionally, the integrator makes LIN more robust to errors in the delay statistics. Future work will design the linear controller in conjunction with an estimator that addresses communication issues in state and output measurement transmission, as considered in the MPC controller [51].

CHAPTER IV

Benchmarking of Aggregate Residential Load Models Used for Demand Response [1]

The power consumption of large numbers of thermostatically controlled loads (TCLs), such as residential air conditioners (ACs), can be coordinated to help the electric power grid balance supply and demand [37,38]. In addition to participating in traditional demand response programs, loads can be controlled to provide ancillary services, such as frequency regulation, by decreasing/increasing consumption with respect to their baseline. Much of the work on load control for ancillary services assumes a load aggregator receives a signal from the system operator and controls an aggregation of loads to match that signal. A significant body of recent work has sought to develop aggregate residential load models, e.g., [17, 39, 81, 82], which the aggregator could use in state estimation and control algorithms. These models capture the dynamics of the total power consumption of the demand-responsive loads. Use of dynamic models generally improves control performance as compared to model-free control approaches. However, modeling aggregations of loads by representing each load individually leads to large and often complex models. For example, TCLs are best modeled as hybrid systems since they cycle on/off within a temperature hysteresis band. A good aggregate model balances simplicity and performance.

Despite significant recent efforts to develop aggregate load models, we only have a partial understanding of which models work best under which conditions. Each model was developed under a different set of assumptions (e.g., homogeneous vs. heterogeneous loads, static vs. dynamic ambient conditions such as outdoor temperature) and assuming a specific type of control (e.g., on/off switching, temperature setpoint control). Models are generally validated in simulation studies, and, often, the same unrealistic assumptions used to build the model are used to validate it.

[1] The work presented in this chapter was originally published in [80].

In this chapter, we seek a better understanding of the advantages and disadvantages of three aggregate load models that represent the dynamics of a heterogeneous AC population. Two of the models use Markov chains [17,39] and one uses a transfer function [81, 82]. We identify each model and then use it to predict the aggregate power consumption of 10,000 air conditioners over a 24 hour period with a time-varying outdoor temperature. Prediction accuracy is computed against a realistic simulation model in which individual air conditioners are represented with hybrid system models including three states: indoor air temperature, indoor mass temperature, and on/off mode. We assume each AC has a time-varying cooling capacity, a time-varying coefficient of performance (COP), and a time-varying power draw (when the AC is on) that all depend on the outdoor temperature, making our plant more accurate than that typically used in the aggregate load modeling literature. In this preliminary work, we assume that the loads are not coordinated by a load aggregator; investigating model performance under aggregator control is a subject for future work.

Beyond the simulation-based benchmarking of three aggregate load models, this chapter also includes several methodological contributions required to enable a fair comparison in a realistic setting with a time-varying outdoor air temperature. In particular, we extend the Markov models [17, 39] to update the Markov transition matrix parameters as a function of the temperature trend, and we discuss other ways in which the model could be improved. We also determine a transfer function structure that may lead to improved model accuracy as compared to the model in [81, 82] and provide suggestions on better ways to identify its parameters.

The rest of this paper is organized as follows: Section 4.1 details the individual TCL model used in the plant and Section 4.2 describes each aggregate load model and our extensions. Section 4.3 describes the simulation setting and results. Section 4.4 provides the conclusions of the chapter.

4.1 Individual TCL Model

We use the hybrid model from the residential module of GridLAB-D [83] to model an individual AC within the plant. It is similar to the model described in Section 2.2.1.1, and we describe it below to introduce notation and to detail the differences. The hybrid model contains continuous states corresponding to the internal air temperature and mass temperature $\theta_t^a, \theta_t^m \in \mathbb{R}$ and a discrete state corresponding to the on/off mode $m_t \in \{0, 1\}$, where the AC is drawing power if $m_t = 1$. Table 4.1

Table 4.1: Air Conditioner Model Parameters for Chapter IV

Parameter	Description	Value
θ^{set}	Temperature Setpoint [°C]	$\mathcal{U}\{20, 24\}$
θ^{db}	Temperature Deadband [°C]	$\mathcal{U}\{1, 2\}$
θ^{o}_t	Outdoor Temperature [°C]	Varies from 25.2 to 35.0
U^{m}	Mass Conductance $[\frac{\text{kW}}{^\circ\text{C}}]$	$\mathcal{U}\{4.4, 5.4\}$
U^{a}	Air Conductance $[\frac{\text{kW}}{^\circ\text{C}}]$	$\mathcal{U}\{0.25, 0.3\}$
Λ^{m}	Mass Heat Capacitance $[\frac{\text{kWh}}{^\circ\text{C}}]$	$\mathcal{U}\{2.0, 2.5\}$
Λ^{a}	Air Heat Capacitance $[\frac{\text{kWh}}{^\circ\text{C}}]$	$\mathcal{U}\{0.5, 0.6\}$
Q^{AC}_t	Cooling Capacity [kW]	$\frac{Q^{\text{rate}}\ (1.32-0.01\ \theta^{\text{o}}_t)}{1+f^{\text{latent}}}$
Q^{rate}	Rated Cooling Capacity [kW]	$\mathcal{U}\{11.1, 13.5\}$
f^{latent}	Fraction of Latent Cooling [-]	0.35
P^{AC}_t	Power Draw [kW]	Q^{AC}_t/η_t
η_t	Coefficient of Performance [-]	$\eta^{\text{std}}(0.33 + 0.02\ \theta^{\text{o}}_t)^{-1}$
η^{std}	COP at Standard Conditions [-]	3.5

summarizes other parameters used within the model where $\mathcal{U}\{a, b\}$ indicates a uniform distribution between a and b. Table 4.1 includes time-varying values for the cooling capacity Q^{AC}_t, COP η_t, and power draw P^{AC}_t, which are all functions of the time-varying outdoor temperature θ^{o}_t. Note that we have neglected several aspects that are included within the GridLAB-D model: minimum cycling times, the power draw and heat injection from the circulation fan, and heating from solar irradiance and internal heat gains; including these are a subject for future work.

The update equations for the hybrid model are

$$\theta_{t+1} = A\ \theta_t + B\ (Q^{\text{AC}}_t m_t) + E\ \theta^{\text{o}}_t, \tag{4.1a}$$

$$m_{t+1} = \begin{cases} 0 & \text{if } \theta^a_{t+1} < \theta^{\text{set}} - \theta^{\text{db}}/2 \\ 1 & \text{if } \theta^a_{t+1} > \theta^{\text{set}} - \theta^{\text{db}}/2 \\ m_t & \text{otherwise,} \end{cases} \tag{4.1b}$$

where (4.1a) updates the temperatures with $\theta_t = \begin{bmatrix} \theta^a_t & \theta^m_t \end{bmatrix}^{\text{T}}$, and (4.1b) updates the on/off mode if the air temperature reaches the edge of the allowable temperature range. Matrices A, B, and E are computed by first constructing continuous-time matrices using the thermal parameters U^{m}, U^{a}, Λ^{m}, and Λ^{a}, and then discretizing the continuous-time matrices using the time-step $\Delta t = 2$ seconds. To simulate a set of n^{AC} ACs, we parameterize the set of ACs by independently selecting the relevant parameters from the distributions in Table 4.1, where $\mathcal{U}\{\cdot, \cdot\}$ refers to a uniform distribution. We update each AC's temperatures and on/off mode by applying (4.1a) and (4.1b) with the AC's parameters.

4.2 Aggregate TCL Models

If n^{AC} is large (e.g., on the order of thousands), then incorporating the n^{AC} hybrid models into control and estimation algorithms can be computationally prohibitive. As a result, these algorithms often employ aggregate models, which model the behavior of the n^{AC} ACs using a single, simpler model. Here, we compare three aggregate models. The first model, developed in [17] and referred to as the two-state Markov model, defines a set of discrete bins based on θ_t^a and m_t, constructs an aggregate state as the portion of air conditioners in each bin, and it uses a Markov transition matrix to update the aggregate state of the model. This is the same model detailed in Section 2.2.2, and we describe it here to reintroduce notation. The second model, developed in [39] and referred to as the three-state Markov model, is similar but defines a set of discrete state bins based on θ_t^a, θ_t^m, and m_t. The third model, developed in [81, 82], referred to as the transfer function model, maps changes in the outdoor temperature to changes in the steady state aggregate demand.

4.2.1 Two-State Markov Model

The two-state Markov model [17] uses an aggregate state $x_t \in \mathbb{R}^{2n^{\text{a}}}$, which is the portion of air conditioners in each of $2n^{\text{a}}$ discrete state bins. The discrete state bins are formed by dividing a normalized temperature deadband into n^{a} temperature intervals, and then creating two discrete states in each interval, one for air conditioners that are on and one for those that are off. An AC maps to a bin based on its air temperature and on/off mode. The two-state autonomous Markov model is

$$x_{t+1} = A_t \, x_t, \tag{4.2}$$

$$y_t = C_t \, x_t, \tag{4.3}$$

where $y_t \in \mathbb{R}^1$ is the aggregate demand, A_t is a transposed Markov transition matrix where the entries correspond to the probability of bin transitions within the time-step, and $C_t = n^{\text{AC}} \, \overline{P}_t^{\text{on}} \begin{bmatrix} 0 & \dots & 0 & 1 & \dots 1 \end{bmatrix}$ where the scalar $\overline{P}_t^{\text{on}}$ is the average power draw of air conditioners that are on. In [17] $A_t = A$, corresponding to a constant outdoor temperature. In [84], A_t is a function of the time-varying outdoor temperature.

Here, to compute A_t and $\overline{P}_t^{\text{on}}$, we first compute a set of time-invariant matrices $\mathcal{A}$

and average power draws $\mathcal{P}^{\text{on}}$ at different outdoor temperatures $\mathcal{T}^{\text{o}}$. Each element of $\mathcal{A}$ is computed by counting the bin transitions at the corresponding outdoor temperature from $\mathcal{T}^{\text{o}}$, and then normalizing the columns to sum to one. We then compute A_t by linearly interpolating the elements of the two time-invariant matrices corresponding to the temperature above and below θ_t^{o}. Each element of $\mathcal{P}^{\text{on}}$ is computed by calculating the average power draw of ACs that are on when θ_t^{o} is at the temperature from $\mathcal{T}^{\text{o}}$. We interpolate $\overline{P}_t^{\text{on}}$ in a similar manner.

We use three approaches to identify $\mathcal{A}$ and $\mathcal{P}^{\text{on}}$.

- MM2-C: We simulate the n^{AC} ACs for each integer temperature within $\mathcal{T}^{\text{o}}$, holding the temperature constant during each simulation. Each entry of $\mathcal{A}$ and $\mathcal{P}^{\text{on}}$ is computed with data from one simulation.

- MM2-V: We simulate the ACs with a time-varying outdoor temperature using historical temperature data. Each entry of $\mathcal{A}$ and $\mathcal{P}^{\text{on}}$ is computed with data corresponding to outdoor temperatures nearest to the entry's associated temperature.

- MM2-S: In addition to simulating the ACs with a time-varying outdoor temperature, we create two sets $\mathcal{A}$ and $\mathcal{P}^{\text{on}}$, one for when θ_t^{o} is increasing and one for when θ_t^{o} is decreasing. This is justified because the interaction between θ_t^{a} and θ_t^{m} is different for an increasing versus decreasing θ_t^{o}.

4.2.2 Three-State Markov Model

The three-state Markov model [39] creates discrete bins, similar to those of the two-state Markov model, but based on both the air and mass temperatures. The deadband is divided into n^{a} temperature intervals for the air temperature, as in the two-state Markov model, and the deadband is divided into n^{m} mass intervals for the mass temperature. The aggregate state is then $x_t \in \mathbb{R}^{2n^{\text{a}}n^{\text{m}}}$, which is the portion of ACs in each of the $2(n^{\text{a}}n^{\text{m}})$ discrete state bins. The three-state autonomous Markov model is (4.2) and (4.3). As with the two-state Markov model, we define three methods of constructing A_t and $\overline{P}_t^{\text{on}}$: MM3-C, MM3-V, and MM3-S. They each construct the sets $\mathcal{A}$ and $\mathcal{P}^{\text{on}}$ using the same methodology as with the two-state Markov model. Note that this model is constructed assuming that measurements of the thermal mass are available within each residence, which are not available in practice.

Also, note that the structure (i.e., the location of zero and non-zero entries) of each time-invariant matrix within $\mathcal{A}$ can be different for this model, depending on the outdoor temperature used to construct the model. As A_t is computed by linearly interpolating between different time-invariant matrices with different structures, the structure of A_t changes over time. Non-zero elements of x_t can then be set to zero due to the changing structure, reducing the probability mass within x_t to less than one. We heuristically ensure that each column in the time-invariant matrices sum to one, ensuring that the probability mass within x_t is not reduced.

4.2.3 Transfer Function Model

The transfer function model, developed in developed in [81,82], maps a change in the ambient temperature to a change in the steady state aggregate demand, and it has the form

$$y(t) = \Big(\sum_{i=1}^{n^{\mathrm{AC}}} P_t^{\mathrm{AC},i} \Big) \, \mathcal{L} \left\{ G(s) \right\}^{-1} (\theta_t^{\mathrm{o}} - \theta^{\mathrm{init}}) \tag{4.4}$$

$$G(s) = \frac{b_1 s + b_2}{s^2 + \zeta \omega_{\mathrm{n}} \, s + \omega_{\mathrm{n}}^2} \tag{4.5}$$

where $\mathcal{L}\left\{\cdot\right\}^{-1}$ is the inverse Laplace transform, θ^{init} is the initial outdoor temperature, s is the variable in the Laplace domain, ζ is the damping coefficient of an under-damped system, ω_{n} is the natural frequency of an under-damped system, b_1 and b_2 are coefficients, and $\sum_{i=1}^{n^{\mathrm{AC}}} P_t^{\mathrm{AC},i}$ is the total power draw if all ACs were on at time t. The coefficients and parameters in (4.5) are computed based on an assumed response to a step change in the outdoor temperature from θ^{init} to θ^{final}, where these temperatures must be assumed *a priori* to compute the model. The details about the calculation of (4.5) can be found in [81,82], but some general aspects of the calculations include the following: 1) the assumed, individual AC model is a simplified version of that in Section 4.1, which only contains one continuous state for the air temperature; 2) many of the parameters characterizing the individual AC model are assumed to be identical across the AC population; 3) estimates for the steady state portion of ACs that are on must be computed for both θ^{init} and θ^{final}; and 4) an estimate for the the period of the oscillations resulting from the step change is computed using the parameter distributions of the population, which are assumed to be known.

The AC models in Section 4.1 differ substantially from the assumptions within [82]. As a result, we define two approaches to generate models based on transfer functions.

- TF-O: We implement (4.4) and (4.5) according to [82]. Thermal parameters for the AC model with one continuous state are identified from historical data of

θ_t^{a}, m_t, and θ_t^{o} data for each AC. Parameters that are assumed to be identical across the AC population in [82] are taken as their average value.

- TF-ID: We use a data-driven approach to identify a transfer function structure as well as the transfer function parameters from historical θ_t^{o} and aggregate demand data. We identify the transfer function parameters using the *tfest*$(\cdot)$ function in Matlab, where we subtract the initial values of the aggregate demand data and θ_t^{o} data. We then select the transfer function structure that achieves the lowest error based on historical data. The resulting model has the following structure:

$$y(t) =\mathcal{L}\left\{\frac{b_3 s^2 + b_4 s + b_5}{s^2 + a_1\, s + a_2}\right\}^{-1} (\theta_t^{\mathrm{o}} - \theta^{\mathrm{init}}) + y^{\mathrm{init}} \tag{4.6}$$

  where y^{init} is the initial demand and the parameters to be identified are b_3, b_4, b_5, a_1, and a_2.

To implement each of the transfer functions within a discrete-time simulation, we convert each transfer function to a separate continuous-time, state-space model and then discretize each state-space model.

4.3 Simulation-Based Case Studies

In this section, we use a number of simulation-based case studies to investigate the prediction accuracy of the three aggregate models and their variations. Section 4.3.1 summarizes the simulation settings for the case studies, and Section 4.3.2 presents the results. In general, more accurate aggregate models lead to more accurate control and state estimation algorithms that could be used to provide frequency regulation with ACs throughout the day. As a result, we evaluate the models using the RMS error (RMSE) of the model's predicted aggregate demand versus the true aggregate demand over the day.

4.3.1 Parameterization

The case studies each use a plant of 10,000 simulated ACs, modeled and parameterized according to Section 4.1, over 24 hours using a time-varying outdoor temperature. The outdoor temperature, available through the Pecan Street Dataport [85], corresponds to that of Austin, TX on July 10, 2016, where we linearly interpolate the data from one-hour to two-second time-steps. Figure 4.1 depicts the

Table 4.2: Simulation Settings for Chapter IV

Parameter	Description	Value
Δt	Time-step duration [s]	2
n^{steps}	Number of time-steps [-]	43,200
n^{AC}	Number of air conditioners [-]	10,000
n^{a}	Number of air temperature bins [-]	20
n^{m}	Number of mass temperature bins [-]	20
$\mathcal{T}^{\text{o}}$	Set of temperatures used to compute the various Markov models	$\{24, 25, \ldots, 36\}$
θ^{init}	Initial outdoor temperature used to compute the transfer function [°C]	28.8
θ^{final}	Final outdoor temperature used when computing the transfer function [°C]	35.0

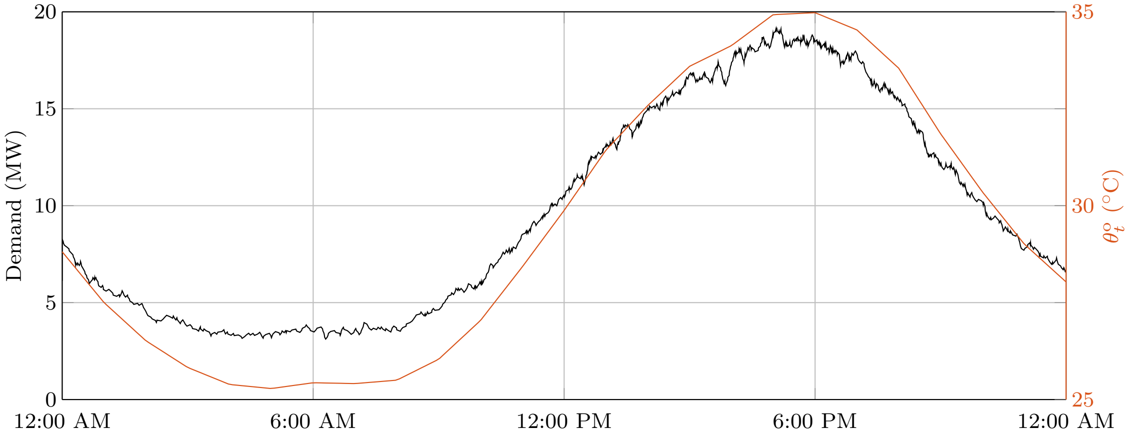


Figure 4.1: The aggregate demand of the simulated ACs and the outdoor temperature for the case study setup

aggregate demand of the simulated ACs, which is the ground-truth demand in our simulations, along with θ_t^{o}. Table 4.2 contains additional simulation parameters. We use the RMSE to evaluate the prediction accuracy of each aggregate model versus the aggregate demand of the simulated ACs.

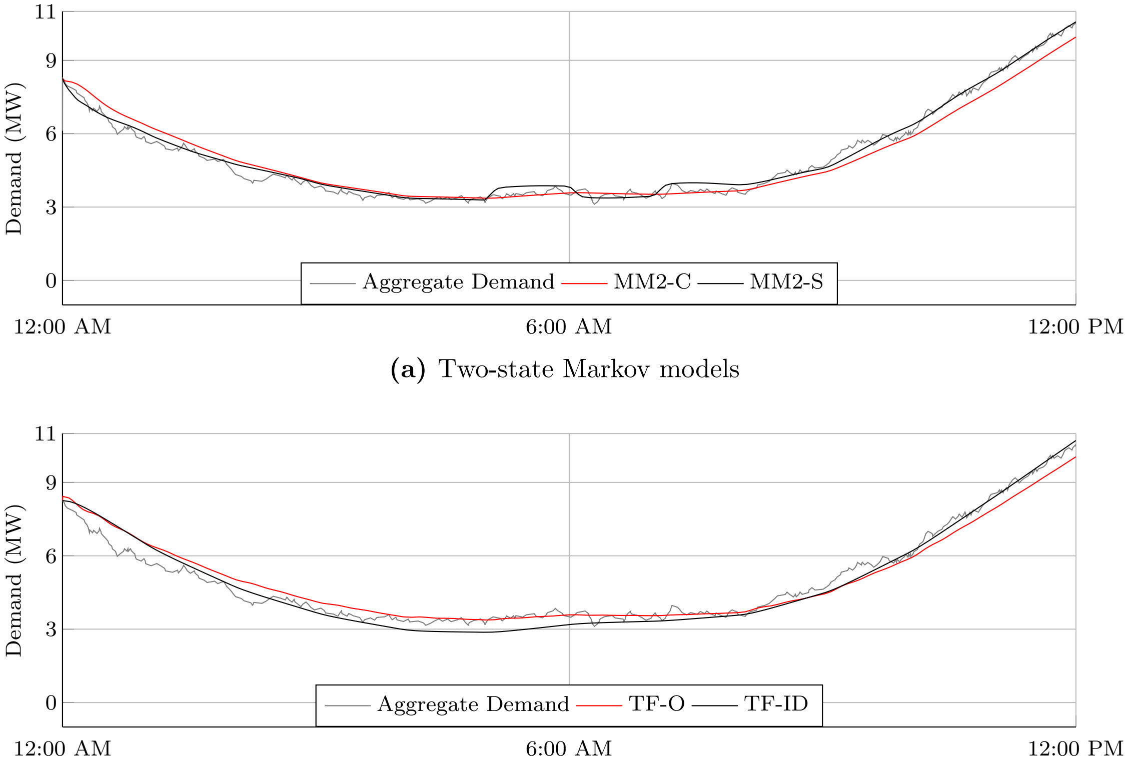
To compute the Markov models, we generate the necessary data by simulating the plant models using outdoor temperature data from July 1-9, 2016. We initialize the Markov model at the true aggregate state value. To generate the parameters for the transfer function models, we simulate the plant models using the outdoor temperature for July 9, 2016. For TF-O, the transfer function was computed using θ^{init} as the actual initial temperature of the simulation and θ^{final} as the maximum temperature of the simulation. The initial state of the discreteized TF-O model is set to a vector of zeros. For TF-ID, y^{init} is set to the initial aggregate demand, θ^{init} is set to the actual initial temperature, and the initial state of the discretized TF-ID model is set to zeros.

4.3.2 Results

Table 4.3 summarizes the aggregate model variations and their RMSE values. Figure 4.2 presents time series of the aggregate demand and the predictions of a number of aggregate models over a portion of the simulated day. Figure 4.2a presents time series for MM2-C and MM2-S where we exclude MM2-V for clarity, and Fig. 4.2b presents time series for TF-O and TF-ID. We do not present time series for the three-state Markov model predictions as they are similar to those of the two-state Markov model. Figure 4.3 presents errors of various aggregate model's predictions versus the aggregate demand over the entire simulated day. Figure 4.3a presents the errors of MM2-C and MM2-S, and Fig. 4.3b presents the prediction error for MM3-C and MM3-S, where we exclude MM2-V and MM3-V for clarity. Figure 4.3c presents the prediction error for TF-O and TF-ID.

As can be seen in Table 4.3, the increased modeling detail of the three-state aggregate model reduces the RMSE errors versus the two-state aggregate model across all variations. Accounting for the trend in θ_t^o in MM2-S and MM3-S reduces the RMSE versus the other Markov-based model varieties substantially. Furthermore, both MM2-S and MM3-S achieve similar RMSE, indicating that accounting for the θ_t^o reduces the gap in modeling accuracy between the two Markov-based aggregate models. The reduced RMSE when using MM2-S or MM3-S can be see in Fig. 4.2a (which shows MM2-S but is true for MM3-S as well). Whereas the prediction of MM2-C lags the aggregate demand, this is corrected in MM2-S by accounting for differences in the AC model behavior when θ_t^o is increasing versus decreasing. Also, note that from 4:00 AM to 8:00 AM, θ_t^o is relatively flat and switches between increasing and decreasing several times, and the resulting prediction of MM2-S undergoes several jumps. This results in significant prediction error in MM2-S and MM3-S, as can be seen in Fig. 4.3a and Fig. 4.3b. Further differentiating between an increasing, decreasing, or flat trend in θ_t^o may improve model performance. The time series for MM3-C and MM3-S, which are not included, show similar trends.

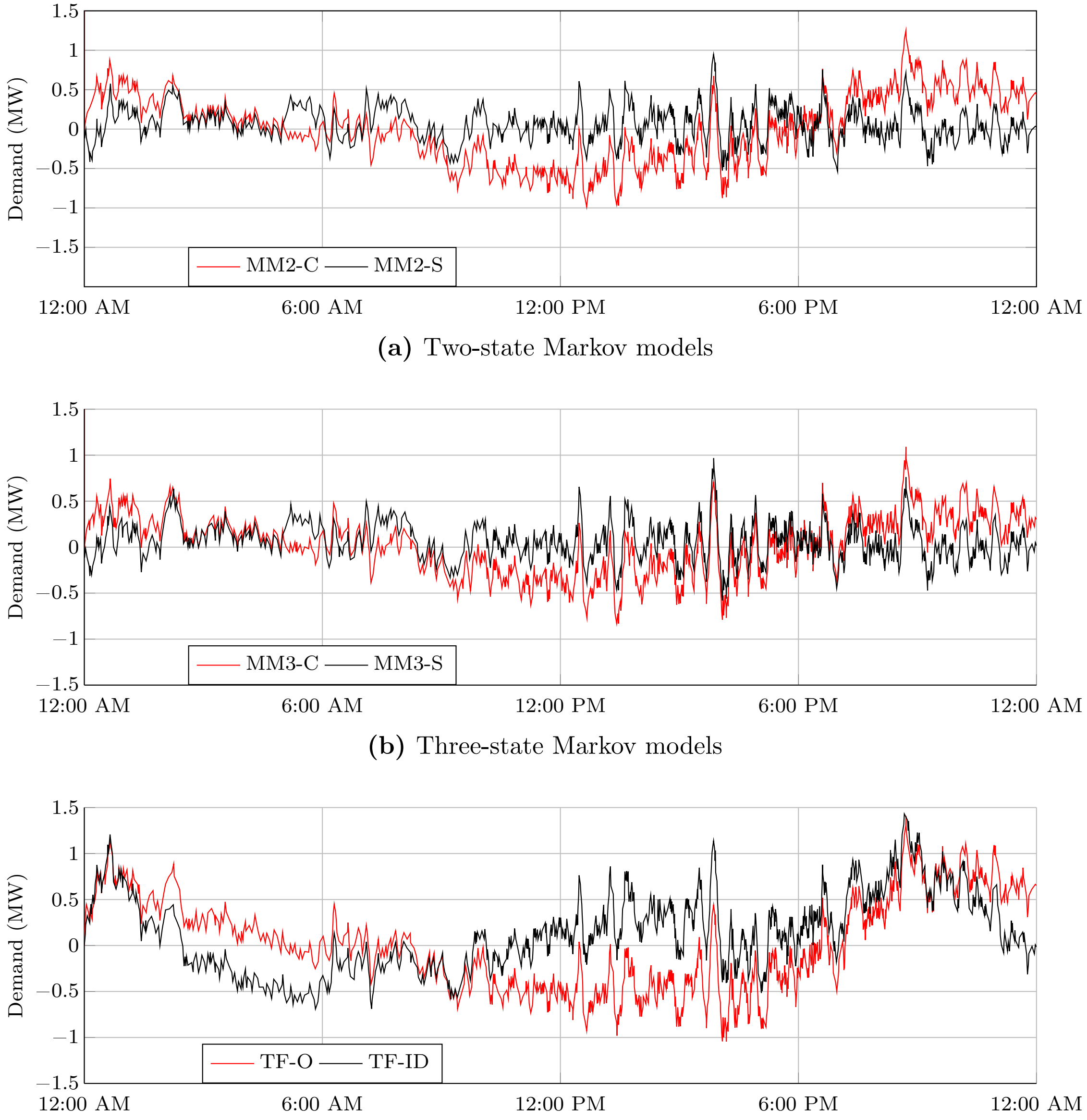
The predictions of TF-O and TF-ID have higher RMSE than either Markov-based aggregate model. This makes sense as the transfer function models are simpler models than the Markov-based models; whereas TF-O and TF-ID have two states and a steady-state demand term, the Markov-based models have 40 and 800 bins, respectively. Figure 4.2b shows that the TF-O prediction generally lags the aggregate demand and the TF-ID prediction does not. The errors in Fig. 4.3c also show that TF-O under-predicts the demand as it increases from 12:00 PM to 6:00 PM.



(a) Two-state Markov models

(b) Transfer function models

Figure 4.2: Time series of the aggregate demand and various model predictions over a portion of the simulated day



(a) Two-state Markov models

(b) Three-state Markov models

(c) Transfer function models

Figure 4.3: Prediction error of various models versus the aggregate demand

Table 4.3: Summary of Models and RMSE (kW) Values

Abbreviation	Base Aggregate Model	Details	RMSE (kW)
MM2-C	Two-State Markov Model	Set of models for different θ_t^o values; data for model computation generated using constant θ_t^o values	436.7
MM2-V	Two-State Markov Model	Set of models for different θ_t^o values; data for model computation generated using time-varying θ_t^o values	437.1
MM2-S	Two-State Markov Model	Set of models for different θ_t^o values and different θ_t^o trends; data for model computation generated using time-varying θ_t^o	226.2
MM3-C	Three-State Markov Model	Set of models for different θ_t^o values; data for model computation generated using constant θ_t^o values	320.9
MM3-V	Three-State Markov Model	Set of models for different θ_t^o values; data for model computation generated using time-varying θ_t^o values	322.9
MM3-S	Three-State Markov Model	Set of models for different θ_t^o values and different θ_t^o trends; data for model computation generated using time-varying θ_t^o	213.4
TF-O	Transfer Function Model	Single model; assumed transfer function structure of two poles and one zero; parameters computed using [82]	504.4
TF-ID	Transfer Function Model	Single model; transfer function structure of two poles and two zeros identified from historical model accuracy; parameters identified with historical input-output data	447.0

4.4 Chapter IV Conclusions

In this work, we benchmarked the prediction accuracy of three existing aggregate models and several variations of them within a common, detailed simulation scenario. The simulation scenario includes a time-varying outdoor temperature and detailed AC models that include a time-varying cooling capacity, COP, and power draw (when the AC is on) that all depend on the outdoor temperature. Results indicate that the three-state aggregate model generally performs better than the other aggregate models. Incorporating the temperature trend improves both Markov-based models and reduces the gap in prediction accuracy between the two. The transfer function model is the least accurate of the three aggregate models, most likely due to its simplicity and due to the fact that the simulation scenario differs substantially from the assumptions used to develop the model in [82]. The transfer function using input-output data to identify the parameters resulted in reduced prediction error. While the three-state aggregate model is the most accurate, it is more computationally complex than the other two models. The simpler two-state aggregate model offers similar performance, when including temperature trends into the model, at lower computational complexity.

Avenues of future work include the following: 1) developing a variation of the Markov-based models that accounts for times of little change in outdoor temperature; 2) deriving transfer function parameters for an AC aggregation undergoing a sinusoidal input rather than a step input, which better approximates realistic temperature changes; 3) investigating the identified transfer function structure and whether its parameters can be derived from the individual AC population; 4) including time-varying solar irradiance and internal heat gains within the AC models; and 5) prediction performance under aggregator control.

CHAPTER V

Exploring Connections Between a Multiple Model Kalman Filter and Dynamic Fixed Share with Applications to Demand Response [1]

State estimation and online learning are two approaches to estimate the state of a dynamic system as measurements arrive at sequential, discrete time-steps. The discrete-time Kalman filter (hereafter referred to simply as a Kalman filter) relies on a model-based update that advances the state estimate in time according to an assumed model of the underlying system, and a measurement-based update that incorporates newly arrived measurements of the system into the state estimate. While a Kalman filter uses a single model of the system to estimate the state, a multiple model Kalman filter (MMKF) [88, 89] uses a set of possible models to compute the state estimate, addressing situations where the model is unknown beforehand. The Kalman filter and MMKF make assumptions on the generative model of the data, i.e., that the system model is linear and that each model's error and measurement noise are normally distributed. The assumptions lead to a fixed structure for the state estimation equations, and a user implementing the algorithms can only tune the model-based parameters within the update equations.

While many online learning algorithms are model-free and data-driven, two recently developed online learning algorithms, Dynamic Mirror Descent (DMD) and Dynamic Fixed Share (DFS) [90], incorporate dynamic models and use both a model-based update and a measurement-based update, similarly to Kalman filtering algorithms. DMD is an online learning analogue to a Kalman filter, and DFS is comparable to a MMKF. DMD and DFS both estimate a dynamic system state without making assumptions of the generative model of the data. These online learning al-

[1] The work presented in this chapter was originally published in [86] and [87].

gorithms are based on online convex optimization [91], where a user-defined convex optimization problem is solved online, or at each time-step, to update estimates using each new measurement as it arrives. These algorithms are more flexible and more data-driven than the Kalman filter algorithms. Flexibility arises from the user's ability to use a nonlinear system model within the algorithm and the ability for the user to design the online, convex optimization problem that dictates the way that a new measurement is incorporated into the estimate. The algorithms are more data-driven in the sense that historical data can be used do design this convex optimization problem as well as to select the form of the possibly nonlinear model(s) along with its (their) parameters. Within a Kalman filter, the user can tune the parameters within the update equations to improve performance, but within DMD, the user can tune the equations themselves (as well as the parameters) to improve performance. However, DMD and DFS are not guaranteed to be the best estimator from specific class of estimators, unlike a Kalman filter, which is the best linear estimator out of all linear estimators.

In this chapter, we show that DMD can be constructed to produce identical state estimates to those produced by a Kalman filter, we compare the multiple model algorithms, DFS and a MMKF, and we apply DFS and a MMKF to a demand response simulation. DMD can produce state estimates that are identical to those of a Kalman filter by choosing the model to match that of a Kalman filter and then by properly selecting the user-defined functions/parameters within the convex optimization problem of DMD. We present two simple examples that illustrate the impact of the user-defined functions and parameters within DMD; the examples also provide empirical evidence that DMD can be constructed to produce the same estimates as a Kalman filter. We then build on this result and show that the user-defined functions/parameters within DFS can be constructed to produce identical states estimates to those produced by a MMKF. We then modify DFS to include three heuristics that are used to improve the performance of a MMKF in order to assess whether they can also be used to improve the performance of DFS. While these heuristics result in a suboptimal MMKF, they often improve its estimation accuracy in practice, which we will demonstrate empirically.

In our simulation study, we aim to estimate the time-varying aggregate demand of a population of residential air conditioners (ACs). Such estimates could be used as a feedback signal within a demand response algorithm that aims to coordinate the aggregate demand to provide services (e.g., frequency regulation) to the electricity grid. We compare the estimation error of various DFS and MMKF implementations, which all rely on a set of aggregate load models that vary in their relative accuracy over the course of the simulation horizon.

The contributions of this chapter are as follows: 1) we show that DMD can be constructed to produce identical updates to those of a Kalman filter; 2) we show that DFS can be constructed to produce identical estimates to those produced by a MMKF; 3) we present two simple examples illustrating the flexibility of the DMD algorithm; 4) we incorporate three heuristics used within MMKFs into DFS; and 5) we compare the performance of DFS and a MMKF within the demand response simulation.

The remainder of this chapter is organized as follows. Section 5.1 presents the general estimation problem; Section 5.2 summarizes the Kalman filter algorithms; Section 5.3 summarizes the DMD and DFS algorithms; Section 5.4 reviews the method to construct the functions/parameters within DMD to achieve the same estimates as a Kalman filter, derives a method to construct the functions/parameters within DFS to achieve the same estimates as a MMKF, and derives heuristic adjustments to DFS based on those commonly used in MMKFs; Section 5.5 provides simple examples of DMD implementations, and it provides empirical evidence that DMD can be constructed to produce the same estimates as a Kalman filter; Section 5.6 describes the demand response simulation study and its results; and Section 5.7 presents the conclusions of the chapter.

5.1 Estimation Problem

The general estimation problem considered within this work is to estimate the value of a dynamic system state using 1) *a priori* knowledge about the system and 2) measurements of the system as they arrive at sequential, discrete time-steps. For the Kalman filter methods summarized in Section 5.2, *a priori* knowledge corresponds to the assumption of a linear system model and the assumption of zero-mean, normally-distributed process and measurement noise. For the online learning methods summarized in Section 5.3, *a priori* knowledge corresponds to the model, which may be nonlinear, and the construction of the convex optimization problem that is solved at each time-step.

Each algorithm uses a model to advance an estimate in time, which we refer to as the model-based update, and then uses a measurement to adjust the estimate, which we refer to as the measurement-based update. At time-step k, the algorithm first advances its estimate of the system state to the next time-step using an assumed model of the system to produce an *a priori* estimate for time-step $k + 1$. The system then produces a measurement for time-step $k + 1$. The algorithm uses this new measurement to adjust the *a priori* estimate and compute the *a posteriori* estimate. The algorithm then uses the model to advance the estimate in time again, and the process repeats.

Furthermore, to address modeling uncertainty, there may be multiple versions of each algorithm running where each version incorporates a separate model from a set of models and forms its own estimate of the state. Then, the individual estimates are combined into an overall estimate. Models that are more accurate are given more weight and can dominate the overall estimate. These "multiple-model" algorithms can be used in situations where the system model is not known beforehand or when the system operates in different regimes best described by different models.

We denote the system state $x_k \in \mathcal{X}$, the *a priori* estimate as $\widehat{x}_k \in \mathcal{X}$, the *a posteriori* estimate as $\widetilde{x}_k \in \mathcal{X}$, and the measurement as $y_k \in \mathcal{Y}$. We assume that the domain of the state $\mathcal{X} \subset \mathbb{R}^p$ is a bounded, closed, convex feasible set, which is required by the online learning algorithms summarized in Section 5.3 but not by the Kalman filter algorithms summarized in Section 5.2. Finally, we assume that the domain of the measurements is $\mathcal{Y} \subset \mathbb{R}^q$. The set of N^{mdl} models is denoted $\mathcal{M}^{\mathrm{mdl}}$, and $i \in \mathcal{M}^{\mathrm{mdl}}$ indexes them.

5.2 Kalman Filter Algorithms

We next summarize the Kalman filter and MMKF algorithms to introduce notation. Section 5.2.1 describes the Kalman filter, and Section 5.2.2 describes the MMKF.

5.2.1 Kalman Filter

A Kalman filter can be viewed as a stochastic approach to estimating the state of a dynamic system where updates to the estimate rely on a system model that is assumed to be linear along with process and measurement noise that are assumed to be normally distributed. The assumed system model is

$$x_{k+1} = A_k \, x_k + \omega_k \tag{5.1}$$

$$y_k = C_k \, x_k + v_k, \tag{5.2}$$

where (5.1) advances the state in time and (5.2) relates the state to a measurement. In these equations, $x_k \in \mathbb{R}^p$ is the state, $\omega_k \in \mathbb{R}^p$ is the process noise (or modeling error), $y_k \in \mathbb{R}^q$ is the output (or observation/measurement), and $v_k \in \mathbb{R}^q$ is the measurement noise. Samples of ω_k and v_k are assumed to be IID and to follow separate zero-mean, normal distributions. Their respective covariances Q_k and R_k are each symmetric, positive definite matrices. The system is assumed to be observable, and the matrices A_k, C_k, Q_k, and R_k are assumed to be known.

The Kalman filter finds a state estimate that minimizes the mean squared estimation error based on the assumed system model. The model-based update is $\widehat{x}_{k+1} = A_k \, \widetilde{x}_k$. The measurement-based update is

$$\widetilde{x}_k = \widehat{x}_k + \widehat{P}_k C_k^T \left[C_k \widehat{P}_k C_k^T + R_k \right]^{-1} (y_k - C_k \widehat{x}_k) \, , \tag{5.3}$$

where $\widehat{P}_k$ is the estimation error covariance, which is known at each time-step. Equation (5.3) can be viewed as the solution to the following online, convex optimization problem [92]:

$$\min_{x, v_k} \quad v_k^{\mathrm{T}} R_k^{-1} v_k \; + \; (x - \widehat{x}_k)^{\mathrm{T}} \, \widehat{P}_k^{-1} \, (x - \widehat{x}_k) \tag{5.4}$$

$$\text{s.t.} \quad y_k = C_k x + v_k. \tag{5.5}$$

When implementing a Kalman filter, the user can choose the model parameters – A_k, C_k, Q_k, and R_k – but the mathematical equations for the model-based update and the measurement-based update are otherwise fixed.

5.2.2 Multiple Model Kalman Filter

The MMKF uses a set of N^{mdl} independent Kalman filters that each run in parallel using one model from $\mathcal{M}^{\text{mdl}}$, and the MMKF combines the estimates of each Kalman filter into an overall estimate. Each model $i \in \mathcal{M}^{\text{mdl}}$ satisfies the assumptions of the Kalman filter and has corresponding matrices denoted A_k^i, C_k^i, Q_k^i, and R_k^i. We denote the estimate using model $i \in \mathcal{M}^{\text{mdl}}$ as $\widehat{x}_k^i$ and the covariance of the output estimation error as $\widehat{P}_k^{\text{y},i} = C_k^i \; \widehat{P}_k^i \; (C_k^i)^T + R_k^i$. Finally, we define a quantity $d_k^{\text{y}}(\widehat{y}_k^i)$ that is the squared Mahalanobis distance of Kalman filter i's output estimate versus the measurement y_k, i.e., $d_k^{\text{y}}(\widehat{y}_k^i) = (\widehat{y}_k^i - y_k)^T (\widehat{P}_k^{\text{y},i})^{-1} (\widehat{y}_k^i - y_k)$ with $\widehat{y}_k^i = C_k^i \widehat{x}_k^i$.

The equations that form the weights and combine the estimates within the MMKF are

$$h(y_k|m^i) = \left[(2\pi)^{q/2}\sqrt{|\widehat{P}_k^{\mathrm{y},i}|}\right]^{-1} \exp\left(-\frac{1}{2}d_k^{\mathrm{y}}(\widehat{y}_k^i)\right) \tag{5.6}$$

$$w_{k+1}^i = \frac{h(y_k|m^i)\ w_k^i}{\sum_{j\in\mathcal{M}^{\mathrm{mdl}}} h(y_k|m^j)\ w_k^j} \tag{5.7}$$

$$\widehat{x}_{k+1} = \sum_{i\in\mathcal{M}^{\mathrm{mdl}}} w_{k+1}^i\ \widehat{x}_{k+1}^i, \tag{5.8}$$

with $i \in \mathcal{M}^{\mathrm{mdl}}$ for (5.6) and (5.7). In the above, (5.6) is a conditional probability of the likelihood of the observation y_k given that the underlying system is model m^i, (5.7) is a weighting function where the weights can be viewed as a probability that model m^i matches the underlying system model, and (5.8) forms the overall estimate using the individual Kalman filter estimates and their weights.

Note that other algorithms exist for situations where the system models switches as time progresses, e.g., the first- and second-order generalized pseudo-Bayesian algorithms and the interacting multiple model algorithm, all from [89]. However, these algorithms require the probability of transitioning from one model to another during a time-step to be known *a priori*; here, we assume that this is unknown *a priori*. In addition, whereas online learning assumes that the underlying models (or “experts”) operate independently, the first- and second-order generalized pseudo-Bayesian algorithms and the interacting multiple model algorithm do not make this assumption. As a result, we do not include further discussion of these algorithms.

5.3 Online Learning Algorithms

In this section we summarize the DMD and DFS algorithms, which were originally developed in [90]. Section 5.3.1 describes DMD, and Section 5.3.2 describes DFS.

5.3.1 Dynamic Mirror Descent

The DMD algorithm uses a user-defined convex optimization formulation and a single model of the underlying system to estimate its state. Specifically, DMD uses the convex optimization formulation to adjust the estimate $\widehat{x}_k$ using the new measurement y_k, and then applies the model to advance the adjusted estimate to the next time-step. The DMD algorithm formulation is [90]

$$\widetilde{x}_k = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, \eta^{\mathrm{s}} \left(\nabla \ell_k(\widehat{x}_k)\right)^T x + D\left(x \| \widehat{x}_k\right) \tag{5.9}$$

$$\widehat{x}_{k+1} = \Phi(\widetilde{x}_k), \tag{5.10}$$

where (5.9) computes the adjusted estimate and (5.10) applies the model. In these equations, $\Phi(\cdot)$ is the (possibly nonlinear) model, $\eta^{\mathrm{s}} > 0$ is a user-defined step size, and $\nabla \ell_k(\widehat{x}_k)$ is the gradient or subgradient of the convex loss function $\ell_k(\widehat{x}_k)$, which computes the error on the output estimate. The function $D(x \| \widehat{x}_k)$ is a Bregman divergence, which is similar to a distance function. As an example, we could use $D(x \| \widehat{x}_k) = \|x - \widehat{x}_k\|_2^2$ and $\ell_k(\widehat{x}_k) = \|C\widehat{x}_k - y_k\|_2^2$, where the matrix C translates the state estimate into an output estimate. The choice of loss function establishes the relationship between output estimate errors and state estimate errors, since the gradient of this function helps to determine how the state estimate is adjusted based on the output estimate errors. The choice of the Bregman divergence helps to establish the relationship between estimation errors within the different components of the state. The parameter η^{s} controls how closely the algorithm matches the output estimates with the measurements (by adjusting the state estimate) versus trusting the models. Section 5.5 presents one example to show the impact of η^{s} and one example to show the impact of the choice of the divergence and loss functions.

5.3.2 Dynamic Fixed Share

The DFS algorithm uses a set of DMD algorithms, each using a separate (possibly nonlinear) model from $\mathcal{M}^{\mathrm{mdl}}$, as experts (i.e., algorithms that generate estimates) into the Fixed Share Algorithm developed in [93]. Similar to a MMKF, DFS forms an overall state estimate from the N^{mdl} experts. The DFS algorithm's weight updates and overall estimate are those of the Fixed Share Algorithm where the estimates $\widehat{x}_k^i$ for $i \in \mathcal{M}^{\mathrm{mdl}}$ are generated using DMD:

$$w_{k+1}^i = \frac{\lambda}{N^{\mathrm{mdl}}} + (1 - \lambda)\, \frac{w_k^i \, \exp\left(-\eta^r \, \ell_k\left(\widehat{x}_k^i\right)\right)}{\sum_{j=1}^{N^{\mathrm{mdl}}} w_k^j \, \exp\left(-\eta^r \, \ell_k\left(\widehat{x}_k^j\right)\right)} \tag{5.11}$$

$$\widehat{x}_{k+1} = \sum_{i \in \mathcal{M}^{\mathrm{mdl}}} w_{k+1}^i \, \widehat{x}_{k+1}^i \tag{5.12}$$

with $i \in \mathcal{M}^{\mathrm{mdl}}$ for (5.11). Equation (5.11) updates the weight of each expert, where w_k^i is the weight of expert i, $\lambda \in (0, 1)$ is a user-defined parameter that sets the minimum weight of each expert, and η^{r} is a user-defined parameter that scales the total accumulated loss (which is related to output estimation errors). Equation (5.12) combines the individual estimates into an overall estimate $\widehat{x}_k$. Setting η^{r} to larger values forces $\exp\left(-\eta^{r}\,\ell_k\left(\widehat{x}^i_k\right)\right)$ to be near one regardless of $\ell_k\left(\widehat{x}^i_k\right)$, and this results in faster changes to the weights.

5.4 Connections Between the Kalman Filtering and Online Learning Algorithms

Section 5.4.1 presents a method to construct the functions/parameters within DMD to produce estimates identical to those produced by a Kalman filter. Section 5.4.2 builds on this result and presents a method to construct the functions/parameters used within DFS to produce estimates identical to those produced by a MMKF. Finally, Section 5.4.3 adapts several heuristics commonly used within MMKFs to DFS.

5.4.1 Producing Identical Estimates with DMD and a Kalman Filter

We construct DMD by choosing the model, user-defined parameters, and user-defined functions. As with a Kalman filter, we assume as linear model and that the model matrices A_k, C_k, Q_k, and R_k are known, and additionally, that $\widehat{P}_k$ is known. Choosing the model used within DMD to be identical to that used within the Kalman filter results in the same model-based update. The remaining step is to construct the convex program (5.9) such it corresponds to (5.3).

In (5.9) we have the ability to choose the η^{s}, $D(x\|\widehat{x}_k)$, and $\ell_k(\widehat{x}_k)$. Recall that the measurement-based update in a Kalman filter is

$$\widetilde{x}_k = \widehat{x}_k + \widehat{P}_k C_k^T \left(\widehat{P}_k^{\mathrm{y}}\right)^{-1}\left(y_k - C_k\widehat{x}_k\right),$$

and the measurement-based update in DMD is

$$\widetilde{x}_k = \arg\min_{x\in\mathcal{X}} \eta^{\mathrm{s}}\left(\nabla\ell_k(\widehat{x}_k)\right)^T\, x + D\left(x\|\widehat{x}_k\right).$$

Choosing the Bregman divergence as $D\left(x\|\widehat{x}_k\right) = \frac{1}{2}\left(x-\widehat{x}_k\right)^T \widehat{P}_k^{-1}\left(x-\widehat{x}_k\right)$, setting $\eta^{\mathrm{s}} = 1$, and solving for the closed form solution of the convex program (i.e., taking the gradient with respect to x and setting this equal to zero) gives

$$\widetilde{x}_k = \widehat{x}_k + \widehat{P}_k\left(-\nabla\ell_k(\widehat{x}_k)\right). \tag{5.13}$$

Note that with the appropriate selection of the Bregman divergence, the structure of DMD's measurement-based update closely matches that of the Kalman filter, and the remaining step is to choose $\ell_k(\widehat{x}_k)$ appropriately. Noting that $\frac{\delta}{da}\left[(Ma-b)^T V(Ma-b)\right] = 2M^T V(Ma-b)$, we choose

$$\ell_k(\widehat{x}_k) = \frac{1}{2}\left(C_k\widehat{x}_k - y_k\right)^T (\widehat{P}_k^{\mathrm{y}})^{-1}\left(C_k\widehat{x}_k - y_k\right),$$

and then $-\nabla\ell_k(\widehat{x}_k) = C_k^T(\widehat{P}_k^{\mathrm{y}})^{-1}\left(y_k - C_k\widehat{x}_k\right)$. Plugging $-\nabla\ell_k(\widehat{x}_k)$ into (5.13) gives the same measurement-based update as the Kalman filter. Section 5.5 presents a simple example that shows DMD producing identical estimates as a Kalman filter.

5.4.2 Producing Identical Estimates with DFS and a MMKF

Since DMD can be constructed to produce identical estimates to those produced by a Kalman filter, all that is needed to produce identical updates with DFS and MMKF is to ensure that the updates to the weights w_k^i are equal. We first set $\lambda = 0$, $\eta^{\mathrm{r}} = 1$, and use the loss function developed in Section 5.4.1. The resulting weight update in DFS is

$$h^{\mathrm{DFS}}(y_k|m^i) = \exp\left(-\ell_k(\widehat{x}_k^i)\right) \tag{5.14}$$

$$w_{k+1}^i = \frac{h^{\mathrm{DFS}}(y_k|m^i)\ w_k^i}{\sum_{j\in\mathcal{M}^{\mathrm{mdl}}} h^{\mathrm{DFS}}(y_k|m^j)\ w_k^j} \tag{5.15}$$

for $i \in \mathcal{M}^{\mathrm{mdl}}$. Note that (5.15) corresponds exactly to (5.7), and the remaining step is to construct $h^{\mathrm{DFS}}(y_k|m^i)$ to equal $h(y_k|m^i)$.

To make $h^{\mathrm{DFS}}(y_k|m^i)$ equal $h(y_k|m^i)$, we will scale $\widehat{P}_k^{\mathrm{y},i}$ by a parameter β^i such that $(2\pi)^{-q/2}\ (|\beta^i\widehat{P}_k^{\mathrm{y},i}|)^{-1/2} = 1$. The parameter β^i will be positive since $\widehat{P}_k^{\mathrm{y},i}$ is positive definite, and we also define $\alpha^i \triangleq \sqrt{\beta^i}$. Scaling $\widehat{P}_k^{\mathrm{y},i}$ by β^i amounts to adjusting our belief of the accuracy of the output estimate, and the output equations should be changed accordingly. To see this, recall that $\widehat{P}_k^{\mathrm{y},i} = C_k^i\widehat{P}_k^i(C_k^i)^T + R_k^i$, and then $\beta^i\widehat{P}_k^{\mathrm{y},i} = (\alpha^i C_k^i)\widehat{P}_k^i(\alpha^i C_k^i)^T + \beta^i R_k^i$, i.e., the scaling parameters only appear with the output-related quantities. As a result, the outputs are scaled, i.e., $\widehat{y}_k^i = C_k^i\widehat{x}_k^i$ becomes $\alpha^i\widehat{y}_k^i = (\alpha^i C_k^i)\widehat{x}_k^i$, the measurement noise covariance R_k^i becomes $\beta^i R_k^i$, and the output $y_k = C_k^i x_k + v_k$ becomes $\alpha^i y_k = \alpha^i C_k^i x_k + \alpha^i v_k$.

To determine β^i, we use the property of determinants where $|\gamma V| = \gamma^n|V|$ for an $n\times n$ matrix V and scalar γ. To set β^i, we replace $\widehat{P}_k^{\mathrm{y},i}$ with $(\beta^i\widehat{P}_k^{\mathrm{y},i})$, and then solve for β^i:

$$1 = \frac{1}{(2\pi)^{q/2}\sqrt{|\beta^i \widehat{P}_k^{\mathrm{y},i}|}} \tag{5.16}$$

$$\implies \beta^i = \sqrt[q]{(2\pi)^{-q}\ |\widehat{P}_k^{\mathrm{y},i}|^{-1}}. \tag{5.17}$$

Using this scaling to adjust $h(y_k|m^i)$ within the MMKF gives

$$h(y_k|m^i) = \exp\left(-\frac{1}{2}(\alpha^i\widehat{y}_k^i - \alpha^i y_k)^T(\beta^i\widehat{P}_k^{\mathrm{y},i})^{-1}(\alpha^i\widehat{y}_k^i - \alpha^i y_k\right),$$

where the scaling eliminated the constant in front of the exponential, and where the scaling cancels out in the exponent. Applying the scaling to $h^{\mathrm{DFS}}(y_k|m^i)$ gives $h^{\mathrm{DFS}}(y_k|m^i) = h(y_k|m^i)$, and so the updates to the weights are equal.

However, since we have modified the expression for $h(y_k|m^i)$ within the MMKF, we must carry the scaling through each MMKF equation. Within the Kalman gain, we see that replacing the output matrices and the estimated output with their scaled values results in a scaled gain:

$$\overline{K}_k^i = \widehat{P}_k^i\alpha^i(C_k^i)^T\left[\alpha^i C_k^i\widehat{P}_k^i\alpha^i(C_k^i)^T + \beta^i R_k^i\right]^{-1} \tag{5.18}$$

$$= \frac{\alpha^i}{\beta^i}K_k^i. \tag{5.19}$$

However, the scaling cancels out within the measurement-based update:

$$\widetilde{x}_k^i = \widehat{x}_k^i + \frac{\alpha^i}{\beta^i}K_k^i\left(\alpha^i y_k - \alpha^i C_k^i\widehat{x}_k^i\right) \tag{5.20}$$

$$= \widehat{x}_k^i + K_k^i\left(y_k - C_k^i\widehat{x}_k^i\right) \tag{5.21}$$

for $i \in \mathcal{M}^{\mathrm{mdl}}$. The model-based update does not contain any scaled quantities, and the scaling factor also cancels out in the update to the state estimation error covariance.

5.4.3 Adapting MMKF Heuristics to DFS

We next show that several heuristic adjustments that are commonly used within MMKFs can be readily incorporated into DFS by modifying the weight update. The heuristics include 1) setting a minimum weight such that a model's weight does not go to zero [89], 2) using exponential decay within the likelihood function [94], and 3) using a sliding window within the likelihood function [94].

To incorporate these methods into DFS, we can change the weighting equation (5.11). The first heuristic is equivalent to setting λ to a value greater than zero within DFS. The second heuristic is equivalent to adjusting (5.11) to

$$w_{k+1}^{i} = \frac{\lambda}{N^{\mathrm{mdl}}} + (1-\lambda)\frac{(w_k^i)^{\gamma}\,\exp\left(-\eta^{r}\ell_k(\widehat{x}_k^i)\right)}{\sum\limits_{j=1}^{N^{\mathrm{mdl}}}(w_k^j)^{\gamma}\,\exp\left(-\eta^{r}\ell_k(\widehat{x}_k^j)\right)}. \tag{5.22}$$

The parameter $\gamma \in (0,1)$ reduces the impact of the previously accumulated loss on the model's weight, where smaller values of γ reduce the effect more dramatically. The third method is equivalent to adjusting (5.11) to

$$w_{k+1}^{i} = \frac{\lambda}{N^{\mathrm{mdl}}} + (1-\lambda)\,\frac{\prod\limits_{t=k-N^{\ell}}^{k}\exp\left(-\eta^{r}\,\ell_t\left(\widehat{x}_t^i\right)\right)}{\sum\limits_{j=1}^{N^{\mathrm{mdl}}}\prod\limits_{t=k-N^{\ell}}^{k}\exp\left(-\eta^{r}\,\ell_t\left(\widehat{x}_t^j\right)\right)}, \tag{5.23}$$

where N^{ℓ} is the number of time-steps within the sliding window. By using (5.22) and (5.23), it is possible to discount and exclude historical estimation errors (and their resulting losses), which leads to a more dynamic set of weights that depend on the recent estimation accuracy. Simulations presented in Section 5.6 investigate how these weighting functions affect the overall estimates within DFS and a MMKF.

5.5 Examples of DMD Implementations

In this section, we present two examples that demonstrate the influence of several of the user-defined functions and parameters within DMD. The first example shows how the choice of η^{s} impacts the estimate in the presence of measurement noise. The second example illustrates how the choice of divergence and loss functions impact the estimates generated by (5.9), and it provides empirical evidence for the result in Section 5.4.1 that DMD can be constructed to produce identical estimates to those of a Kalman filter. These examples are constructed to isolate impact of the component of interest. In reality, the various parameters and function choices influence each other in nontrivial ways, which generally cannot be known *a priori*.

In the examples below, the plant model, whose state we are trying to estimate, consists of (5.1) and (5.2), where $C = \begin{bmatrix} 0 & 1 \end{bmatrix}$ and

$$A = \begin{bmatrix} \cos(\pi/500) & -\sin(\pi/500) \\ \sin(\pi/500) & \cos(\pi/500) \end{bmatrix}. \tag{5.24}$$

The the state is $x_k \in \mathbb{R}^2$, its initial value is $x_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$, $\omega_k \in \mathbb{R}^2$, and $v_k \in \mathbb{R}$. We assume that ω_k and v_k satisfy the assumptions of a Kalman filter, and their time-invariant covariances are $Q \in \mathbb{R}^{2\times 2}$ and $R \in \mathbb{R}$, respectively, where we detail their values in each example.

5.5.1 Varying the Gradient Descent Step Size η^{s}

The parameter η^{s} influences how closely DMD adjusts the state estimate $\widehat{x}_k$ to match the (possibly noisy) measurement versus trusting the predictions of the system model $\Phi(\cdot)$. In this example, we assume the plant model contains no process noise, i.e., $Q = 0$, and the measurement noise covariance is $R = 1$. The DMD model $\Phi(\widetilde{x}_k)$ is set to the plant model (5.1) and (5.2) excluding ω_k and v_k. The divergence is set to $D(x\|\widehat{x}_k) = \frac{1}{2}\|x - \widehat{x}_k\|_2^2$ and the loss function is set to $\ell_k(\widehat{x}_k) = \frac{1}{2}\|C\widehat{x}_k - y_k\|_2^2$. The resulting closed-form measurement-based update (5.9) is

$$\widetilde{x}_k = \widehat{x}_k + \eta^{\mathrm{s}} C^{\mathrm{T}}(y_k - C\widehat{x}_k). \tag{5.25}$$

We apply DMD for two different values of η^{s} (i.e., $\eta_k^0 = 0.0$ and $\eta_k^1 = 1.0$, where the resulting estimates are denoted $\widehat{x}_k^0$ and $\widehat{x}_k^1$, respectively) and compare the results. All estimates are initialized at the true state.

Figure 5.1 presents the resulting time series of the second elements of x_k, $\widehat{x}_k^0$, and $\widehat{x}_k^1$; we exclude time series of the first elements as they exhibit similar characteristics. The second term of (5.25) is 0 for $\widehat{x}_k^0$, and so there is no adjustment to the state estimate based on the measurement. As a result, $\widehat{x}_k^0$ matches x_k exactly because the model within DMD exactly matches the plant model. Alternatively, for $\widehat{x}_k^1$, the convex program adjusts the state estimate to match the noisy measurements rather than trusting DMD's model, resulting in significant estimation error.

5.5.2 Varying the Choice of Divergence and Loss Functions

The choice of the divergence and loss functions within DMD influences the algorithm's measurement-based adjustments. In this example, we vary DMD's measurement-based update by using two choices for the divergence and loss functions – one

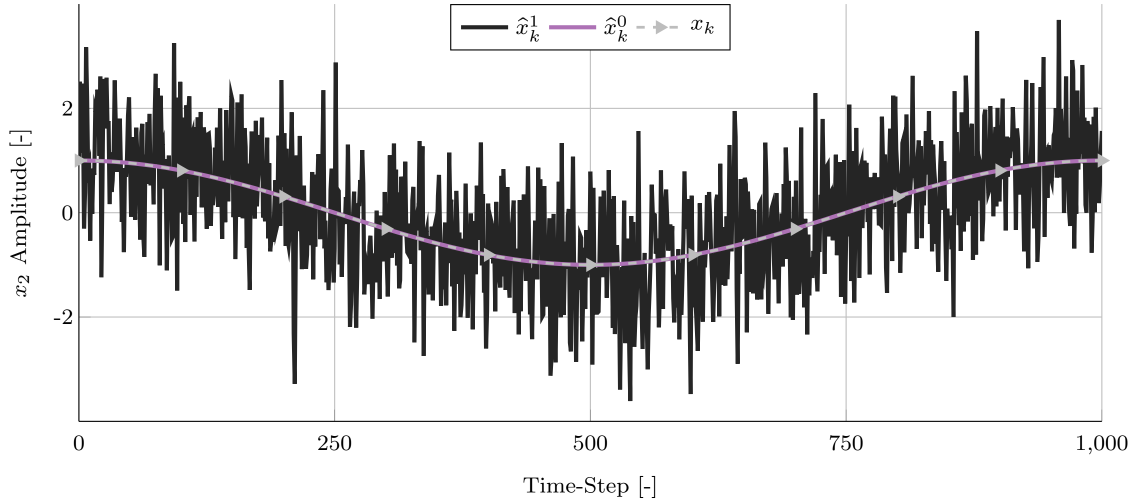


Figure 5.1: Time series of the second element of x_k, $\widehat{x}_k^0$, and $\widehat{x}_k^1$ for the example in Section 5.5.1

that includes covariance matrices explicitly and one that does not. We also simulate a Kalman filter to empirically show that the DMD estimates match those of a Kalman filter when the divergence and loss functions are constructed as described in Section 5.4.1.

In this example, we assume the measurement noise covariance is $R = 2$, and the process noise covariance is

$$Q = \begin{bmatrix} 0.25 & 0.1 \\ 0.1 & 0.25 \end{bmatrix}.$$

We construct a steady-state discrete-time Kalman filter, whose estimates are denoted $\widehat{x}_k^{\mathrm{KF}}$, using the underling system model and covariances. Both DMD formulations use $\eta^{\mathrm{s}} = 1$. The first DMD formulation, whose estimates are denoted $\widehat{x}_k^{\mathrm{a}}$, uses the same loss function, divergence function, and resulting measurement-based update equation as in the previous example. The second DMD formulation, whose estimates are denoted $\widehat{x}_k^{\mathrm{b}}$, uses the divergence and loss functions needed to produce measurement-based updates that are equivalent to those of the Kalman filter, i.e., (5.3), and we set $\widehat{P}_k = \overline{P}$ and $\widehat{P}_t^{\mathrm{y}} = \left[C\overline{P}C^T + R\right]$. The quantity $\overline{P}$ is a steady-state estimation error covariance that the Kalman filter converges to if the system is time-invariant. Note that the second formulation explicitly includes accurate model prediction error statistics via the covariances, whereas the first estimate implicitly assumes the covariances are identity matrices.

Figure 5.2 presents the time series of x_k, $\widehat{x}_k^{\mathrm{a}}$, $\widehat{x}_k^{\mathrm{b}}$ and $\widehat{x}_k^{\mathrm{KF}}$. Note that the estimates $\widehat{x}_k^{\mathrm{b}}$ and $\widehat{x}_k^{\mathrm{KF}}$ coincide exactly, empirically supporting our claim that we can choose the

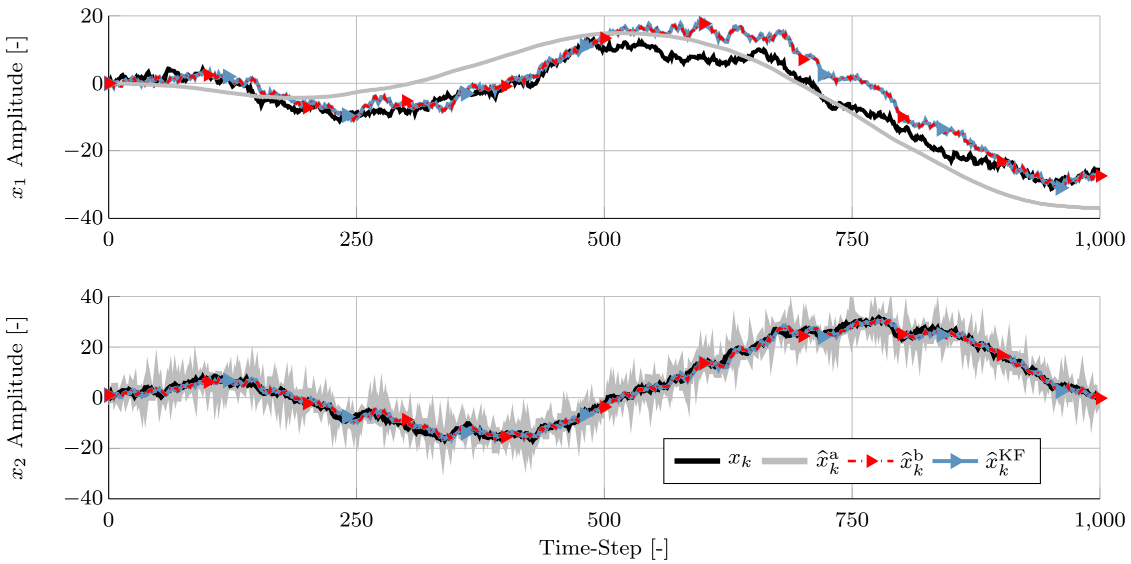


Figure 5.2: Time series of x_k, $\widehat{x}_k^{\mathrm{a}}$, $\widehat{x}_k^{\mathrm{b}}$, and $\widehat{x}_k^{\mathrm{KF}}$ for the example in Section 5.5.2

DMD model, divergence function, and loss function to achieve a measurement-based update equivalent to that of Kalman filter. In estimating the second element of x_k, we first note that both $\widehat{x}_k^{\mathrm{a}}$ and $\widehat{x}_k^{\mathrm{b}}$ follow the general trajectory of the true state, but $\widehat{x}_k^{\mathrm{b}}$ is noticeably smoother than $\widehat{x}_k^{\mathrm{a}}$. By including the covariance matrices into the measurement-based update, $\widehat{x}_k^{\mathrm{b}}$ is better able to account for the measurement noise resulting in a less erratic estimate and reduced estimation error versus $\widehat{x}_k^{\mathrm{a}}$. In estimating the first element of x_k, both methods have significant deviations from the true state value; however, the root mean square estimation error in $\widehat{x}_k^{\mathrm{b}}$ is smaller that of $\widehat{x}_k^{\mathrm{a}}$, indicating that $\widehat{x}_k^{\mathrm{b}}$ is more accurate over the duration of the simulation. Again, the inclusion of accurate statistical information into the measurement-based update has led to a more accurate estimate.

It should be noted that this example was constructed such that the Kalman filter is the optimal estimator. In reality, the assumptions of the Kalman filter rarely hold. A Kalman filter can still be applied with varying degrees of success, but it may not be the optimal estimator. The DMD algorithm relaxes some of the underlying assumptions, which allows greater flexibility in designing the updates, but the theoretical guarantees of the Kalman filter do not apply. Additionally, it should be noted that in this example we assume we have a perfect estimate of the covariance matrices.

5.6 Case Studies

In this section, we use the MMKF and DFS algorithms to estimate the total active power demand of an aggregation of residential ACs. Specifically, we investigate the estimation accuracy of DFS and a MMKF, with and without the scaling in Section 5.4.2, and with and without the heuristics in Section 5.4.3, where each expert (i.e., instance of DMD) in DFS is a separate Kalman filter from the MMKF. Such estimates could be used as a feedback signal within a demand response algorithm that aims to coordinate the aggregate demand to provide services (e.g., frequency regulation) to the electricity grid. However, here we assume that the loads are not controlled (making the estimation problem more difficult).

5.6.1 Problem Setup and Simulation Details

Figure 5.3 gives the block diagram of the estimation problem. The plant is our representation of the physical system. It consists of a set of n^{AC} residential AC models along with other loads within a distribution network. The demand response provider would like an estimate of the aggregate AC demand (i.e., the flexible demand), but it only has a measurement of the total demand. It subtracts an estimate of the demand of the other loads from the measurement of the total demand to obtain a noisy estimate of the aggregate AC demand. This noisy estimate is used as a measurement within the DFS/MMKF algorithms to obtain a better estimate of the aggregate AC demand.

Each AC within the plant is modeled similarly to Section 2.2.1.2, where the equations from [95] and the notation are the following:

$$\theta_{k+1} = a\,\theta_k + (1-a)\,(\theta_k^{\mathrm{o}} - m_k \Lambda P) \tag{5.26}$$

$$m_{k+1} = \begin{cases} 0 & \text{if } \theta_{k+1} < \theta^{\mathrm{set}} - \frac{\theta^{\mathrm{db}}}{2} \\ 1 & \text{if } \theta_{k+1} > \theta^{\mathrm{set}} + \frac{\theta^{\mathrm{db}}}{2} \\ m_k & \text{otherwise,} \end{cases} \tag{5.27}$$

where $\theta_k \in \mathbb{R}$ is the internal air temperature of the house, $m_k \in \{0,1\}$ is the AC's on/off switch value, $\theta_k^{\mathrm{o}} \in \mathbb{R}$ is the time-varying outdoor temperature, $a = \exp(-\Delta t/\Lambda C)$, and $\Delta t \in \mathbb{R}$ is the time-step. The remaining parameters are sampled from uniform distributions with ranges from [17], where $\theta^{\mathrm{set}} \in \mathbb{R}$ is the temperature set-point, $\theta^{\mathrm{db}} \in \mathbb{R}$ is the temperature dead-band, $\Lambda \in \mathbb{R}$ is the thermal resistance, $C \in \mathbb{R}$ is the thermal capacitance, $\eta \in \mathbb{R}$ is the coefficient of performance, and $P \in \mathbb{R}$ is the energy transfer

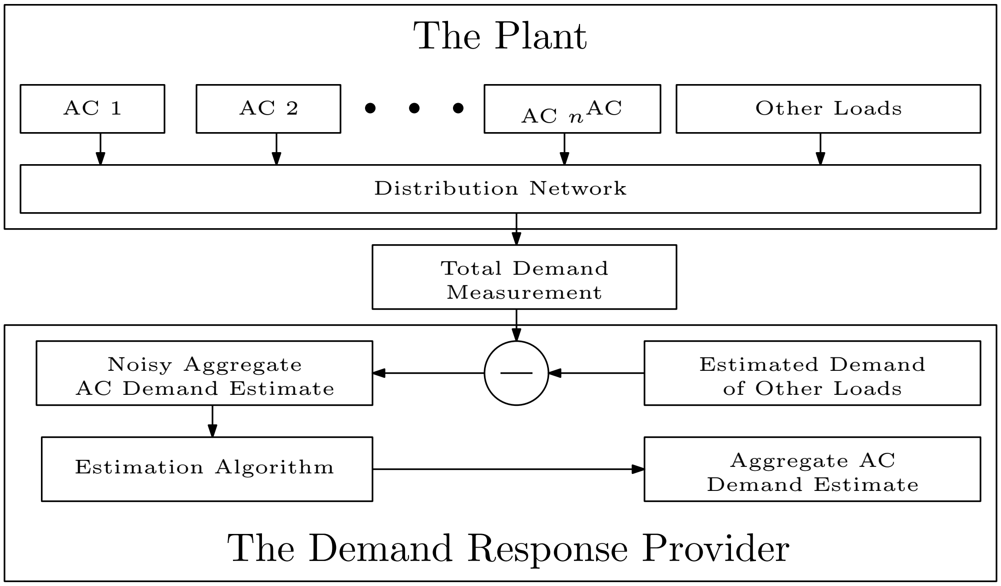

The Plant
AC 1
AC 2
AC n^{AC}
Other Loads
Distribution Network
Total Demand Measurement
—
Noisy Aggregate AC Demand Estimate
Estimated Demand of Other Loads
Estimation Algorithm
Aggregate AC Demand Estimate
The Demand Response Provider

Figure 5.3: Block diagram of the estimation problem in Chapter V.

rate. The aggregate power draw of the set of ACs is $y_k^{\mathrm{Agg}} = \sum_{i=1}^{n^{\mathrm{AC}}} m_k^i P^i (\eta^i)^{-1}$. In this work, we assume the estimation error v_k associated with the demand of the other loads is normally distributed with variance R. Therefore, the noisy aggregate AC demand estimate (i.e., the measurement used within the DFS and MMKF algorithms) is $y_k = y_k^{\mathrm{Agg}} + v_k$.

We simulate the plant with $n^{\mathrm{AC}} = 1000$, $\Delta t = 4$ seconds, and with a time-varying outdoor temperature over the course of six hours. The time-varying outdoor temperature corresponds to one period of a sine wave initialized at 31°C and varying from $28 - 34$°C over the course of the simulation. We set the standard deviation of the measurement noise, i.e., $\sqrt{R}$, equal to 10% of the AC demand's average value over the simulation.

Each algorithm uses a set of dynamic models, developed in [17, 44], that capture the aggregate AC demand. Each model is a linear, time-invariant autonomous system $x_{k+1} = Ax_k$, $y_k = Cx_k$, where the state $x_k \in \mathbb{R}^2$ captures the portion of ACs that are on versus off, $A \in \mathbb{R}^{2\times 2}$ is a transposed Markov transition matrix, and $C \in \mathbb{R}^{1\times 2}$ multiplies the the portion of ACs switched on by a scalar, resulting in the aggregate AC demand $y \in \mathbb{R}$. Both A and C are a function of the outdoor temperature. We use $N^{\mathrm{mdl}} = 3$ aggregate models, where A and C are identified by simulating a set of ACs with the same parameter distributions as those within the plant at outdoor temperatures $\theta^{\mathrm{o}} = 28$, 31, and 34°C. We denote these aggregate models as m^{28}, m^{31}, and m^{34}.

Table 5.1: Summary of Algorithms and Their RMS Estimation Errors (kW)

Abbreviation	Details	RMS Error (kW)
MMKF	The standard MMKF algorithm without any heuristics	104.5
MMKF-S	A MMKF with scaling performed according to Section 5.4.2	104.5
MMKF-M	A MMKF using a minimum weight for each model, i.e., using an equation similar to (5.11) as the weight update	63.4
MMKF-W	A MMKF using a sliding window weight update and a minimum weight for each model	-
MMKF-E	A MMKF using an exponential decay weight update and a minimum weight for each model	61.1
DFS-S	DFS with scaling performed according to Section 5.4.2	104.5
DFS-M	DFS with the standard weight update (5.11), which includes a minimum weight for each model	61.9
DFS-W	DFS using the sliding window weight update (5.23), which includes a minimum weight for each model	61.4
DFS-E	DFS using the exponential decay weight update (5.22), which includes a minimum weight for each model	60.9

The algorithm implementations are summarized in Table 5.1. We choose the functions within DFS such that the updates resemble the updates of a MMKF. However, for DFS-M, DFS-W, and DFS-E, we do not use the scaling from Section 5.4.2 and we tune η^{r} (which does not appear in the MMKF weight update) resulting in different performance between the comparable DFS and MMKF algorithms. By comparing MMKF and MMKF-S we see that the MMKF scaling in Section 5.4.2 achieves approximately the same performance in this case, and by comparing MMKF-S and DFS-S we can verify that the algorithms produce identical updates. We set $\lambda = 1e{-}5$ in DFS-M, DFS-W, DFS-E, and MMKF-M (which uses a weight update similar to (5.11)), which allows a single model to dominate the overall estimate if one proves to be the most accurate. We set η^{r} to 0.8 in DFS-M, 0.5 in DFS-W, and 1.2 in DFS-E. We set the window duration to $N^{\ell} = 250$ time-steps in DFS-W and MMKF-W. We set the exponential decay parameter to $\gamma = 0.995$ in DFS-E and MMKF-E. The values of η^{r}, N^{ℓ}, and γ were tuned qualitatively in the given simulation scenario to provide weights that are responsive but not overly erratic, e.g., from measurement noise.

In order to compare the performance of the weight updates we need to ensure that the estimates $\widehat{x}_k^i$ for $i \in \mathcal{M}^{\mathrm{mdl}}$ are the same within each implementation. Since DMD can be constructed to produce identical updates to a Kalman filter, we implement an identical set of Kalman filters within each DFS/MMKF implementation. In each Kalman filter, we set the measurement noise covariance to R and compute the process noise covariance based on the estimation error of the model. Note that the Kalman filters are sub-optimal estimators since the process noise is not normally distributed.

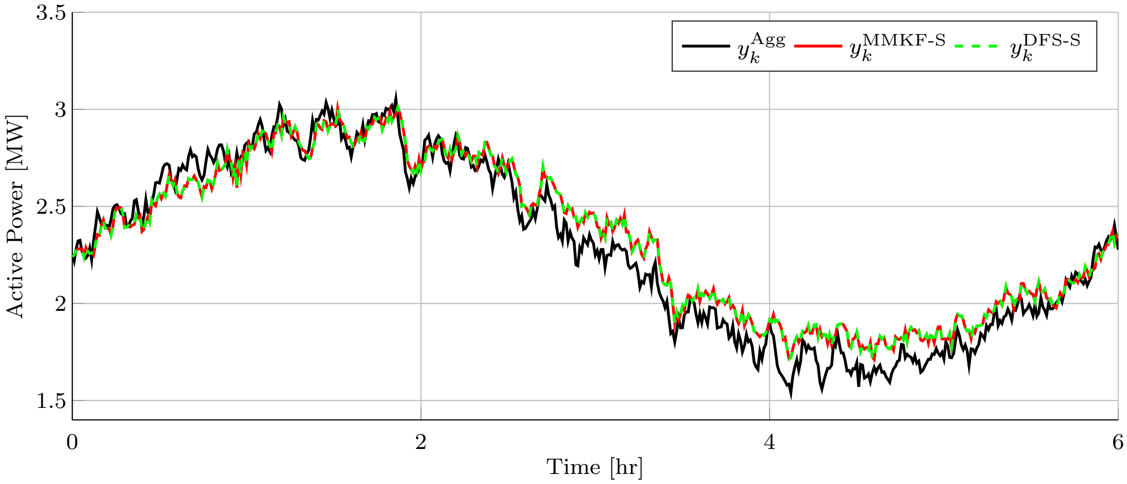


Figure 5.4: Time series of the aggregate AC demand, denoted y_k^{Agg}, versus the MMKF-S and DFS-S estimates, denoted $y_k^{\text{MMKF-S}}$ and $y_k^{\text{DFS-S}}$, respectively.

5.6.2 Results

Figure 5.4 presents time series for the MMKF-S and DFS-S. Figure 5.5 presents time series of the MMKF and DFS-M estimates along with the total AC demand; we exclude time series of the other algorithm implementations as they are difficult to distinguish from one another. Figure 5.6 presents time series of the Kalman filter estimates obtained using each model along with the aggregate AC demand. Table 5.1 summarizes the RMS error in kW of the aggregate AC demand estimates for each algorithm implementation, and Fig. 5.7 presents time series of the weights for various algorithm implementations. Note that in Fig. 5.7 we exclude weight time series for MMKF-W as results could not be computed due to numerical issues, and we exclude weight time series for MMKF-E as they are similar to those of DFS-E. The numerical issues with MMKF-W arise due to the coefficient in front of the exponential in (5.6), which results in values that are approximately zero when computing a windowed weight, i.e., using an update that is similar to (5.23).

From Fig. 5.4, we can see that the estimates for MMKF-S and DFS-S are the same. From Table 5.1, we can see that the RMS estimation errors of MMKF, MMKF-S, and DFS-S are the same, which empirically validates the equivalence established via scaling the output equations in Section 5.4.2 and demonstrates that, in this case, the MMKF scaling achieves approximately the same result as without scaling.

From Table 5.1, we can see that the MMKF performance can be improved with heuristics. In general, the DFS implementations slightly outperform the comparable MMKF implementations. It is unsurprising that the results are similar due to the

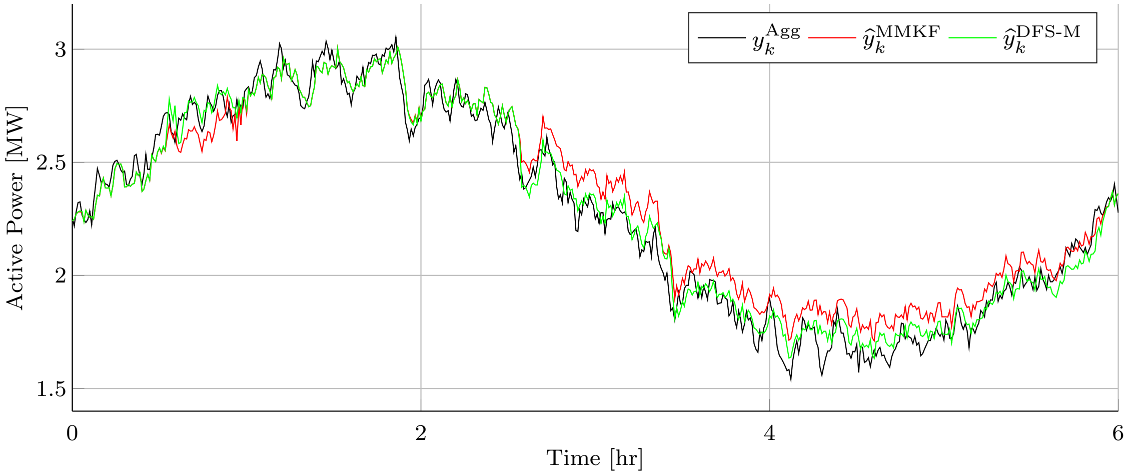


Figure 5.5: Time series of the aggregate AC demand, denoted y_k^{Agg}, versus the MMKF and DFS-M estimates, denoted $\widehat{y}_k^{\text{MMKF}}$ and $\widehat{y}_k^{\text{DFS-M}}$, respectively.

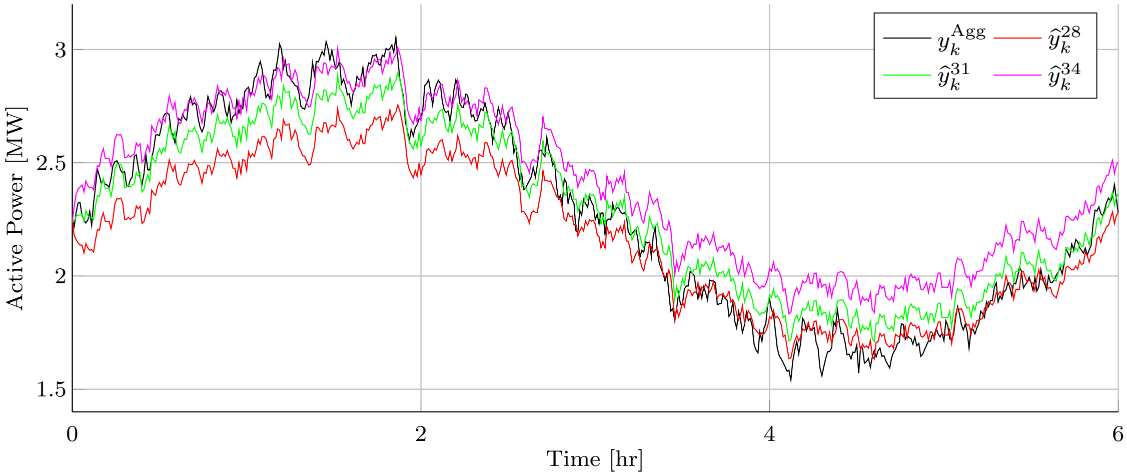
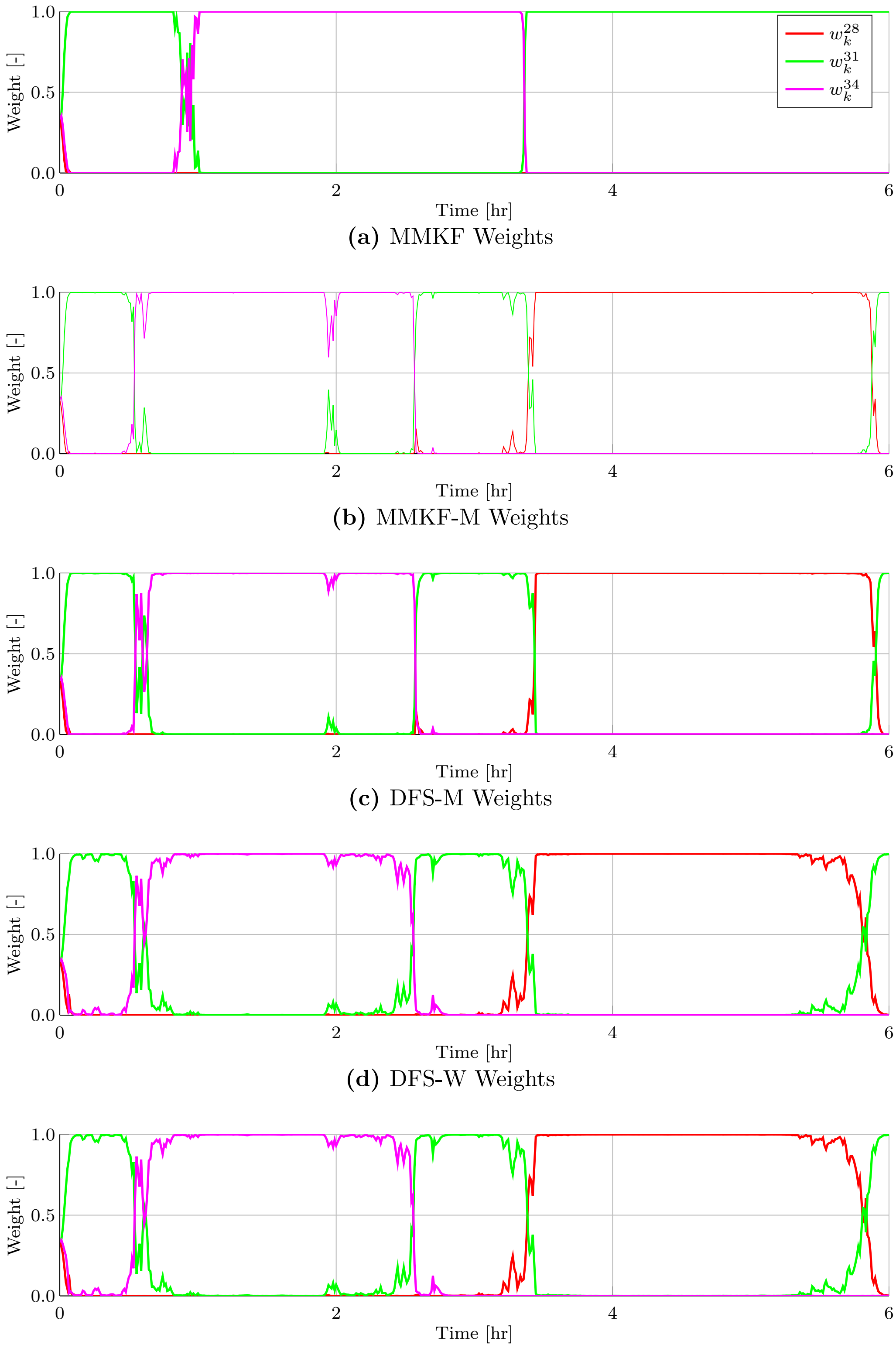


Figure 5.6: Time series of the aggregate AC demand, denoted y_k^{Agg}, versus the individual, underlying Kalman filter estimates, denoted $\widehat{y}_k^{28}$, $\widehat{y}_k^{31}$, and $\widehat{y}_k^{34}$ for models m^{28}, m^{31}, and m^{34}, respectively.



(a) MMKF Weights

(b) MMKF-M Weights

(c) DFS-M Weights

(d) DFS-W Weights

(e) DFS-E Weights

Figure 5.7: Time series of the weights for MMKF, MMKF-M, DFS-M, DFS-W, and DFS-E where w_k^{28}, w_k^{31}, and w_k^{34} denote the weights for m^{28}, m^{31}, and m^{34}, respectively

similarities in the MMKF and DFS algorithms. Tuning the covariance matrices within the underlying Kalman filters may improve the performance of all implementations, but this improvement would be identical across all implementations as they all use the same underlying estimates. The improvement in DFS is due to the parameter η^{r}, which allows the algorithm to generate more dynamic weights by setting η^{r} to larger values. This is not possible in a MMKF.

Comparing the various time series of the weights to the accuracy of the underlying Kalman filter estimates illuminate the differences in algorithm performance. Specifically, the MMKF weights presented in Fig. 5.7a show that the MMKF does not ever weight m^{28} heavily, even though the model is accurate over the final two hours of the simulation. This is because m^{28} was inaccurate over the early portion of the simulation, resulting in a low likelihood and a low weight, and it was unable to regain weight once it became accurate. Including a minimum weight into the MMKF overcomes this issue, as can be seen in Fig. 5.7b, which shows the weight time series for MMKF-M. In contrast, the DFS algorithm weights m^{28} heavily in the final two hours of the simulation, as can be see in Fig. 5.7c.

Another characteristic of the DFS-M, MMKF, and MMKF-M weights are that they become smoother as the simulation progresses. This is because the weights sum the losses (related to the output estimation errors) as the simulation progresses, and the weights become more stagnant as the losses accrue. Alternatively, the behavior of the DFS-W and DFS-E weights in Fig. 5.7d and Fig. 5.7e, respectively, are more consistent throughout the simulation. The MMKF-E weights behave similar to those of DFS-E. This is because the recent losses have larger influence on the weights. As a result, the weights are able to react to the models' recent performance. Incorporating a sliding window or exponential decay into the weight function performs a similar function, which results in similar weights. This can be seen in that DFS-W and DFS-E weights are almost identical and result in very similar RMS error values.

5.7 Chapter V Conlusions

In this chapter, we showed that Dynamic Mirror Descent (DMD), which is used within the DFS algorithm, can be constructed to be equivalent to a discrete-time Kalman filter through proper choice of user-defined functions and parameters. We showed that DFS can produce updates that are identical to that of a MMKF. We also showed that DFS can be modified to incorporate heuristics that are commonly used within a MMKF. We applied various implementations of DFS and a MMKF to a demand response simulation. This simulation scenario empirically validated the scaling that is used to produce identical updates between DFS and a MMKF. We showed that including a minimum weight threshold improves the performance of DFS and a MMKF. We also showed that including exponential decay or a sliding window in the DFS or MMKF weight update allows more consistent, responsive behavior in the weights. In addition, the minimum weight, exponential decay, and sliding window versions of the DFS and MMKF algorithms effectively estimate the aggregate AC demand within the demand response simulation.

CHAPTER VI

Real-Time Energy Disaggregation of a Distribution Feeder's Demand Using Online Learning [1]

Distributed energy resources (DERs) such as demand-responsive, electric loads and residential solar generation are becoming more common within electricity distribution networks [98, 99]. Sensing infrastructure, such as household smart meters, are also becoming more common [100]. However, distribution system operators still often lack an accurate real-time picture of overall DER characteristics such as i) the total power consumption of the air conditioners connected to a distribution feeder, or ii) the total power production of all solar panels installed on a distribution feeder.

Knowing overall DERs characteristics in real-time can help system operators, utilities, and third-party companies (such as energy efficiency and demand response providers) improve power system reliability, economic efficiency, and environmental impact. For example, 1) a system operator can better determine balancing reserve requirements by knowing the real-time production of intermittent distributed generation; 2) a utility can better plan demand response actions by knowing the weather forecast and the real-time portion of weather-dependent loads (e.g., air conditioners, heaters, dehumidifiers); 3) a demand response provider offering ancillary services to the system operator can better determine its bid capacity by knowing the real-time consumption of demand responsive loads; and 4) a demand response provider can use the real-time consumption of demand responsive loads as the feedback control signal within a load coordination algorithm.

Perfect real-time knowledge of DER characteristics requires a sensor at each of the large number (e.g., thousands) of spatially distributed devices and a communication infrastructure capable of reliably transmitting the data at the necessary frequency

[1] The work presented in this chapter was originally published in [86, 96], and preliminary work was published in [97].

(e.g., every few seconds). Rather than installing additional, costly metering and communication infrastructure, this chapter shows that it is possible to estimate real-time DER characteristics using existing sensing capabilities and some knowledge of the underlying system. Specifically, we show how to separate measurements of the net demand served by a distribution feeder into its components in real-time, using knowledge of the physical processes driving load/generation. We refer to this task as *feeder-level energy disaggregation*.

In this work, the feeder-level energy disaggregation problem framework is specified, Dynamic Fixed Share (DFS) [90] is applied to separate the active power demand served by a feeder into two components: the active power demand of a population of residential air conditioners and the active power demand of all other loads connected to the feeder. DFS incorporates dynamical system models of arbitrary forms, blending aspects of machine learning and state estimation. The contributions of this work are the following:

- Frame the feeder-level energy disaggregation problem
- Adapt DFS to the feeder-level energy disaggregation problem
- Develop a variation of DFS that allows it to include models with different underlying states
- Demonstrate the performance of DFS via a realistic, data-driven case study
- Compare the performance of DFS to that of a set of Kalman filters
- Demonstrate the influence of including model prediction error statistics on the performance of DFS.

Section 6.1 compares our problem and approach to related problems/work. Section 6.2 defines the problem framework. Section 6.3 describes the data used to construct the underlying system, and Section 6.4 describes the models used within the algorithm. Section 6.5 summarizes the DFS algorithm, and it summarizes the implementations of DFS for the feeder-level energy disaggregation problem. Section 6.6 constructs case studies and summarizes their results. Section 6.7 presents the conclusions of the chapter.

6.1 Comparison to related problems and work

The feeder-level energy disaggregation problem combines aspects of building-level energy disaggregation and load forecasting. Building-level energy disaggregation, also referred to as nonintrusive load monitoring [101], separates building-level demand measurements into estimates of the demand of individual or small groups of devices [49]. Building-level energy disaggregation algorithms typically use an aggregate signal that is sampled at high frequencies (e.g., 10 KHz to over 1 MHz) and composed of 10-100 component loads. Disaggregation algorithms use data sampled at frequencies ranging from over 1 MHz to 0.3 mHz (i.e., hourly interval data) where higher-frequency data allows separation of more devices [49]. The algorithms generally leverage assumptions stemming from the relatively small number of underlying loads (e.g., a single device turns on or off per time-step [102] or that step changes can be seen in the aggregate signal [103]), and the problem is not usually solved online. Both unsupervised and supervised learning approaches have been proposed, with the latter often using models developed with submetering data.

Load forecasting predicts the total future demand within a given area over time horizons ranging from hours to years [104]. Whereas energy disaggregation typically deals with small load aggregations, load forecasting typically deals with large aggregations, e.g., thousands to millions of loads. For example, forecasting the load served by a distribution transformer is considered a "small" forecasting problem [104]. Very short term load forecasting, corresponding to intraday forecasts, generally uses 15 min to one hour interval data [104, 105]. Smart meter data enables offline development of detailed load models [106], which may be used online for operational decisions [107], e.g., for predicting the curtailable load [106]. However, load forecasting is typically done offline and used for planning.

In contrast to building-level energy disaggregation, feeder-level energy disaggregation involves disaggregating the demand of a large number of loads, e.g., thousands, into a small number of source signals, e.g., two. In contrast to load forecasting, feeder-level energy disaggregation estimates *portions* of the total demand and assumes real-time demand measurements, e.g., taken by SCADA systems at distribution substations, are available on timescales of seconds to minutes. This corresponds to relatively fast sampling for load forecasting and relatively slow sampling for building-level energy disaggregation. In contrast to both building-level energy disaggregation and load forecasting, feeder-level energy disaggregation is done online. However, much like load forecasting and some building-level energy disaggregation approaches, we assume

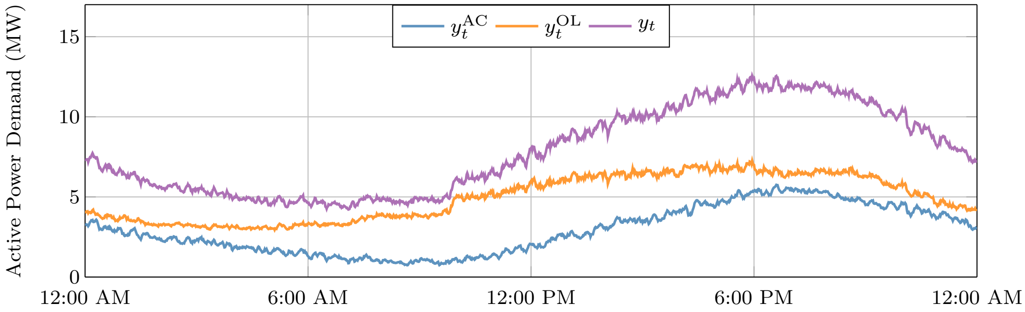


Figure 6.1: Example time series of y_t and its components y_t^{OL} and y_t^{AC}.

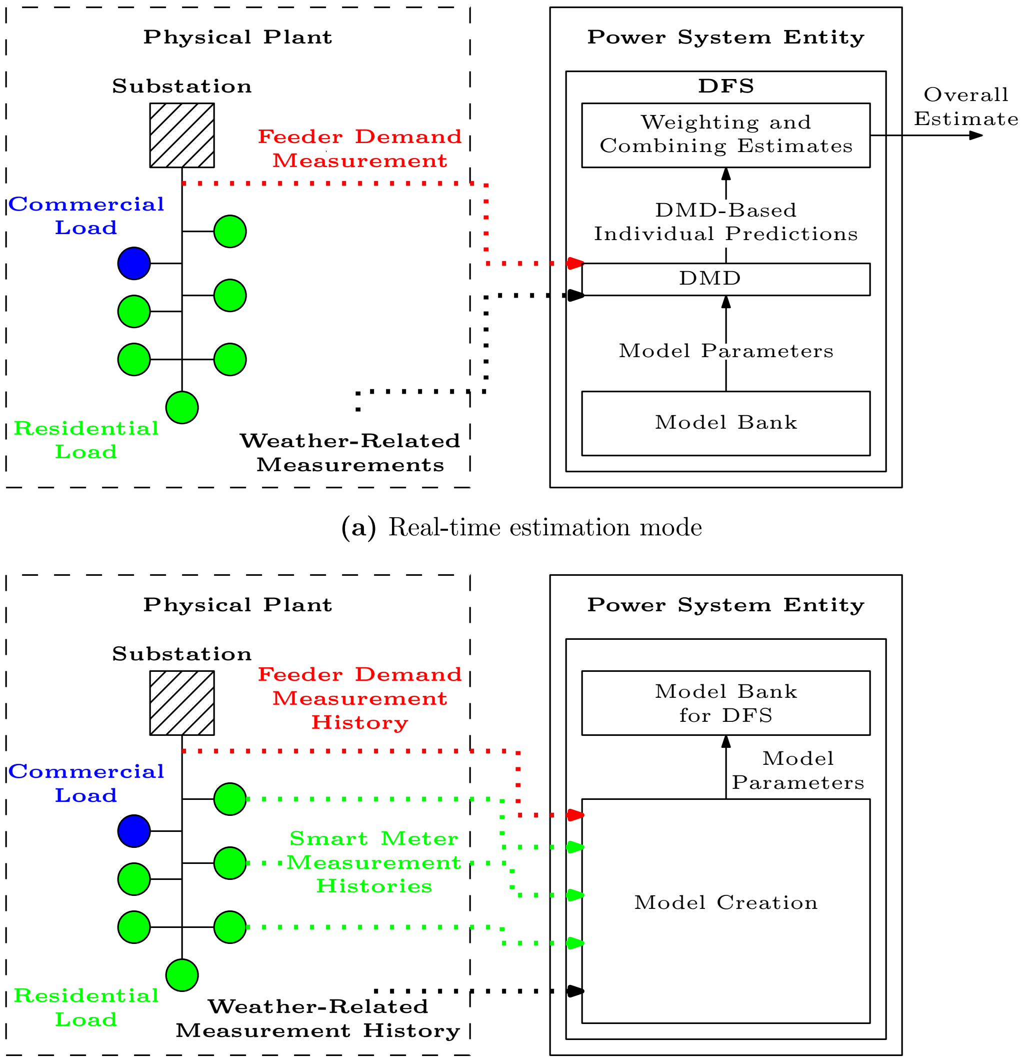
detailed historical load data are available and used offline to construct models.

Machine learning algorithms have been proposed to address a number of problems in power systems including security assessment, forecasting, and optimal operation [108]. A variety of machine learning techniques have been used to forecast load, renewable generation, and prices [109–113]. References [45, 114–117] apply learning approaches to demand response. However, to the best of our knowledge, this is the first paper to pose and solve the feeder-level energy disaggregation problem, or to apply the approach in [90] to a power systems problem.

6.2 Problem Framework

We assume that a power system entity (e.g., a system operator, utility, or third-party company) has access to real-time measurements of the electricity demand served by a distribution feeder. The power system entity is interested in separating these measurements into two components in real-time, i.e., at each time-step. The first component is the power demand of a population of residential air conditioners served by the feeder, referred to as the "AC demand." Air conditioners generally draw power periodically to maintain a building's indoor temperature within a range centered at a user-defined temperature set-point. The AC demand varies in time due to each air conditioner's power cycling, weather-related influences, and building occupant influences. The second component is the power demand of the other loads on the feeder, referred to as the "OL demand," which we assume includes both residential and commercial loads. Figure 6.1 displays example time series for the measured total demand y_t, the AC demand y_t^{AC}, and the OL demand y_t^{OL} over a day. We measure y_t at each time-step and try to estimate y_t^{AC} and y_t^{OL} at each time-step as each measurement arrives.

The power system entity has two distinct modes of operation. The first is the real-



(a) Real-time estimation mode

(b) Offline model generation mode

Figure 6.2: Problem framework: real-time and offline modes.

time estimation mode depicted in Fig. 6.2a. The second is the offline model generation mode, depicted in Fig. 6.2b. During real-time operation, we assume that the power system entity has access to active power measurements corresponding to the demand served by the distribution feeder as well as weather-related measurements. The power measurements are time-averaged active power demands over one minute intervals, and they are the sum of the AC and OL demand. The weather-related measurements could include, for example, temperature and humidity, and can be obtained from existing weather sensors; load-specific weather monitoring is not required.

Model generation occurs offline using historical smart meter, feeder, and weather data. To apply DFS to feeder-level energy disaggregation we assume real-time measurements of the demand components are unavailable, but models of the components are available. These models could be created using a variety of techniques (e.g., via system identification using historical measurements obtained from the same system or a different system, or using analytical methods and parameters from the literature). In this work, we assume that smart meters are installed at all houses, and they enable the collection of household-level demand measurements at one minute intervals. The smart meters' communication limitations [49] make real-time communication of this information infeasible, and so we assume that it is only available offline for prior days. Because we do not need real-time, device-level demand measurements, we assume that historical device-level demand estimates can be obtained offline from the historical household-level measurements either by applying non-intrusive load monitoring (NILM) algorithms or by using information from communicating or advanced thermostats, which are becoming more common within residences. These thermostats can measure and record the on/off mode of a residence's AC unit and measurement histories can be used to estimate the power draw of these devices. The resulting device-level demand estimates may not be exact, but they are accurate enough to be used within the computation of model parameters. As a result, we use historical, device-level measurements to construct the AC demand models. We also assume that the power system entity has access to historical feeder and weather data. Once the models are formed, they are used along with the real-time measurements to estimate the AC and OL demand.

An online learning algorithm, Dynamic Mirror Descent (DMD) [90], uses a single model to generate predictions of the total demand, a loss function to penalize errors between the predicted and measured total demand, and a convex optimization formulation to adjust this prediction based on the measured total demand. Dynamic Fixed Share (DFS) [90], uses DMD within the Fixed Share Algorithm [93] to include predictions from a bank of models. Specifically, DFS applies DMD separately to each model and uses a weighting algorithm to associate a weight with each model's adjusted prediction before combining the predictions into an overall estimate. In DFS, these models are weighted based on their prediction accuracy – better prediction-measurement matching leads to larger weighting and more influence in the overall prediction.

Rather than predicting the AC demand using a single load forecast, the proposed approach has two main advantages. First, in DMD, the AC and OL demand predictions are adjusted in real-time based on the real-time, realized feeder demand. This feedback improves future predictions; in contrast, load forecasting is open-loop. It is necessary to predict both the AC and OL demand since only the total demand is measured. If only the AC demand is predicted, the prediction cannot be adjusted in real-time because measurements of the realized AC demand are not available in real-time. Second, the DFS algorithm can incorporate a number of AC demand predictions into an overall AC demand prediction. Predictions associated with prediction methods that have performed well recently are weighted more heavily and the weights evolve over time so different predictors will be preferred at different times. The algorithm implementation is described in detail in Section 6.5.2, but first we describe the construction of the underlying physical system, i.e., the plant, used within the case studies and the models used within the algorithm.

6.3 Construction of Plant

In this section, we detail the methods used to form the AC and OL demand time series and the associated weather time series over one day. These time series incorporate data from real households, the devices within those households, commercial buildings, and nearby weather stations. The data for individual, residential air conditioners are summed to form the AC demand, the data for household non-AC devices are summed to form the residential OL demand, and the data for commercial building demand signals are summed and scaled to form the commercial, OL demand signal. Lastly, the outdoor temperature data consists of real data from nearby weather stations, and the data is interpolated to make it applicable on the time-steps used within the problem scenario. These time series are then used as the plant, i.e., the underlying physical system or the ground-truth signals. The time series for a day consist of n^{steps} one-minute time-steps with $t = 0$ at 12:00 AM. Because we were unable to find sufficient data from a single location/day, we use demand and weather data from a variety of sources.

We use feeder model R5-25.00-1 from GridLAB-D's feeder taxonomy [118] to set the average residential and commercial demand on the feeder to 5.8 MW and 2.1 MW, respectively. Ignoring network losses (which, if included, would be treated as part of the OL demand), the total feeder demand measurements are the sum of the AC and OL demand, i.e., $y_t = y_t^{\text{AC}} + y_t^{\text{OL}}$, where y_t^{OL} is the sum of the other residential demand and the commercial demand $y_t^{\text{OL}} = y_t^{\text{OL,res}} + y_t^{\text{OL,com}}$.

Both y_t^{AC} and $y_t^{\text{OL,res}}$ are constructed using residential data from the Pecan Street Dataport [85]. The data consists of historical, one-minute interval, household- and device-level demand measurements for a set of single family homes in Texas. Daily household demand signals were randomly drawn with replacement and added together until the total residential signal's mean matched that of the feeder model, resulting in $2,499$ total houses. To construct the AC demand signal, we summed the demand of each household's primary air conditioner and air blower unit. Note that some houses have no/multiple air conditioner and air blower units. We assume that only one unit per household contributes to the AC demand, resulting in $2,269$ units. The remaining demand is the residential OL demand.

The commercial data consists of 4-second interval, whole-building demand measurements from two buildings in California, a municipal building and a big box retail store. We summed the demand of the two buildings, and then scaled the sum by 2.61 to match the average commercial demand of the feeder model. We also down-sampled the data to one minute intervals by averaging the values over each minute.

The plant's weather data is constructed from data obtained from the Pecan Street Dataport [85] and the National Climatic Data Center [119]. The Pecan Street weather data corresponds to the residential demand. It consists of the outdoor air temperature for Austin, TX, and it is sampled at one hour intervals. We linearly interpolate the data down to one minute intervals. The NOAA weather data corresponds to the commercial demand. It consists of outdoor temperature data from the Concord, CA weather station, sampled at one hour intervals. Again, we linearly interpolate the data down to one minute intervals. All weather data was taken from the same day as the demand data.

6.4 System Models

In this section, we describe the models used to generate predictions of the AC and OL demands. These models are generated offline, using historical data, and then used within the online learning algorithm implementations detailed in Section 6.5.2. The historical demand signals were constructed in the same manner as described in Section 6.3, using the same combination of houses as used to construct the plant signals. The OL demand is modeled using two different linear regression methods as detailed in Section 6.4.1, and the AC demand is modeled using several linear dynamic systems as well as a linear regression method as detailed in Section 6.4.2. Note that other models may prove to be more accurate on average than the models that are used within this work. However, the intended objective within this work is to use an array of models (including some that are known to be overly simple or less accurate) to investigate the performance of DFS on the feeder-level energy disaggregation problem.

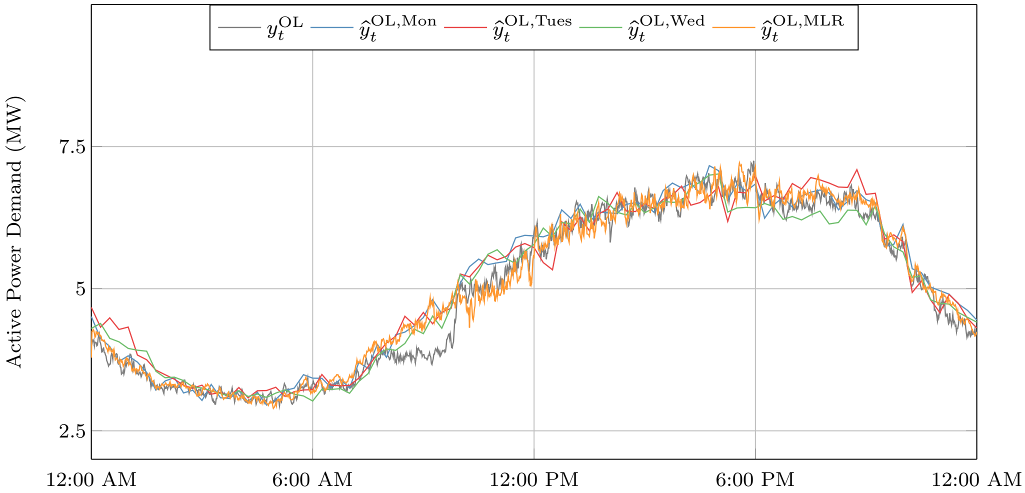


Figure 6.3: Example OL demand and several OL demand model predictions.

6.4.1 OL Demand Models

We use two types of regression models to predict the OL demand: time-of-day (TOD) regression models and a multiple linear regression (MLR) model. Figure 6.3 displays a day of data for y_t^{OL}, several TOD regression model predictions, e.g., $\widehat{y}_t^{\mathrm{OL,Mon}}$, and the MLR model prediction $\widehat{y}_t^{\mathrm{OL,MLR}}$. We next describe the construction of these models.

6.4.1.1 TOD Regression Models

The TOD regression model is a lookup table where an OL demand prediction is generated for each minute of the day based on the OL demand of a single day in the past

$$\widehat{y}_t^{\mathrm{OL,TOD}} = \alpha_k^{\mathrm{OL,TOD}} = \alpha^{\mathrm{OL,TOD}}\ x_t^{\mathrm{OL,TOD}}. \tag{6.1}$$

Whereas t indexes overall time-steps, k indexes the time of day in minutes, i.e., $k = 0$ for 12:00 AM and $k = 60$ for 1:00 AM. The scalar α_k^{OL} corresponds to the predicted OL demand value for time-of-day k, $\alpha^{\mathrm{OL,TOD}}$ is a row vector containing all α_k^{OL} values, and $x_t^{\mathrm{OL,TOD}}$ is a column vector that selects the appropriate $\alpha^{\mathrm{OL,TOD}}$ based on the corresponding time of day for t. We generate $\alpha^{\mathrm{OL,TOD}}$ by smoothing the OL demand signal of a previous day using a piecewise linear and continuous, least-squares fit. Each linear segment corresponds to a 15 minute interval of the historical data. We generate one TOD regression model for each weekday, and the models are denoted $\Phi^{\mathrm{OL,Mon}}$, $\Phi^{\mathrm{OL,Tues}}$, $\Phi^{\mathrm{OL,Wed}}$, $\Phi^{\mathrm{OL,Thurs}}$, and $\Phi^{\mathrm{OL,Fri}}$. Their corresponding predictions are $\widehat{y}_t^{\mathrm{OL,Mon}}$, $\widehat{y}_t^{\mathrm{OL,Tues}}$, $\widehat{y}_t^{\mathrm{OL,Wed}}$, $\widehat{y}_t^{\mathrm{OL,Thurs}}$, and $\widehat{y}_t^{\mathrm{OL,Fri}}$, respectively.

6.4.1.2 MLR Model

The MLR model of the OL demand is denoted $\Phi^{\mathrm{OL,MLR}}$, and it uses input features that include calendar-based variables, e.g., the day of the week, as well as weather-based variables, e.g., the outdoor temperature, to generate an OL demand prediction. We split the MLR model into two distinct components: one model for the commercial demand and one model for the residential OL demand since the underlying data corresponds to different geographic areas and time periods. The overall MLR model of the OL demand is then the sum of the predicted residential OL demand $\widehat{y}_t^{\mathrm{OL,res}}$ and the predicted commercial demand $\widehat{y}_t^{\mathrm{OL,com}}$, i.e.,

$$\begin{aligned} \widehat{y}_t^{\mathrm{OL,MLR}} &= \widehat{y}_t^{\mathrm{OL,res}} + \widehat{y}_t^{\mathrm{OL,com}} && (6.2) \\ &= \beta^{\mathrm{OL,res}}\, x_t^{\mathrm{OL,res}} + \gamma^{\mathrm{OL,com}}\, x_t^{\mathrm{OL,com}}, && (6.3) \end{aligned}$$

where the row vectors $\beta^{\mathrm{OL,res}}$ and $\gamma^{\mathrm{OL,com}}$ are regression parameters for the residential OL demand and the commercial demand, respectively. The column vectors $x_t^{\mathrm{OL,res}}$ and $x_t^{\mathrm{OL,com}}$ are the corresponding input features.

The MLR model for the residential OL demand uses input features $x_t^{\mathrm{OL,res}} = \left[(x_t^{\mathrm{TOW}})^{\mathrm{T}} \quad T_t^{\mathrm{TX}} \quad y_{t-1}\right]^{\mathrm{T}}$ where x_t^{TOW} is an indicator vector for the time of week in minutes, T_t^{TX} is the outdoor temperature for Austin, TX, and y_{t-1} is the measured total demand of the previous time-step. The commercial regression model corresponds to "Baseline Method 1" from [120]; it uses input features $x_t^{\mathrm{OL,com}} = \left[(x_t^{\mathrm{TOW}})^{\mathrm{T}} \quad T_t^{\mathrm{CA}} \cdot (x_t^{\mathrm{TOW}})^{\mathrm{T}}\right]^{\mathrm{T}}$ where T_t^{CA} is the outdoor temperature for Concord, CA and $T_t^{\mathrm{CA}} \cdot (x_t^{\mathrm{TOW}})$ is a vector that associates the temperature to the corresponding time of week.

6.4.2 AC Demand Models

We use three types of models to predict the AC demand: a MLR model, linear time invariant (LTI) system models, and linear time varying (LTV) system models. Figure 6.4 displays y_t^{AC} for a simulated day, several LTV model predictions, e.g., $\widehat{y}_t^{\mathrm{AC,LTV1}}$, and the MLR regression model prediction $\widehat{y}_t^{\mathrm{AC,MLR}}$. We next describe the construction of these models.

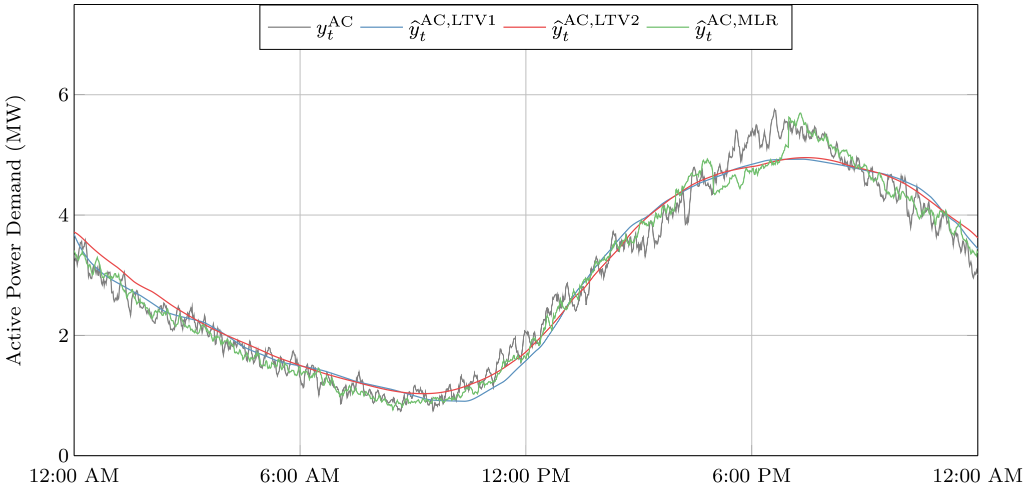


Figure 6.4: Examples AC demand and several AC demand model predictions.

6.4.2.1 MLR Model

The MLR model of the AC demand, denoted $\Phi^{\mathrm{AC,MLR}}$, is similar to the MLR model in Section 6.4.1.2 with different input features

$$x_t^{\mathrm{AC,MLR}} = \left[(x_t^{\mathrm{TOW}})^{\mathrm{T}} \quad T_{t-\tau^{\mathrm{l}}}^{\mathrm{TX}} \quad (T_{t-\tau^{\mathrm{l}}}^{\mathrm{TX}})^2 \quad (T_{t-\tau^{\mathrm{l}}}^{\mathrm{TX}})^3 \quad (T_{t-\tau^{\mathrm{l}}}^{\mathrm{TX}})^4 \right]^{\mathrm{T}},$$

where $T_{t-\tau^{\mathrm{l}}}^{\mathrm{TX}}$ is the temperature in Austin, TX from τ^{l} time-steps ago and τ^{l} is the time lag that maximizes the cross correlation between the historical AC demand signal and temperature signal (119 minutes for our plant).

6.4.2.2 LTI Models

We construct a set of LTI models $\mathcal{M}^{\mathrm{LTI}}$ that are similar to the aggregate model detailed in Section 2.2.2. As in [84], each model within the set captures the aggregate behavior of the population of air conditioners at outdoor temperature T^m and has the form

$$\widehat{x}_{t+1}^{\mathrm{LTI},m} = A^{\mathrm{LTI},m} \, \widehat{x}_t^{\mathrm{LTI},m} \tag{6.4}$$

$$\widehat{y}_t^{\mathrm{AC,LTI},m} = C^{\mathrm{LTI},m} \, \widehat{x}_t^{\mathrm{LTI},m}, \tag{6.5}$$

with $m \in \mathcal{M}^{\mathrm{LTI}} = \{1, \ldots, N^{\mathrm{LTI}}\}$. The state vector $\widehat{x}_t^{\mathrm{LTI},m} \in \mathbb{R}^{N^{\mathrm{x}} \times 1}$ consists of the portion of the air conditioners within each of N^{x} discrete states. In this chapter, we use one state to represent the portion of air conditioners that are drawing power and another to represent those that are not, i.e., $N^{\mathrm{x}} = 2$. The state transition matrix,

$A^{\mathrm{LTI},m} \in \mathbb{R}^{N^{\mathrm{x}} \times N^{\mathrm{x}}}$, is a transposed Markov transition matrix. Its entries capture the probabilities that air conditioners maintain their current state or transition to the other state during the time-step. The output matrix $C^{\mathrm{LTI},m}$ estimates the AC demand $\widehat{y}_t^{\mathrm{AC,LTI},m}$ from the portion of air conditioners that are drawing power, i.e., $C^{\mathrm{LTI},m} = N^{\mathrm{ac}}\overline{P}^m \begin{bmatrix} 0 & 1 \end{bmatrix}$, where $\overline{P}^m$ is a parameter approximating of the average power draw of air conditioners drawing power and N^{ac} is the number of air conditioners, which we assume is known.

To identify $A^{\mathrm{LTI},m}$ and $C^{\mathrm{LTI},m}$ for all m, we first define a set of N^{LTI} evenly spaced temperatures $\mathcal{T}^{\mathrm{temps}} = \left\{T^{\mathrm{min}}, \ldots, T^{\mathrm{max}}\right\}$ and denote the m-th temperature of the set as T^m. The difference between successive temperatures T^m and T^{m+1} is ΔT. Matrices $A^{\mathrm{LTI},m}$ and $C^{\mathrm{LTI},m}$ are constructed using power demand signals from each air conditioner corresponding to periods when $T^m - \frac{\Delta T}{2} \leqslant T^{\mathrm{TX}}_{t-\tau^1} < T^m + \frac{\Delta T}{2}$. Some heuristics were used to exclude anomalous high or low power demand measurements. Parameter $\overline{P}^m$ is set as the average power draw of air conditioners that are drawing power. The four entries of $A^{\mathrm{LTI},m}$ were determined by checking whether an air conditioner 1) started drawing power, 2) stopped drawing power, 3) continued to draw power, or 4) continued to not draw power during each time-step. The occurrences for each case were counted for every air conditioner at every time-step and the totals were placed into their respective entries in $A^{\mathrm{LTI},m}$, and then each column was normalized so that the sum of the column entries was 1. In our case studies, we construct an LTI model for each integer temperature in the set $\{74, \ldots, 99\}$ °F. If the outdoor temperature lies outside of this range, we use the model corresponding to the closest temperature.

6.4.2.3 LTV Models

We use two LTV models, where the models both have the general form of that in Section 4.2.1, but the method of computing the time-varying matrices are different for each model. The first $\Phi^{\mathrm{AC,LTV1}}$ uses the delayed temperature and has the form

$$\widehat{x}_{t+1}^{\mathrm{LTV1}} = A_t^{\mathrm{LTV1}}\, \widehat{x}_t^{\mathrm{LTV1}} \tag{6.6}$$

$$\widehat{y}_t^{\mathrm{AC,LTV1}} = C_t^{\mathrm{LTV1}}\, \widehat{x}_t^{\mathrm{LTV1}}, \tag{6.7}$$

where A_t^{LTV1} and C_t^{LTV1} are generated by linearly interpolating the matrix entries based on $T^{\mathrm{TX}}_{t-\tau^1}$. The second $\Phi^{\mathrm{AC,LTV2}}$ uses a moving average of the past temperature over τ^{w} time-steps to generate the prediction $\widehat{y}_t^{\mathrm{AC,LTV2}}$. We chose τ^{w} to be the value that maximizes the cross correlation between the historical moving average temperature and the historical AC demand signal (270 min for our plant). When evaluating either LTV model, if the temperature lies outside of the range used to generate the model, we extrapolate using the difference between the nearest two models.

6.5 Online Learning Algorithm

In this section, we first summarize the DFS algorithm developed in [90] and then describe two algorithm implementations, one inspired by DFS and one a direct implementation of it. DFS incorporates DMD, also developed in [90], into the Fixed Share algorithm originally developed in [93]. The Fixed Share algorithm combines a set of predictions that are generated by independent experts, e.g., models, into an estimate of the system parameter using the experts' historical accuracy with respect to observations of the system. DMD extends the traditional online learning framework by incorporating dynamic models, enabling the estimation of time-varying system parameters (or states). DFS uses DMD, applied independently to each of the models, as the experts within the Fixed Share algorithm.

6.5.1 The DFS Algorithm

The objective of DFS is to form an estimate $\widehat{\theta}_t \in \Theta$ of the dynamic system parameter $\theta_t \in \Theta$ at each discrete time-step t where $\Theta \subset \mathbb{R}^p$ is a bounded, closed, convex feasible set. The underlying system produces observations, i.e., measurements, $y_t \in \mathcal{Y}$ at each time-step after the prediction has been formed, where $\mathcal{Y} \subset \mathbb{R}^q$ is the domain of the measurements. From a control systems perspective, this is equivalent to a state estimation problem where θ_t is the system state.

DFS uses a set of N^{mdl} models defined as $\mathcal{M}^{\mathrm{mdl}} = \{1, \ldots, N^{\mathrm{mdl}}\}$ to generate the estimate $\widehat{\theta}_t$. To do this, DFS applies the DMD algorithm to each model, forming predictions $\widehat{\theta}_t^m$ for each $m \in \mathcal{M}^{\mathrm{mdl}}$. DMD is executed in two steps (similar to a discrete-time Kalman filter): 1) an observation-based update incorporates the new measurement into the parameter prediction, and 2) a model-based update advances the parameter prediction to the next time-step. DFS then uses the Fixed Share algorithm to form the estimate $\widehat{\theta}_t$ as a weighted combination of the individual model's DMD-based predictions. A weighting algorithm computes the weights based on each model's historical accuracy with respect to the observations y_t. Models that perform poorly have less influence on the overall estimate. The DFS algorithm is [90]

$$\widetilde{\theta}_t^m = \arg\min_{\theta \in \Theta} \eta^{\mathrm{s}} \left\langle \nabla \ell_t(\widehat{\theta}_t^m),\ \theta \right\rangle + D\left(\theta \| \widehat{\theta}_t^m\right) \tag{6.8}$$

$$\widehat{\theta}_{t+1}^m = \Phi^m(\widetilde{\theta}_t^m) \tag{6.9}$$

$$w_{t+1}^m = \frac{\lambda}{N^{\mathrm{mdl}}} + (1-\lambda) \frac{w_t^m \exp\left(-\eta^r \, \ell_t\left(\widehat{\theta}_t^m\right)\right)}{\sum_{j=1}^{N^{\mathrm{mdl}}} w_t^j \exp\left(-\eta^r \, \ell_t\left(\widehat{\theta}_t^j\right)\right)} \tag{6.10}$$

for each $m \in \mathcal{M}^{\mathrm{mdl}}$, and

$$\widehat{\theta}_{t+1} = \sum_{m \in \mathcal{M}^{\mathrm{mdl}}} w_{t+1}^m \, \widehat{\theta}_{t+1}^m, \tag{6.11}$$

where each term is defined below. DMD is applied to each model in (6.8) and (6.9) to form the expert predictions, where (6.8) is a convex program that constructs the measurement-based update to the previous prediction and (6.9) is the model-based advancement of the adjusted prediction. The Fixed Share algorithm consists of (6.10) and (6.11), where (6.10) computes the weights and (6.11) computes the estimate as a weighted combination of the individual experts' estimates. We note that the Fixed Share algorithm's updates are independent of the dynamics and only use the experts' predictions and their resulting losses.

In (6.8), we minimize over the variable θ, $\eta^{\mathrm{s}} > 0$ is a step-size parameter, and $\langle \cdot, \cdot \rangle$ is the standard dot product. The value $\nabla \ell_t(\widehat{\theta}_t)$ is a subgradient of the convex loss function $\ell_t : \Theta \to \mathbb{R}$, which penalizes the error between the predicted and observed values y_t using a known, possibly time-varying, function $h_t : \Theta \to \mathcal{Y}$ that maps θ_t to an observation, i.e., $y_t = h_t(\theta_t)$, to form predictions of the measurements. An example loss function is $\ell_t(\widehat{\theta}_t) = \|C\widehat{\theta}_t - y_t\|_2^2$ where the matrix C is $h_t(\cdot)$. In (6.9), the function $\Phi^m(\cdot)$ applies model m to advance the adjusted estimate $\widetilde{\theta}_t^m$ in time. Each $\Phi^m(\cdot)$ can have arbitrary form and time-varying parameters. In (6.10), the weight associated with model m at time-step t is w_t^m, $\lambda \in (0, 1)$ determines the amount of weight that is shared amongst models, and η^r influences switching speed. The weight for model m is based on the loss of each model and the total loss of all models. The term $\eta^{\mathrm{s}}\langle \nabla \ell_t(\widehat{\theta}_t),\ \theta \rangle$ captures the alignment of the variable θ with the positive gradient of $\ell_t(\widehat{\theta}_t)$. To minimize this term alone, we would choose θ to be exactly aligned with the negative gradient direction. The term $D(\theta \| \widehat{\theta}_t)$ is a Bregman divergence that penalizes the deviation between the new variable θ and the old variable $\widehat{\theta}_t$. For simplicity, we have excluded regularization within (6.8), which DMD readily incorporates [90].

6.5.2 Algorithm Implementations

We next describe two algorithm implementations to update the expert predictions. First, we describe an implementation that uses the concept of DMD but it is not a direct implementation of DMD. This method treats the models as black boxes and adjusts only their output, i.e., the OL and AC demand predictions, using the measured and predicted total feeder demand. Second, we describe a direct implementation of DMD, which updates the state x_t of the LTI and LTV AC demand models. In the following, the total demand model is $\Phi(\cdot) = \{\Phi^{\mathrm{AC}}(\cdot), \Phi^{\mathrm{OL}}(\cdot)\}$ where $\Phi^{\mathrm{AC}}(\cdot)$ is an AC demand model and $\Phi^{\mathrm{OL}}(\cdot)$ is an OL demand model, with predictions $\widehat{y}_t^{\mathrm{AC}}$ and $\widehat{y}_t^{\mathrm{OL}}$, respectively.

6.5.2.1 Update Method 1

The models used within this chapter have different underlying parameters, dynamic variables, and/or structures, which makes it difficult to define a common θ_t across all of the models used. Therefore, we develop a variation of the DMD algorithm that adjusts the demand predictions directly, rather than applying the updates to quantities influencing the demand predictions. This allows us to include a diverse set of models. Specifically, we modify the DMD formulation to

$$\widehat{\kappa}_{t+1} = \underset{\theta\in\Theta}{\arg\min}\, \eta^{\mathrm{s}} \left\langle \nabla\ell_t(\widehat{\theta}_t),\ \theta \right\rangle + D\left(\theta \| \widehat{\kappa}_t\right) \tag{6.12}$$

$$\breve{\theta}_{t+1} = \Phi(\breve{\theta}_t) \tag{6.13}$$

$$\widehat{\theta}_{t+1} = \breve{\theta}_{t+1} + \widehat{\kappa}_{t+1}. \tag{6.14}$$

The AC and OL demand models generate their predictions independently from one another, and so (6.14) can be rewritten as

$$\widehat{\theta}_{t+1} = \Phi(\breve{\theta}_t) + \widehat{\kappa}_{t+1} = \begin{bmatrix} \Phi^{\mathrm{AC}}(\breve{\theta}_t) \\ \Phi^{\mathrm{OL}}(\breve{\theta}_t) \end{bmatrix} + \widehat{\kappa}_{t+1}. \tag{6.15}$$

The convex program (6.12) is now used to update a value $\widehat{\kappa}_t$ that accumulates the deviation between the predicted and actual measurements. The model-based update (6.13) computes an open-loop prediction $\breve{\theta}_{t+1}$, meaning that the measurements do not influence $\breve{\theta}_{t+1}$. The measurement-based updates and model-based, open-loop predictions are combined in (6.14). In contrast, DMD uses a closed-loop model-based update where the convex program adjusts the parameter estimate to $\widetilde{\theta}_t$, which is used to compute the next parameter estimate $\widehat{\theta}_{t+1}$.

In this method, we define θ_t as the AC and OL demand, i.e., $\theta_t = \begin{bmatrix} y_t^{\mathrm{AC}} & y_t^{\mathrm{OL}} \end{bmatrix}^{\mathrm{T}}$. The mapping from the parameter to the measurement is $h_t(\theta_t) = C_t\theta_t$ where the matrix $C_t = \begin{bmatrix} 1 & 1 \end{bmatrix}$. While the mapping and matrix are time-invariant, they may be time-varying in Section 6.5.2.2, and so we use the more general notation. We choose the loss function as $\ell_t(\widehat{\theta}_t) = \frac{1}{2}\left\|(\widehat{P}_t^{\mathrm{y}})^{-\frac{1}{2}}(C_t\widehat{\theta}_t - y_t)\right\|_2^2$ and the divergence as $D(\theta\|\widehat{\kappa}_t) = \frac{1}{2}\|(\widehat{P}_t)^{-\frac{1}{2}}(\theta - \widehat{\kappa}_t)\|_2^2$ where $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ are symmetric, positive definite, covariance matrices corresponding to the model prediction errors and the measurement prediction errors, respectively. The quantity $G^{-\frac{1}{2}} = U\left(\Sigma^{-\frac{1}{2}}\right)U^{\mathrm{T}}$ denotes a matrix square root of an arbitrary symmetric positive-definite matrix G, where U is orthonormal and Σ a diagonal matrix with positive entries on the diagonal. The square roots of $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ are also symmetric and positive definite [121]. We can then write (6.12) in closed form as

$$\widehat{\kappa}_{t+1} = \widehat{\kappa}_t + \eta^{\mathrm{s}}\widehat{P}_t C_t^T \left(\widehat{P}_t^{\mathrm{y}}\right)^{-1}\left(y_t - C_t\widehat{\theta}_t\right). \tag{6.16}$$

The estimation error covariance $Q_t \in \mathbb{R}^{2\times 2}$ and the measurement noise covariance $R_t \in \mathbb{R}^1$ are used to compute $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$. We set $Q_t = \mathrm{diag}(R_t^{\mathrm{AC}}, R_t^{\mathrm{OL}})$, where $\mathrm{diag}(\cdot)$ forms a diagonal matrix from the scalar arguments. The values $R_t^{\mathrm{AC}} \in \mathbb{R}$ and $R_t^{\mathrm{OL}} \in \mathbb{R}$ correspond to the variances of the AC and OL demand models' prediction errors. We detail several sets of assumptions and methods for computing the parameters R_t, R_t^{AC}, R_t^{OL}, $\widehat{P}_t$, and $\widehat{P}_t^{\mathrm{y}}$ in Section 6.6.2.

6.5.2.2 Update Method 2

This method applies only to dynamic system models with dynamic states, i.e., in this paper the LTI or LTV AC demand models, which have dynamic states x_t. We set $\theta_t = \begin{bmatrix} x_t^{\mathrm{T}} & y_t^{\mathrm{OL}} \end{bmatrix}^{\mathrm{T}}$, where x_t is $\widehat{x}_t^{\mathrm{LTI},m}$ in (6.4), $\widehat{x}_t^{\mathrm{LTV1}}$ in (6.6), or $\widehat{x}_t^{\mathrm{LTV2}}$ in an update equation similar to (6.6). The mapping from the parameter to the measurement is then $C_t = \begin{bmatrix} C_t^{\mathrm{AC}} & 1 \end{bmatrix}$ where C_t^{AC} is the output matrix of the LTI or LTV AC demand model, i.e., $C^{\mathrm{LTI},m}$, C_t^{LTV1}, or C_t^{LTV2}. Defining the system parameter in this way allows us to update the dynamic states of the LTI and LTV AC demand models, rather than just the output as in Update Method 1. The model-based update is then

$$\widehat{\theta}_{t+1} = \begin{bmatrix} \Phi^{\mathrm{AC}}(\widetilde{\theta}_t) \\ \Phi^{\mathrm{OL}}(\breve{\theta}_t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\widehat{\kappa}_{t+1}, \tag{6.17}$$

where we update the AC demand model using the adjusted parameter estimate, as in DMD. Because the OL demand models do not include dynamic states, we continue to update their estimates according to Update Method 1. We again use (6.16) as the measurement-based update where we replace $\widehat{\kappa}_t$ with $\widehat{\theta}_t$.

The estimation error covariance $Q_t \in \mathbb{R}^{3\times 3}$, and the measurement noise covariance $R_t \in \mathbb{R}$ are used to compute $\widehat{P}_t$ and $\widehat{P}^{\mathrm{y}}_t$. The process noise matrix is $Q_t = \mathrm{blkdiag}(Q^{\mathrm{AC}}_t, R^{\mathrm{OL}}_t)$ where $\mathrm{blkdiag}(\cdot)$ constructs a block diagonal matrix from the arguments. The matrix $Q^{\mathrm{AC}}_t \in \mathbb{R}^{2\times 2}$ corresponds to the process noise of the (LTI or LTV) AC demand model. We detail several methods for constructing Q^{AC}_t, R^{OL}_t, $\widehat{P}_t$, and $\widehat{P}^{\mathrm{y}}_t$ in Section 6.6.2.

6.6 Case Studies

In this section, we define the scenarios, describe the benchmark, summarize the parameter settings, and present the results. In [90], performance bounds for DMD and DFS were established in terms of a quantity called regret. Regret is the total (or cumulative) loss of an online learning algorithm's prediction sequence versus that of a comparator sequence, often a best-in-hindsight offline algorithm. In [90], the DMD regret bound uses a comparator that can take on an arbitrary sequence of values from the feasible domain Θ. The DFS regret bound uses a comparator that chooses the best-in-hindsight possible sequence of models chosen from the same model collection used by DFS, where the number of model switches is a predefined number. In lieu of developing formal performance bounds for the given problem scenario, we benchmark the algorithms' performance using Kalman filters, which is described in Section 6.6.3. Future work will investigate regret bounds for our particular problem.

6.6.1 Model Set Scenarios

We define the three sets of models for use within DFS:

1. $\mathcal{M}^{\mathrm{Full}}$, all of the models developed in Section 6.4, i.e., every combination of AC and OL demand models from the AC demand model set

$$\mathcal{M}^{\mathrm{AC,Full}} = \{\mathcal{M}^{\mathrm{LTI}}, \Phi^{\mathrm{AC,MLR}}, \Phi^{\mathrm{AC,LTV1}}, \Phi^{\mathrm{AC,LTV2}}\}$$

and the OL demand model set

$$\mathcal{M}^{\mathrm{OL}} = \{\Phi^{\mathrm{OL,Mon}}, \Phi^{\mathrm{OL,Tues}}, \Phi^{\mathrm{OL,Wed}}, \Phi^{\mathrm{OL,Thurs}}, \Phi^{\mathrm{OL,Fri}}, \Phi^{\mathrm{OL,MLR}}\};$$

2. $\mathcal{M}^{\mathrm{Red}}$, a reduced set that excludes the LTI models, which are not accurate over the course of the day;

3. $\mathcal{M}^{\mathrm{KF}}$, a further reduced set that excludes the MLR AC demand model, which can not be used in a Kalman filter;

Since the Update Method 2 is only applicable to the LTI and LTV AC demand models, case studies using Update Method 2 apply the method to all applicable model combinations and otherwise use Update Method 1.

6.6.2 Error Covariance Scenarios

In this section, we detail three approaches for computing covariance matrices, referred to as "Identity", "Historical", and "Real-Time", that are used within Update Method 1 and 2. The first approach does not explicitly include any model prediction error statistics into the measurement-based updates. The second approach uses historical data from the week preceding the first testing day, where testing days are detailed in Section 6.6.4, to compute covariance matrices. The third approach uses an unrealistic assumption, i.e., that the total, AC, and OL demand are measured at each time-step and used to compute the exact covariance at each time-step. The details of each approach are as follows:

Identity: We assume that $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ are appropriately sized identity matrices for both Update Method 1 and Update Method 2.

Historical: Update Methods 1 and 2 assume that the process noise covariance is time-invariant, i.e., $Q_t = Q$ and that the measurement noise covariance is $R_t \approx 0$ as the total demand measurements are accurate. The covariances Q^{AC}, R^{AC}, R^{OL} used within the two variations of Q are computed using historical estimation errors, and Q^{AC} is used within the Kalman filter. Update Method 2 updates $\widehat{P}_t$ according to

$$\widetilde{P}_t = \widehat{P}_t - \widehat{P}_t C_t^{\mathrm{T}} \left[C_t \widehat{P}_t C_t^T + R_t \right]^{-1} C_t \widehat{P}_t \tag{6.18}$$

$$\widehat{P}_{t+1} = A_t \; \widetilde{P}_t \; A_t^{\mathrm{T}} + Q_t, \tag{6.19}$$

which are the update equations for the estimation error covariance of a Kalman filter [121, p.190], and Update Method 1 sets $\widehat{P}_t = Q$. Both methods set $\widehat{P}_t^{\mathrm{y}} = (C \widehat{P}_t C^T + R_t)$. In Update Method 2, we use $A_t = \mathrm{blkdiag}(A_t^{\star}, 0) \in \mathbb{R}^3$ to update $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ where $A_t^{\star}$ is the state update matrix of the (LTI or LTV) AC demand model; constructing A_t this way assumes that errors in the AC demand model are decoupled from errors in the OL demand model, and that the errors of the OL demand model are independent at each time-step.

Real-time: Update Methods 1 and 2 assume that $\widehat{P}_t = Q_t$ where the covariances are computed at each time-step using measurements of the AC and OL demand. Variance R_t is computed at each time-step using measurements of the total demand. Both methods set $\widehat{P}_t^{\mathrm{y}} = (C\widehat{P}_t C^T + R_t)$.

6.6.3 Kalman filter benchmark

A set of Kalman filters are used to establish a benchmark for the DFS algorithm. A Kalman filter uses measurements, an assumed system model, and known statistics of random variables, which are assumed to be zero-mean and normally distributed, to estimate the value of dynamic system parameters, i.e., the system state, at each time-step. Additional background on Kalman filters can be found in [121].

We use the LTV AC demand models within the Kalman filters. For each LTV model, the covariance of the process noise is computed using a week of historical data, where the true state is generated using the measured AC demand and the LTV model's matrices. The Kalman filter estimates the state of the AC demand model, i.e., $\theta_t = x_t$ where x_t is $\widehat{x}_t^{\mathrm{LTV1}}$ or $\widehat{x}_t^{\mathrm{LTV2}}$, using output pseudo-measurements of the AC demand $\widetilde{y}_t^{\mathrm{AC}} = y_t - \widehat{y}_t^{\mathrm{OL}}$. We assume that y_t is noise-free, but $\widetilde{y}_t^{\mathrm{AC}}$ is noisy due to OL demand prediction error. The covariance of the measurement errors depends on the OL demand model used, and is computed for each model using a week of historical errors.

We use one Kalman filter for each model pair in the set $\mathcal{M}^{\mathrm{KF}}$. We compare the performance of the DFS algorithm to that of the best Kalman filter (BKF), which takes the lowest *ex post* root mean squared error (RMSE) achieved by a Kalman filter within the set $\mathcal{M}^{\mathrm{KF}}$, and the average Kalman filter (AKF), which is the average RMSE across all of the Kalman filters.

6.6.4 Data

We test the methods on data from Aug 3-5, 10-14, 17, and 18, where the commercial data is from 2009 and the residential data is from 2015, and we refer to these days as the testing days. Note that the dates in both years pertain to the same days of the week. The household and commercial demand data for Aug 3 were used to determine the set of houses included on the feeder and construct the plant. To generate the

Table 6.1: Parameter η^{s} Used in the Scenarios in Section 6.6.5

Model Set	$\mathcal{M}^{\mathrm{Full}}$	$\mathcal{M}^{\mathrm{Full}}$	$\mathcal{M}^{\mathrm{Red}}$	$\mathcal{M}^{\mathrm{Red}}$	$\mathcal{M}^{\mathrm{KF}}$	$\mathcal{M}^{\mathrm{KF}}$
Update Method	1	2	1	2	1	2
η^{s}	0.013	0.015	0.4	0.013	0.4	0.5

MLR regression models of the AC and OL demand, we use data from June 24 to Aug 2, 2015 and commercial data from June 24 to Aug 2, 2009. The LTI and LTV models of the AC demand were constructed using device-level data from individual air conditioners from May 2 to Aug 2, 2015. The TOD regression models and Kalman filter covariance matrices were generated using data from the week preceding Aug 3.

6.6.5 Investigation of Algorithm Performance

Table 6.1 gives the settings of η^{s}. Identity matrices are used as covariances according to Section 6.6.2 throughout the case studies in this section. Also, we set $\lambda = \eta^{\mathrm{r}} = 1.0 \times 10^{-5}$ across all case studies in this subsection. Parameter λ dictates the amount of weight shared amongst the models, where values near 1 force the DFS algorithm to generate estimates that are close to an average of the predictions of all models. By using a λ value near 0, a single model can dominate the estimate if one model is more accurate than the rest. Parameters η^{r} and η^{s} were roughly tuned to achieve qualitative characteristics of fast switching between models without overfitting. The optimal η^{r} and η^{s} for a given day will generally not be optimal across all days, and so tuning to achieve the desired qualitative characteristics is appropriate. In practice, η^{r} and η^{s} can be tuned based on recent historical data, and λ can be tuned based on the historical accuracy of the models within the algorithm. An avenue for future research is to develop methods for online parameter tuning using real-time data.

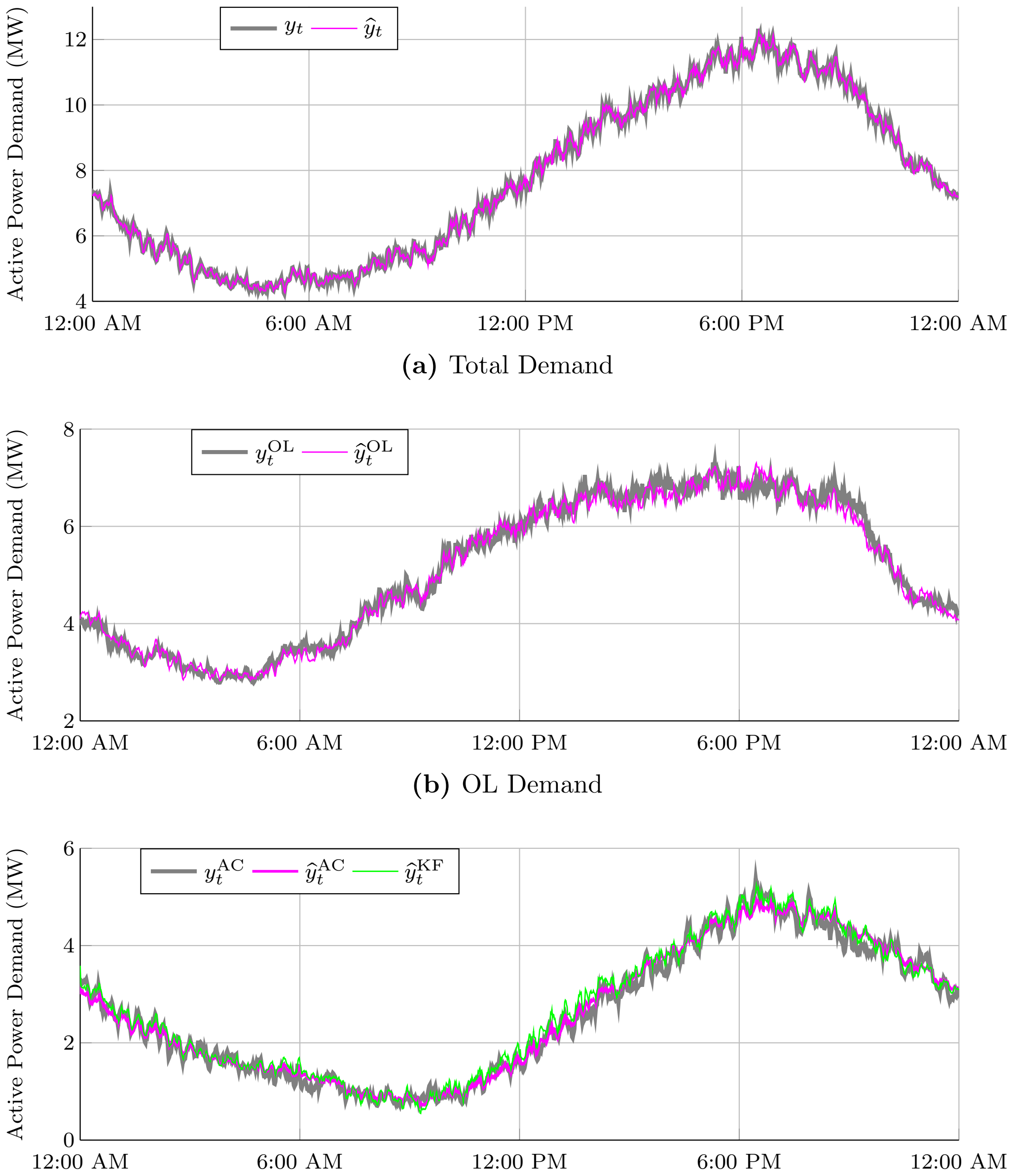
Figure 6.5 depicts time series for the Aug 17 case study with $\mathcal{M}^{\mathrm{Red}}$ while using Update Method 1. Figure 6.6 shows the evolution of the dominant model weights. The weights of the remaining models are summed and referred to as “Other Models.” In this scenario, the total demand is accurately separated into its AC demand and OL demand components in real-time, where the RMSE of the total demand, AC demand, and OL demand is 93.2 kW, 151.0 kW, and 150.8 kW, respectively. In this scenario, DFS produces a more accurate AC demand estimate than BKF, which has an AC demand RMSE of 177.3 kW. The RMSE of the AC demand for AKF is 214.0 kW. The majority of the weight is initially given to the “Other Models,” because we initialize all models with the same weight. As the simulation progresses, the weight shifts between different model combinations. Since the combinations $\{\Phi^{\mathrm{AC,LTV2}}, \Phi^{\mathrm{OL,MLR}}\}$ and $\{\Phi^{\mathrm{AC,LTV1}}, \Phi^{\mathrm{OL,MLR}}\}$ perform best, they eventually earn more weight and dominate the predictions. It is unsurprising that the $\Phi^{\mathrm{OL,MLR}}$ is the most accurate OL demand model as it captures weather and time variables with the most detail, and it is unsurprising that $\Phi^{\mathrm{AC,LTV1}}$ and $\Phi^{\mathrm{AC,LTV2}}$ are the most accurate AC demand models as they capture the physical phenomenon driving changes in the AC demand as the outdoor temperature changes.

Figure 6.7 presents the minimum, mean, and maximum RMSE across the full set of testing days for the total demand, the AC demand, and the OL demand for each DFS scenario. For comparison, BKF achieves a minimum RMSE of 148.4 kW, a mean RMSE of 195.3 kW, and a maximum RMSE of 318.9 kW for the AC demand, and AKF achieves a minimum RMSE of 173.1 kW, a mean RMSE of 259.4 kW, and a maximum RMSE of 357.5 kW for the AC demand. The model corresponding to the BKF varies from day to day and so it is not possible to obtain a single Kalman filter that always outperforms DFS.[2] The models corresponding to the BKF vary from day to day. To demonstrate the value of the measurement-based updates, we generated results for the full set of testing days using the model set $\mathcal{M}^{\mathrm{Red}}$ and with $\eta^{\mathrm{s}} = 0$; the measurement-based update is irrelevant with this parameter setting. The resulting total demand, AC demand, and OL demand RMSEs were 260.4 kW, 254.2 kW, and 245.2 kW, respectively. These are significant increases over the DFS scenarios using $\mathcal{M}^{\mathrm{Red}}$.

The scenarios using $\mathcal{M}^{\mathrm{Full}}$ have significantly higher AC demand RMSEs than the simulations using $\mathcal{M}^{\mathrm{Red}}$ as well as the BKF and AKF simulations. Each of the LTI models may be accurate for a portion of the day when the AC demand is near the steady-state demand of the particular model. However, as the AC demand changes due to changes in the outdoor temperature, a given LTI model will become highly inaccurate. The DFS algorithm takes time to shift weight from the inaccurate model that was heavily weighted to the new model, and this results in increased RMSE. Eliminating these “bad models”, by using $\mathcal{M}^{\mathrm{Red}}$ rather than $\mathcal{M}^{\mathrm{Full}}$, eliminates this issue.

The scenarios using $\mathcal{M}^{\mathrm{Red}}$ generally do better, in terms of AC demand RMSE, than AKF and worse than BKF. On some simulated days DFS also outperforms BKF, as was shown in Fig. 6.5. Within this set of simulations, Update Method 2

[2] Choosing BKF on a particular day and applying it to all other days, we find that DFS performs better on approximately half of the days when using $\mathcal{M}^{\mathrm{Red}}$. However, the loss function, divergence function, and initial model weights within DFS could be modified based on historical performance, which would improve its performance relative to the Kalman filter.



(a) Total Demand

(b) OL Demand

(c) AC Demand

Figure 6.5: Total, OL, and AC demands versus their DFS estimates (Aug 17, $\mathcal{M}^{\mathrm{Red}}$, Update Method 1). The best Kalman filter estimate of the AC demand is also shown.

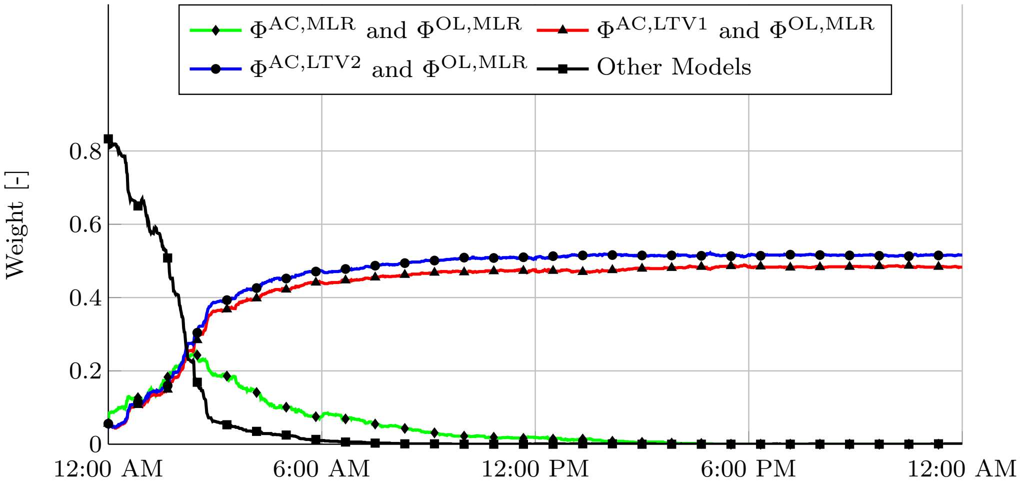
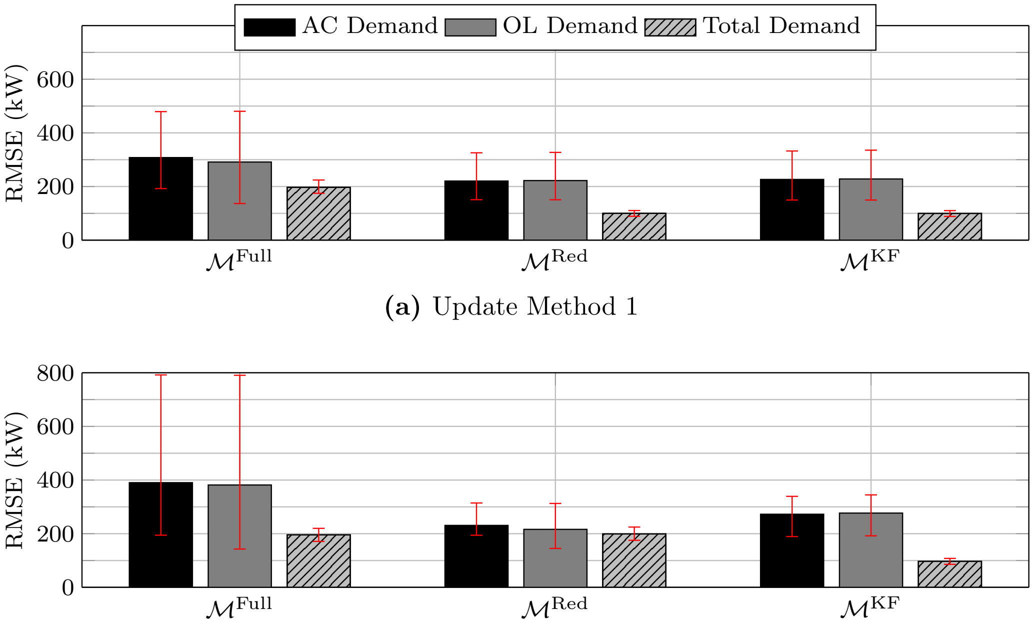


Figure 6.6: Model weights (Aug 17, $\mathcal{M}^{\mathrm{Red}}$, Update Method 1).



(a) Update Method 1

(b) Update Method 2

Figure 6.7: Minimum, mean, and maximum RMSE (kW) for the DFS scenarios in Section 6.6.5.

results in higher AC demand RMSE than Update Method 1. The increased RMSE in Update Method 2 versus Update Method 1 can be explained due to the usage of only two discrete states within the LTV models. Specifically, the states reach their steady-state values rapidly, and so the measurement-based updates to the state do not persist for very long, whereas the measurement-based updates to the output used in Update Method 1 do. Using LTV models with more discrete states may allow Update Method 2 to achieve better RMSE, but this would complicate system identification.

Finally, the scenarios with $\mathcal{M}^{\text{KF}}$ result in larger AC demand RMSE than those with $\mathcal{M}^{\text{Red}}$. The MLR model of the AC demand is often weighted heavily in the $\mathcal{M}^{\text{Red}}$ simulations, especially for Update Method 2. Given this, it makes sense that excluding this model would result in increased RMSE.

6.6.6 Sensitivity to the Parameters η^{r} and η^{s}

We apply DFS to the full set of testing days while varying η^{r} and η^{s} to investigate the impact of those parameters on the results. We vary η^{s} from 0.0 to 0.9 using increments of 0.1, and we vary η^{r} from 10^{-7} to 10^{-3}, where we increment the order (i.e., 10^{-7}, $10^{-6}, \dots$). We apply DFS using every combination of these parameter values while using Update Method 1, identity covariances according to Section 6.6.2, $\mathcal{M}^{\text{Red}}$, and $\lambda = 1.0 \times 10^{-5}$ as in Section 6.6.5.

Figure 6.8 provides the average RMSE of the AC demand across the full set of testing days for each parameter value combination. With η^{s} near zero the RMSE is relatively large as DFS makes small adjustments to the model predictions based on the realized prediction errors. The RMSE with η^{s} near zero decreases slightly as η^{r} increases because this allows for faster transitions in the weighting of the models. However, it should be noted that at larger η^{r} values (e.g., 10^{-3}), the model weights within DFS become erratic or noisy, and overfitting is possible. The RMSE is also relatively high with large η^{s} (e.g., 0.9) and small η^{r} as DFS adjusts the model predictions too aggressively and the model weights change slowly. Alternatively, as η^{r} increases with large η^{s}, the RMSE decreases, but again the weights become vulnerable to overfitting. The RMSE using moderate η^{s} values (e.g., from 0.2-0.7) are similar. The η^{s} and η^{r} values used within Section 6.6.5 do not achieve the lowest RMSE, but they achieve low RMSE while ensuring changes in the weights are reasonably fast but not erratic.

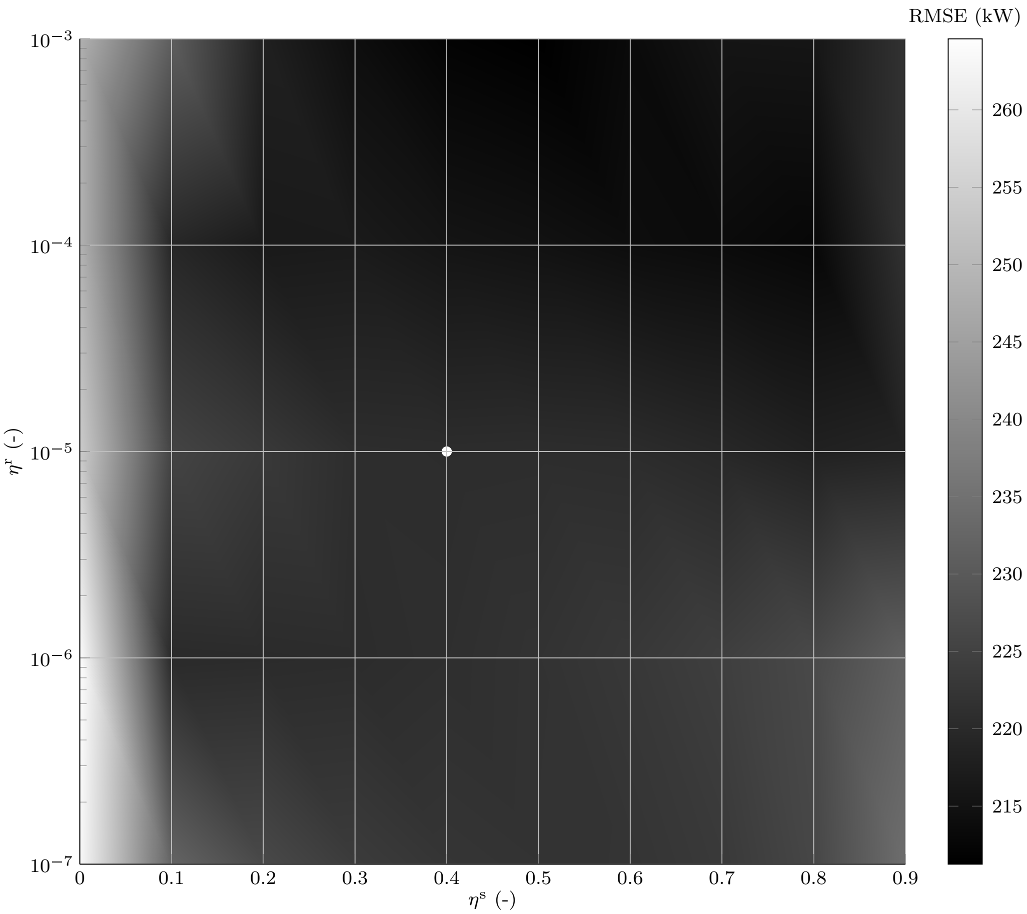


Figure 6.8: Average RMSE of the estimated AC demand across all days as a function of η^{s} and η^{r}, using Update Method 1, identity covariances according to Section 6.6.2, $\mathcal{M}^{\mathrm{Red}}$, and $\lambda = 1.0 \times 10^{-5}$ where the marker indicates the parameter values used in Section 6.6.5.

6.6.7 Sensitivity to the Parameter λ

We apply DFS to the full set of testing days while varying λ from 1.0×10^{-7} to 1.0 to investigate the impact of λ on the results. Within DFS, we use Update Method 1, identity covariances according to Section 6.6.2, and $\mathcal{M}^{\mathrm{Red}}$, and we also set $\eta^{\mathrm{s}} = 0.4$, and $\eta^{\mathrm{r}} = 1.0 \times 10^{-5}$ as in 6.6.5.

Figure 6.9 gives the average RMSE of the estimated AC demand across all days as a function of λ. The average RMSE of the AC demand decreases as λ is increased from 1.0×10^{-7} and reaches a minimum RMSE of 211.3 kW with $\lambda = 0.005$. As λ increases from 0.005, the RMSE increases until it remains relatively constant from 0.1 to 1.0. While we set λ in Section 6.6.5 to allow a single model to dominate the DFS estimate if one model proved to be more accurate than the rest, Fig. 6.9 indicates that tuning λ on a set days that are similar to the testing days may allow a reduction in the RMSE.

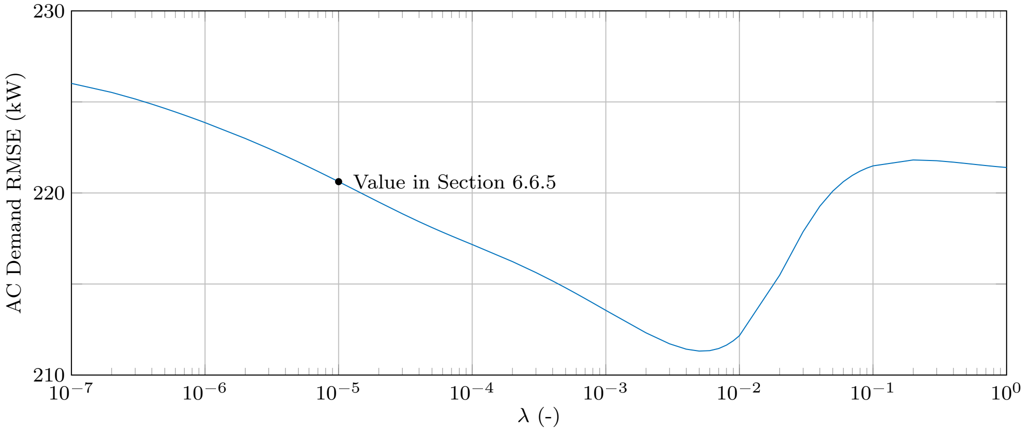


Figure 6.9: Average RMSE of the estimated AC demand across all days as a function of λ, using Update Method 1, identity covariances according to Section 6.6.2, $\mathcal{M}^{\text{Red}}$, $\eta^{\text{s}} = 0.4$, and $\eta^{\text{r}} = 1.0 \times 10^{-5}$.

Table 6.2: Parameters η^{s} and η^{r} Used in Update Methods 1 and 2 in Section 6.6.8

Update Method	1	1	1	2	2	2
Covariance	Identity	Historical	Real-Time	Identity	Historical	Real-Time
η^{s}	0.4	0.5	0.5	0.013	0.5	1.0
η^{r}	1.0×10^{-5}	10	1.0×10^{-3}	1.0×10^{-5}	10	1.0×10^{-3}

6.6.8 Investigating the Impact of Including Error Covariances

This section presents case studies that illustrate the impact of including different covariance matrices within the algorithm implementations. Table 6.2 lists the values of the parameters η^{s} and η^{r} used in each scenario; we set $\lambda = 1.0 \times 10^{-5}$ in all scenarios. We tuned η^{r} and η^{s} qualitatively, as described in Section 6.6.5. All case studies run Update Method 1 and 2 using $\mathcal{M}^{\text{red}}$.

Figure 6.10 presents time series of the total demand, OL demand, AC demand, their respective estimates, and the model weights from the August 4 case study while running Update Method 1 with covariances generated from historical data. In Fig. 6.10d, Φ^{Other} is used to denote the combined weight of all model combinations not explicitly specified. Table 6.3 summarizes the mean, minimum, and maximum RMSE for each demand component across the simulated days and scenarios. Figure 6.11 presents time series of the AC demand and various estimates across several scenarios from the August 11 case study.

From Fig. 6.10c, it is clear that, in this case, Update Method 1 effectively estimates the AC demand in real-time. In this scenario, BKF achieves an RMSE of 148.4 kW for the AC demand, and Update Method 1 performs similarly, achieving an

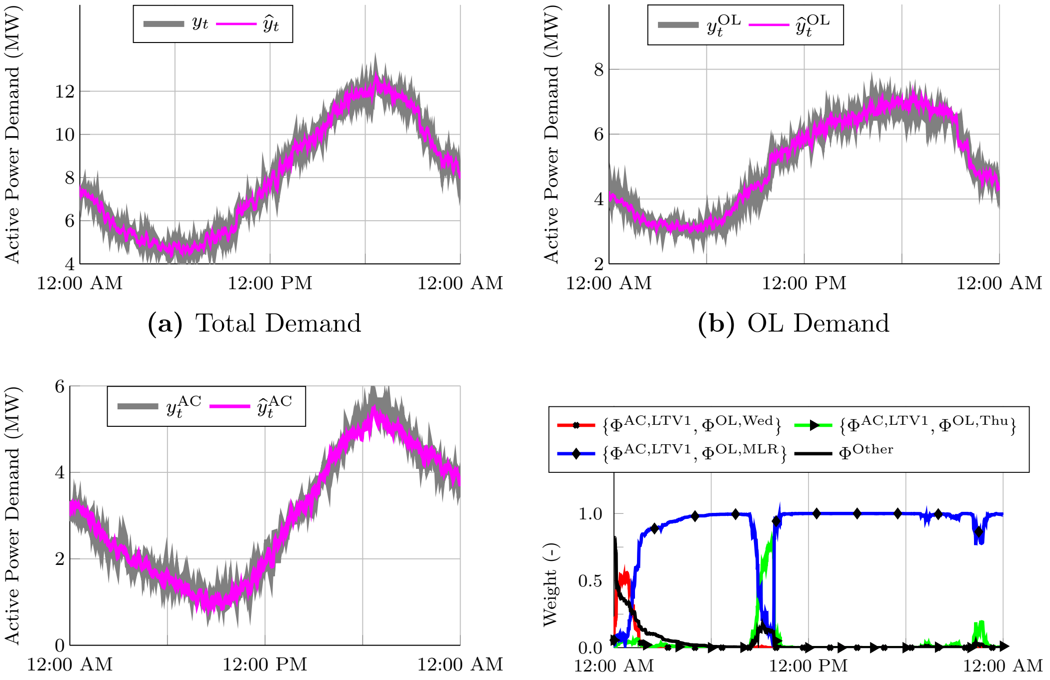
Table 6.3: Mean, Minimum (Min), and Maximum (Max) RMSE in kW over 10 Test Days for each Update Method and Covariance Computation Approach from Section 6.6.8

Update Method	Covariance	Total Demand			AC Demand			OL Demand		
		Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
1	Identity	88.9	100.0	110.5	151.0	220.6	325.8	150.8	222.3	327.2
1	Historical	98.4	114.8	123.2	155.0	252.2	371.5	150.2	250.1	372.5
1	Real-Time	146.6	154.3	168.4	120.2	125.3	131.8	104.8	114.5	130.5
2	Identity	175.4	199.1	224.8	194.2	230.9	314.5	145.0	216.2	312.7
2	Historical	100.5	119.5	126.1	192.0	259.8	311.5	190.6	265.5	320.2
2	Real-Time	120.8	125.2	129.1	104.0	116.5	140.1	96.6	109.4	131.9
BKF	Historical	-	-	-	148.4	195.3	318.9	-	-	-
AKF	Historical	-	-	-	173.1	259.4	357.5	-	-	-

RMSE of 155.0 kW for the AC demand. It should be noted that Update Method 1 is determining the model of the underlying system in real-time, as can be seen in Fig. 6.10d. Alternatively, the BKF algorithm selects the most accurate model after the simulation, which is not feasible in practice. For comparison, AKF achieves an RMSE of 173.1 kW. The weights within Update Method 1 is initially dominate by Φ^{Other}, which makes sense as the weight of each model combination is initialized to the same value. As the simulation progresses, the weight shifts to $\{\Phi^{\text{AC,LTV1}}, \Phi^{\text{OL,MLR}}\}$, which is the most accurate model. At points of the simulation, it loses accuracy, and the weight shifts to other model combinations during those times. The total demand is estimated closely, which can be achieved based on the parameter settings. Finally, it should be noted that while Update Method 1 did not achieve lower RMSE than BKF in this case, in some cases it does outperform BKF.

As Table 6.3 shows, Update Method 1 achieves AC demand RMSEs that are worse than BKF but generally better than the AKF when using realistic (i.e., historical) covariance data. Update Method 2 achieves AC demand RMSE that is comparable to the AKF. Figure 6.11a shows time series for AKF, BKF, and DFS using historical covariances. When using unrealistic (i.e., real-time) covariance data, both Update Method 1 and 2 outperform BKF, which is still using historical data to compute the covariance matrices. An example of this is shown in Fig. 6.11b.

Figure 6.11c provides example time series of the AC demand estimates for Update Methods 1 and 2 when using historical covariances, and Fig. 6.11d provides similar example time series when using real-time covariance data. As can be seen in Fig. 6.11c, the Update Method 1 generally achieves better RMSE for the AC demand than Update Method 2. However, as can be seen in Fig. 6.11d Update Method 2 achieves lower RMSE than Update Method 1 when real-time errors are used to generate the



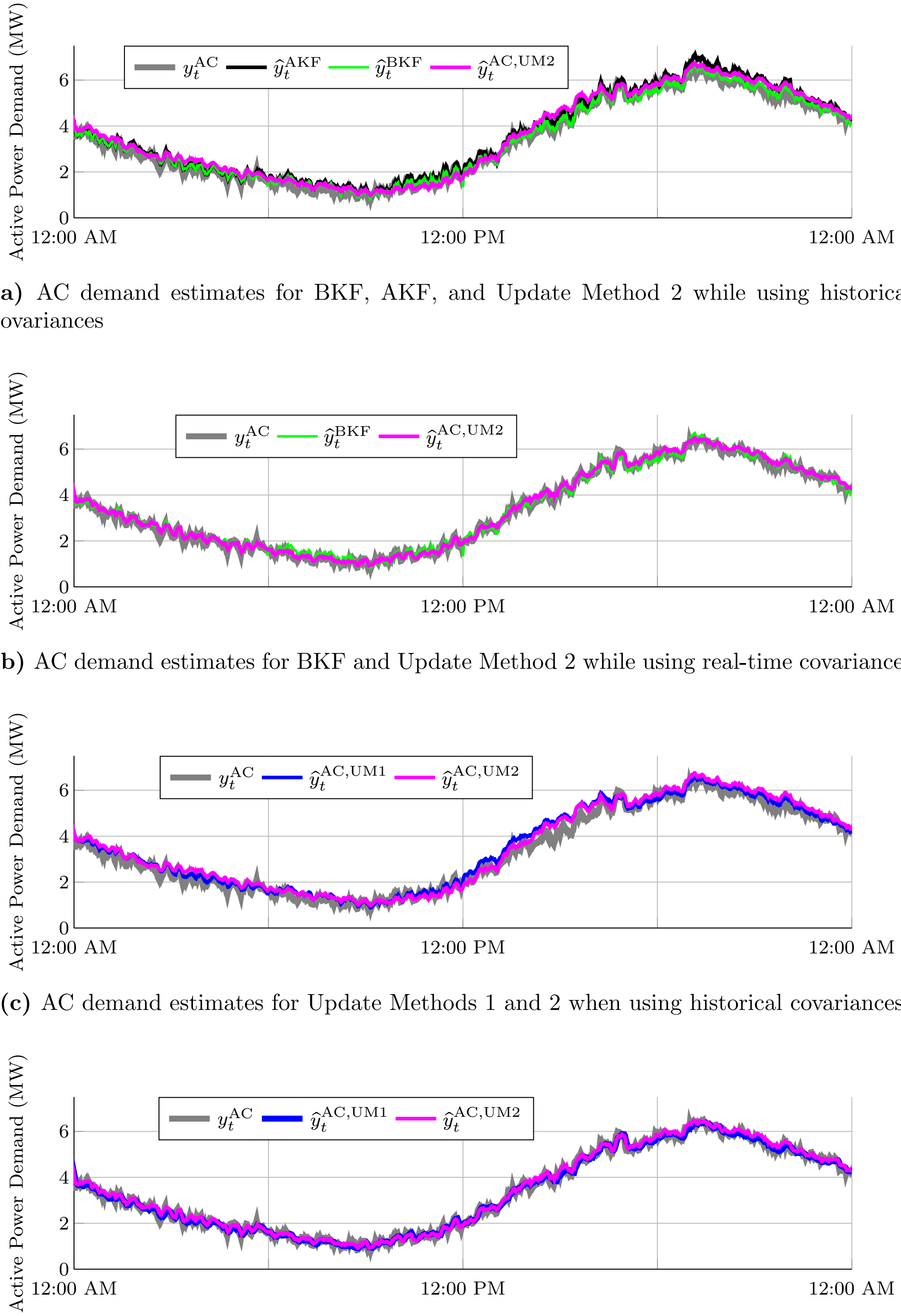
(a) Total Demand

(b) OL Demand

(c) AC Demand

(d) Model Weights

Figure 6.10: Time series from Section 6.6.8 of the total, OL, and AC demands versus their estimates as well as times series of the weights from the August 4 case study while running Update Method 1 with historical covariances on model set $\mathcal{M}^{\mathrm{Red}}$



(a) AC demand estimates for BKF, AKF, and Update Method 2 while using historical covariances

(b) AC demand estimates for BKF and Update Method 2 while using real-time covariances

(c) AC demand estimates for Update Methods 1 and 2 when using historical covariances

(d) AC demand estimates for Update Methods 1 and 2 when using real-time covariances

Figure 6.11: Time series of the AC demand and various estimates from the August 11 case study in Section 6.6.8 where UM1 and UM2 refer to Update Method 1 and 2, respectively

covariance matrices. Part of the reasoning for this is that the LTV AC demand models only include two states, and for a given outdoor temperature, the models rapidly converge to a steady-state value. When running Update Method 2, this means that the measurement-based adjustment at a given time-step may not have an effect on the model's predictions after several time-steps. Alternatively, Update Method 1 continually adjusts the model predictions based on its accuracy, and by separating these adjustments from the model, these adjustments persist.

Also, our method of computing the covariances with historical data degrades performance. This implies that our assumptions regarding the errors are overly coarse. However, the inclusion of unrealistically accurate covariance information, which is done when using real-time covariance data, the performance of Update Methods 1 and 2 improve dramatically.

6.7 Chapter VI Conclusions

In this chapter, we applied an online learning algorithm, DFS, which uses DMD together with the Fixed Share algorithm, to estimate the real-time AC demand on a distribution feeder using feeder demand measurements, weather data, and system models. Two implementations of algorithms based on DMD were developed and compared via case studies. Our results showed that DFS can effectively estimate the real-time AC demand on a feeder. DFS achieved lower AC demand RMSE than the average across a set of Kalman filters. When selecting the most accurate Kalman filter *ex post*, DFS generally results in larger RMSE. However, DFS learns the most accurate model, or combination of models, in real-time whereas the best Kalman filter can only be chosen after the simulation. The performance of DFS depends heavily on the inclusion of models within its set. Including models that are inaccurate for majority of the day degraded the algorithm performance as did removing models that were frequently weighted heavily. Furthermore, including prediction error statistics, e.g., error covariances, into the update equations influences the algorithm performance significantly.

In this work, we separated the demand into only two components. However, the algorithm is applicable to scenarios with more than two components, assuming that we have at least one model of each demand component. As the number of components increases, it may become more difficult to disaggregate them, but these difficulties could be counteracted by incorporating more real-time measurements, e.g., the reactive power demand. Future work will develop improved AC demand models and incorporate active control into the problem framework.

CHAPTER VII

Separating a Feeder's Demand into Components Using Smart Meter and Feeder Measurements

Distributed energy resources (DERs) such as residential solar installations and residential demand responsive loads are becoming more prevalent within distribution networks [122, 123]. These technologies can cause more variability in the power flows within the distribution network, requiring additional sensing to maintain acceptable power quality to end-users. Traditionally, real-time sensing capabilities in distribution networks have been limited to within the distribution substations [124]. However, additional real-time sensing capabilities, e.g., of the voltage magnitude and phase angle, at points within the distribution feeder are becoming more common [125, 126]. In addition, smart meters have been widely deployed in distribution networks [123], and they are capable of providing power usage information at the device level using energy disaggregation algorithms [49]. Due to smart meter communication limitations, the measurements may not be available in real-time, or they may only be available intermittently, e.g., every fifteen minutes.

In this chapter, we seek to utilize real-time and historical measurements from the distribution feeder and smart meters to perform real-time, feeder-level energy disaggregation on time-scales of seconds to minutes. Real-time, feeder-level energy disaggregation seeks to separate the measured demand of a distribution feeder into components, e.g., the aggregate demand/generation of different types of loads/generators. It can benefit distribution system operators and demand response providers by providing information about 1) potential balancing reserve requirements due to changes in solar generation, and 2) the real-time aggregate demand of the demand-responsive loads, e.g., as a feedback signal for control algorithms, among other applications.

The energy disaggregation algorithm developed in this chapter is an online learning algorithm that incorporates sensor fusion to utilize real-time measurements from a variety of sources that are available on different timescales. Historical measurements are used to compute models of the demand components, where the models produce predictions of the demand components in real-time. Real-time measurements are then used to adjust these predictions. Case studies apply the algorithm to disaggregate the real-time feeder demand into the total demand of air conditioners (ACs), the total demand of the non-air conditioning loads, and the total demand of network equipment, i.e., the losses and capacitor banks. Unlike existing works in feeder-level energy disaggregation, which neglect the effects of the distribution feeder and do not take advantage of the full real-time sensing capabilities available within the distribution feeder, we account for the distribution feeder explicitly, and we include a general sensor fusion methodology to incorporate real-time measurements that is adaptable to the available measurements.

Related work in feeder-level energy disaggregation includes [86, 96, 127–130]. Ref. [127] uses substation power measurements to disaggregatee the feeder demand into the solar generation on the feeder and the remaining demand on one minute time intervals using multiple linear regression. Ref. [128] and [129] use substation power and voltage measurements to disaggregate loads into categories on one-hour intervals using an artificial neural network and using a method of computing the load components directly, respectively. Ref. [130] uses substation power and voltage measurements to disaggregate loads into categories on one-minute intervals using an artificial neural network. Our prior work, [86, 96], presented in Chapter VI, disaggregates the active power demand of a distribution feeder into two components on one-minute intervals using online learning algorithms. None of these works explicitly model the power consumed by the distribution feeder equipment, and they do not take advantage of real-time measurements outside of the distribution substation.

The contributions of this chapter are the following: 1) we formulate a more general disaggregation problem than [86, 96] that includes reactive power into the formulation, that models the feeder explicity, and that includes sensor fusion methodology to adapt the algorithm to the available real-time measurements; 2) we develop methods to incorporate active and reactive power flow, smart meter active and reactive power, complex voltage, and complex current measurements; 3) we develop models of components of the demand, specifically for the AC loads, the non-AC loads, and the network demand, i.e., the losses and the power injections of capacitor banks; and 4) we evaluate the value of additional real-time measurements on the aggregate modeling and disaggregation accuracy. This work differs from [86, 96, 127–130] in a number of ways. Whereas [86, 96] include only the active demand into the disaggregation algorithm,

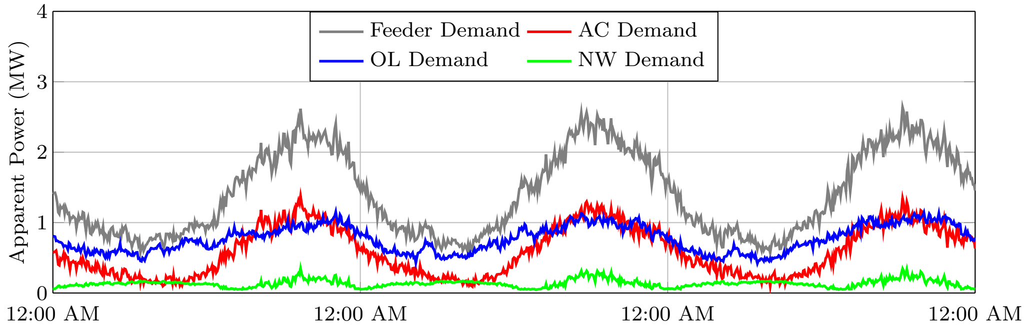


Figure 7.1: Time series of the measured feeder demand and the underlying time series of the three underlying demand components: the AC demand; the OL demand, which consists of the non-AC loads; and the network (NW) demand, which consists of the losses and capacitor bank power injections.

we include the reactive demand and take advantage of additional sensing capability in the distribution network. Unlike, [86, 96, 127–130], which only incorporate real-time measurements from the distribution substation, we incorporate additional real-time measurements from points along the feeder and from smart meters. Also, we incorporate measurements that are available on different time-scales, and we explicitly model the losses and capacitor bank power injections, whereas [86, 96, 127–130] do not.

The remainder of the work is organized as follows: Section 7.1 describes the problem overview; Section 7.2 details the energy disaggregation algorithm and methods to incorporate the various measurements types; Section 7.3 describes the aggregate modeling used within the disaggregation algorithm; Section 7.4 details the case studies; and Section 7.5 concludes the chapter.

7.1 Problem Overview

In this chapter, a power system entity performs real-time feeder-level energy disaggregation to estimate the real-time aggregate demand of N demand components within a distribution feeder on timescales of seconds to minutes. Figure 7.1 depicts example time series, where the entity seeks to disaggregate the measured power flow into a distribution feeder, i.e., the measured total demand of the feeder, into three components: the AC demand; the OL demand, which consists of the non-AC loads; and the network (NW) demand, which consists of the losses and capacitor bank power injections.

Figure 7.2 depicts a block diagram of the problem overview that summarizes the plant (i.e., the physical system of interest), the disaggregation algorithm of the power system entity, and the potential real-time measurements. The plant contains

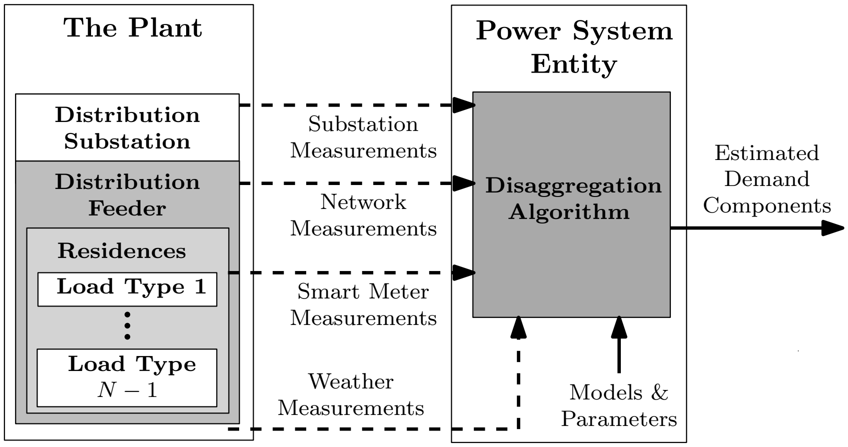


Figure 7.2: Block diagram of the two main components in the problem overview and the real-time measurements

a distribution substation, a distribution feeder, and the residences connected to the distribution feeder. We assume that all loads and capacitor banks are wye connected (line-to-neutral); including delta connected loads is a topic of future research. The residences contain $N-1$ different load types that have active and reactive demand components. The Nth demand component is the NW demand.

We assume that the power system entity receives some combination of substation, network, smart meter, and weather measurements in real-time. Table 7.1 summarizes examples of measurements that could be available, and it summarizes the assumptions on the frequency of their availability for the case studies of Section 7.4. We assume 1) that substation measurements are available as substations are extensively metered [124], 2) that network measurements, i.e., measurements at points within the distribution feeder outside of the substation, may be available from micro-phasor measurement units [126], 3) that real-time smart meter measurements may be available as smart meters have been widely installed, but, if available, the measurements are transmitted to the power system entity at infrequent invervals, e.g., every 10-60 minutes, due to smart meter communication limitations [49], and 4) that weather measurements are available from either nearby weather stations or weather sensing capabilities of the power system entity.

We assume that smart meters have capabilities that enable the computation of the N demand components when real-time smart meter measurements are available. Smart meters are capable of measuring the real power, reactive power, voltage magtnidue, and current magnitude, and phase difference between the current and voltage of a residence [49]. The smart meter measurements may be transmitted to the power system entity infrequently in real-time, but smart meters can sample measurements

Table 7.1: Examples of Potential Real-Time Measurements

Measurement Type	Frequency
Substation Measurements	
Active and reactive power flow into the feeder	1 minute
Complex current flowing into the feeder	1 minute
Complex voltage at the feeder head	1 minute
Network Measurements	
Active and reactive power flow within the feeder	1 minute
Complex voltage within the feeder	1 minute
Smart Meter Measurements	
Active and reactive power flowing into the residence	10-60 minutes
Current magnitude flowing into the residence	10-60 minutes
Voltage magnitude at the residence	10-60 minutes
Voltage and current phase difference at the residence	10-60 minutes
Weather Measurements	
Outdoor temperature	1 minute

on timescales of seconds to minutes and transmit histories of measurements [49]. We assume that these capabilities enable building-level energy disaggregation algorithms to separate the measured building/residence demand into the $N-1$ demand components [49], which allows the computation of the real-time aggregate demand of the $N-1$ load types when smart meter measurements are transmitted. The NW demand can be computed by subtracting the total smart meter demand from substation measurements of the total feeder demand. Due to these smart meter capabilities, we also assume that historical smart meter measurements are always available on timescales of seconds to minutes, regardless of whether they are transmitted in real-time.

Figure 7.3 summarizes the feeder-level energy disaggregation algorithm developed in this work that uses demand component estimates for the previous time-step, aggregate models (and their parameters), and substation, network, smart meter, and weather measurements to compute demand component estimates for the present time-step. The algorithm first fuses the available substation, network, and smart meter measurements. Then, the algorithm computes measurement-based predictions using the fused measurements. Following this, the algorithm computes model-based estimates using aggregate models for each demand component, weather measurements, and possibly substation measurements. Finally, the algorithm computes demand component estimates for the present time-step based on the model-based and measurement-based predictions.

```
Input: Demand component estimates for previous time-step; aggregate models; substation, network, smart meter, and weather measurements
1: Fuse the available substation, network, and smart meter measurements
2: Compute measurement-based predictions using the fused measurements
3: Compute the model-based predictions using the aggregate models, weather measurements, and possibly substation measurements
4: Estimate the demand components for the present time-step
Output: Demand component estimates for present time-step
```

Figure 7.3: Steps within the energy disaggregation algorithm to compute estimates of the demand components for the next time-step

7.2 The Energy Disaggregation Algorithm

In this section, we describe the feeder-level energy disaggregation algorithm, which consists of an estimation algorithm, sensor fusion methodology, and output equations for distribution system measurements. Subsection 7.2.1 summarizes a previously developed estimation algorithm. Subsection 7.2.2 develops sensor fusion methodology, which allows the disaggregation algorithm to use measurements from multiple sources on different timescales. Subsection 7.2.3 develops output equations for four measurement types within the distribution system: power flows, squared voltage magnitude differences, voltage angle differences, and smart meter measurements. These output equations can then be included into the estimation algorithm via the sensor fusion methodology.

The disaggregation algorithm does not model coupling between the three phases of the distribution network. As a result, the algorithm is applied to each phase individually, and notation specifying the specific phase is not used in the discussion below for simplicity. Note that while coupling is not considered in the disaggregation algorithm, the plant within the case studies includes a three-phase distribution feeder model with unbalanced loads.

7.2.1 The Estimation Algorithm

The estimation algorithm is a modified version of Dynamic Mirror Descent (DMD). DMD is an online learning algorithm that was developed in [90], and it was modified in [96], which was presented in the previous chapter. DMD iterates between 1) a measurement-based update that computes an adjusted state estimate via a convex optimization problem that incorporates newly arrived measurements, and 2) a model-based update that advances the adjusted state estimate to the next time-step via a possibly nonlinear model. Ref. [96] modified DMD to separate the measurement-based update from the model-based update, allowing a wider range of models to be used within the algorithm. We use the modified DMD algorithm in this chapter.

The estimation algorithm computes estimates $\widehat{\theta}_t \in \Theta$ of a system parameter, or state, $\theta_t \in \Theta$ that changes over time using measurements $y_t \in \mathcal{Y}$ that arrive sequentially in time. The domain of the state $\Theta \subset \mathbb{R}^p$ is a bounded, closed, convex feasible set, and $\mathcal{Y} \subset \mathbb{R}^q$ is the domain of the measurements. The details of the algorithm are the following:

$$\begin{aligned} \widehat{\kappa}_{t+1} &= \underset{\theta \in \Theta}{\arg\min}\, \eta^{\mathrm{s}} \left\langle \nabla \ell_t(\widehat{\theta}_t),\ \theta \right\rangle + D\left(\theta \| \widehat{\kappa}_t\right) && (7.1)\\ \breve{\theta}_{t+1} &= \Phi(\breve{\theta}_t) && (7.2)\\ \widehat{\theta}_{t+1} &= \breve{\theta}_{t+1} + \widehat{\kappa}_{t+1}. && (7.3) \end{aligned}$$

where $\widehat{\kappa}_t$ is a quantity that accumulates the measurement-based adjustment to the state estimate $\widehat{\theta}_t$, and the remaining quantities are detailed below. Eq. (7.1) updates $\widehat{\kappa}_t$ to include the new measurement-based update, (7.2) creates a model-based prediction for the next time-step, and (7.3) adjusts the model-based prediction using the accumulated measurement-based adjustments of $\widehat{\kappa}_t$. In the above, $\eta^{\mathrm{s}} \in (0, 1]$ is a step-size parameter that the user selects, $\langle \cdot,\ \cdot \rangle$ is the standard dot product, $\nabla \ell_t(\cdot)$ is a gradient or subgradient of the loss function $\ell_t(\cdot)$, which penalizes deviations in the predicted measurement versus the actual measurement y_t, θ is the optimization variable, and $D\left(\cdot \| \cdot\right)$ is a Bregman divergence that the user can choose. The variable $\breve{\theta}_t$ is the model-based prediction, and $\Phi(\cdot)$ is the model.

The loss function $\ell_t(\widehat{\theta}_t)$ contains a mapping from $\widehat{\theta}_t$ to the estimated output $\widehat{y}_t$. In this work, we assume that this mapping is linear, i.e., that $\widehat{y}_t = C_t\widehat{\theta}_t$ where the output matrix is C_t. We also assume that $\ell_t(\widehat{\theta}_t)$ contains an output estimation error covariance matrix Π_t^{y}, i.e., a covariance matrix that quantifies the accuracy of the output estimates, as this can improve disaggregation accuracy [86].

7.2.2 Sensor Fusion Methodology

Sensor fusion is commonly used within Kalman filtering [131], but it has not been applied feeder-level energy disaggregation. We assume that the following output equation describes each measurement:

$$y_t = C_t\theta_t + w_t, \tag{7.4}$$

where w_t is the measurement noise, which we assume is a random vector with positive-definite covariance R. Given two measurements y_t^1, y_t^2, their respective output matrices C_t^1, C_t^2, and their respective measurement error covariances R^1 and R^2, a combined/fused output equation is formed as

$$\begin{bmatrix} \widehat{y}_t^1 \\ \widehat{y}_t^2 \end{bmatrix} = \begin{bmatrix} C_t^1 \\ C_t^2 \end{bmatrix} \widehat{\theta}_t.$$

We assume that the measurement noise for separate sensors are independent, and so

$$R = \begin{bmatrix} R^1 & 0 \\ 0 & R^2 \end{bmatrix}.$$

The output estimation error covariance used within the estimation algorithm is computed as $\Pi_t^{\mathrm{y}} = C_t \widehat{\Pi}_t C_t^{\mathrm{T}} + R$, where $\widehat{\Pi}_t$ is the state estimation error covariance. The formulation of Π_t^{y} assumes that the estimation errors are independent from each other and from the measurement noise. Because R is positive definite, Π_t^{y} is invertible.

7.2.3 Output Equations

In this section, the feeder is described from a graphical perspective, portions of the feeder are defined using the graphical representation, a state is defined for feeder-level energy disaggregation using the graphical representation, and a general output matrix is defined for this state. In the following subsections, output matrices are developed using the state for active and reactive power flow measurements, squared voltage magnitude difference measurements, and voltage angle difference measurements based on power flow modeling in a radial distribution feeder; an output equation is also developed for smart meter measurements using energy disaggregation at the building level and substation power measurements. We exclude time indices in this subsection to simplify notation; all sets are time invariant, and we point out the other time invariant quantities.

The feeder can be viewed as a graph consisting of a set of buses/nodes $\mathcal{N}$ and a set of power lines/edges $\mathcal{E}$ connecting the nodes in a radial/rooted-tree topology. In a radial/tree topology, 1) the graph is connected, i.e., a path consisting of a sequence of edges and nodes exists from every node to any other node, and 2) no path exists from a node back to itself without backtracking over edges. In a rooted tree, one node is designated as the root, a node within the substation in this case, and a parent-child designation exists between nodes connected via an edge. An edge connecting nodes m and n is labeled as $(m, n) \in \mathcal{E}$ where m is the parent (i.e., the upstream node, or the node closer to the root) and where n is the child (i.e., the downstream node, or the node further from the root). The root has no parent node, every other node has one parent, and nodes can have multiple/no child nodes. The descendants of a given node n are the set of nodes downstream of n (i.e., the set of nodes that contain n on the path from the node of interest to the root).

In this setting, knowledge of the feeder topology means that the adjacency matrix is known and that the node that each residence, and its constituent loads, is connected to is known. The adjacency matrix indicates which nodes are children of each node. The feeder topology could be known, e.g., if it is provided by the distribution system operator, or it could be estimated, e.g., using voltage magnitude measurements from smart meters [132].

If the topology is known or estimated, then portions of the feeder can be defined as subsets of nodes and edges. We define $\mathcal{N}_n$ as the set of node n plus its descendants, $\mathcal{E}_n$ as the set of edges downstream from n, and $\mathcal{E}_{n+}$ as the set $\mathcal{E}_n$ plus $(p,n) \in \mathcal{E}$ where p is the parent of n. For an arbitrary set $\mathcal{Y}$, we use the notation $\mathcal{Y}_{m\backslash n}$ to indicate the set $\mathcal{Y}_m$ excluding the elements of $\mathcal{Y}_n$, and we use this notation to define the sets $\mathcal{N}_{m\backslash n}$ and $\mathcal{E}_{m\backslash n+}$. Finally, we construct two portions of the feeder $\mathcal{A}$ and $\mathcal{B}$ that each consist of a set of nodes and a set of edges. The portion of the feeder $\mathcal{B}$ consists of the edge (m,n) and all nodes and edges downstream from n, and $\mathcal{A}$ is the remainder of the feeder starting from the substation and excluding the nodes and edges in $\mathcal{B}$, i.e., $\mathcal{B} = \{\mathcal{N}_n, \mathcal{E}_{n+}\}$ and $\mathcal{A} = \{\mathcal{N}_{0\backslash n}, \mathcal{E}_{0\backslash n+}\}$, where 0 denotes a node within the substation. Note that $\mathcal{A}$ and $\mathcal{B}$ are a partition of the feeder, i.e., they have no nodes/edges in common and combining the two sets together forms the set of all of the nodes and edges within the feeder.

We define the feeder-level energy disaggregation state as

$$\theta \triangleq \begin{bmatrix} P_{\mathcal{A}}^{\mathrm{T}} & P_{\mathcal{A}}^{\mathrm{NW}} & P_{\mathcal{B}}^{\mathrm{T}} & P_{\mathcal{B}}^{\mathrm{NW}} & Q_{\mathcal{A}}^{\mathrm{T}} & Q_{\mathcal{A}}^{\mathrm{NW}} & Q_{\mathcal{B}}^{\mathrm{T}} & Q_{\mathcal{B}}^{\mathrm{NW}} \end{bmatrix}^{\mathrm{T}} \tag{7.5}$$

where $(\cdot)^{\mathrm{T}}$ denotes the transpose. In (7.5), $P_{\mathcal{A}} \in \mathbb{R}^{N-1}$ is a column vector consisting of elements $P_{\mathcal{A}}^{l}$ for $l = 1, \ldots, N-1$, where $P_{\mathcal{A}}^{l}$ is the active power demand of load type l aggregated over the nodes in $\mathcal{A}$, i.e., $P_{\mathcal{A}}^{l} = \sum_{k \in \mathcal{N}_{0\backslash n}} P_k^l$. Also in (7.5), $Q_{\mathcal{A}} \in \mathbb{R}^{N-1}$ is a column vector consisting of the reactive power demand of the load types aggregated over $\mathcal{A}$, and $P_{\mathcal{A}}^{\mathrm{NW}} \in \mathbb{R}^1$ and $Q_{\mathcal{A}}^{\mathrm{NW}} \in \mathbb{R}^1$ are the active and reactive NW demand for $\mathcal{A}$, respectively. The quantities $P_{\mathcal{B}}$, $Q_{\mathcal{B}}$, $P_{\mathcal{B}}^{\mathrm{NW}}$ and $Q_{\mathcal{B}}^{\mathrm{NW}}$ are defined similarly for $\mathcal{B}$.

Given this θ, the objective in each of the subsections below is to determine the elements within the output matrix of the output equation for each of the four measurement types and to determine the corresponding measurement noise covariance R.

The output equation containing the general output matrix is

$$y = \begin{bmatrix} c_N & d_N & e_N & f_N \end{bmatrix} \theta \tag{7.6}$$

where y depends on the measurement considered within each subsection. The quantities c_N, d_N, e_N, and f_N are N-element row vectors where each element in the vector is a scalar c, d, e, and f, respectively. In the following subsections, we also discuss when θ can be defined using aggregations over the entire feeder, when aggregations over additional portions of the feeder are needed, and how to define the portions of the feeder in θ to accommodate all available measurements.

7.2.3.1 Active and Reactive Power Flow Measurements

We develop output equations for measurements of the active and reactive power flow out of a node by manipulating the DistFlow power flow equations from [133] into sums of the $N-1$ load types and the NW demand over a portion of the feeder. The manipulated equations are then used to define the scalars in (7.6) both for power flow measurements at a node within the feeder and for measurements at the feeder head within the substation. We then describe how to construct the state and output matrices for an arbitrary combination of power flow measurements.

The DistFlow equations compute Q_{mn}, the reactive power flowing out of node m towards node n, as

$$Q_{mn} = \sum_{k:(n,k)\in\mathcal{E}} Q_{nk} + \sum_{l=1}^{N-1} Q_n^l + Q_n^{\text{cap}} + x_{mn} i_{mn}^2, \tag{7.7}$$

where the first term is the power flowing out/downstream of node n, the second term is the reactive load at node n, the third term Q_n^{cap} is the reactive power injection of a capacitor bank at node n, and the last term is the reactive power consumed by (m, n) where x_{mn} is the time-invariant reactance of the power line/edge, and i_{mn}^2 is the squared current magnitude flowing from node m to n. The active power flow P_{mn} is computed similarly using the active power flows and active demand components, where the active power injection of the capacitor bank is zero, and where x_{mn} is replaced by the time-invariant resistance r_{mn} to compute the active power losses.

We manipulate (7.7) and the corresponding active power flow equation into sums of the $N-1$ load types and the NW demand over a set of nodes and edges. Ref. [134] approximates P_{mn} and Q_{mn} as sums of the net load (i.e., demand minus generation) at each node in $\mathcal{N}_n$. We extend this to include losses on the downstream edges, and we modify the equation into sums over the N demand components to relate elements of θ to the power flow measurements. The resulting reactive power flow equation is

$$Q_{mn} = \sum_{l=1}^{N-1} \sum_{k \in \mathcal{N}_n} Q_k^l + \sum_{k \in \mathcal{N}_n} Q_k^{\text{cap}} + \sum_{k \in \mathcal{E}_{n+}} x_k i_k^2 \tag{7.8}$$

$$= \sum_{l=1}^{N} Q_{\mathcal{B}}^l, \tag{7.9}$$

where $Q_{\mathcal{B}}^N = Q_{\mathcal{B}}^{\text{NW}} = \sum_{k \in \mathcal{N}_n} Q_k^{\text{cap}} + \sum_{k \in \mathcal{E}_{n+}} x_k i_k^2$. The active power flow P_{mn} is computed similar to (7.9) using the active power components (setting capacitor bank injections to zero).

Using this formulation of the active and reactive power flows, we parameterize the output matrix in (7.6) for power flow measurements at two separate locations: at a point within the distribution feeder, and at the substation. The first location measures the active and reactive power flow from a node m towards node n, where n corresponds to the node used to define the set of nodes and edges within $\mathcal{B}$. Based on (7.9), Q_{mn} is the sum of N reactive demand components aggregated over $\mathcal{B}$, and so the scalars c, d, and e are zero, and f is one. Similarly, P_{mn} is the sum of the active demand component aggregated over $\mathcal{B}$. In this case the scalars c, e, and f are zero, and d is one.

The second location measures the active and reactive power flow at the feeder head in the substation, i.e., it measures the total active/reactive demand of the distribution feeder. The total active/reactive demand of the feeder consists of the active/reactive demand of the portion of the feeder $\mathcal{B}$ plus the active/reactive demand of the remaining nodes and edges on the feeder, i.e., of $\mathcal{A}$. As a result, for active power flow measurements at the feeder head, c and d are one, and e and f are zero, and vice versa for reactive power measurements.

If power flow measurements are available at multiple points within the feeder outside of the substation, rather than at a single point within the feeder as discussed above, then θ must be defined using additional portions of the feeder that form a partition of the feeder; the output matrix can then be constructed for each measurement based on θ. To define θ for an arbitrary set of power flow measurements, start with the measurement that is furthest downstream and define a portion of the feeder as the nodes and edges within the power flow equation, e.g., (7.8), at the point of the measurement. Then, take the next furthest downstream measurement, and again form the set of nodes and edges needed to compute the power flow; if this new set of nodes and edges does not contain a previously defined portion of the feeder, then define the set as a new portion of the feeder; if the set contains a previously defined portion of the feeder, then define the new portion of the feeder such that the existing portion and the new portion together form the necessary set of nodes and edges. Continue taking the next furthest downstream measurement; construct the set of nodes and edges needed to compute the power flow at the point of the measurement; define the new portion of feeder such that it, possibly together with some previously defined portions of the feeder, forms the set of nodes and edges needed to compute the power flow. Once this has been done for all of the measurements, construct the output matrix for each measurement such that the matrix sums the demand components over the portions of the feeder that are needed to compute the power flow at the point of the measurement.

We assume that power flow measurements are accurate, and so we set the measurement noise covariance for power flow measurements as $R^{\mathrm{PQ}} \approx 0$.

7.2.3.2 Squared Voltage Magnitude Difference Measurements

We develop an output equation for measurements of the difference in squared voltage magnitude between two nodes that may be connected via intermediate nodes. We first summarize the linear Disflow equation approximating the difference in squared voltage magnitude between two ends of an edge, and we relate this to θ. We then extend this to compute the difference in squared voltage magnitudes between two nodes, separated first by a single node and then by an arbitrary sequence of nodes, and we describe how to construct θ in both cases. Finally, we describe the construction of the output matrix for the general case and define the parameters in (7.6) for the original definition of θ. While [134] also computes the difference over an arbitrary sequence of nodes, they consider only the impact of the net load at each node and do not consider the impact of the losses. In contrast, we include the impact of losses, and we formulate the equations in terms of sums of load components over portions of the network.

The linear DistFlow equation approximating the difference in squared voltage magnitudes over (m, n) is

$$\begin{aligned} V_{mn}^2 \approx& 2\left(r_{mn}P_{mn} + x_{mn}Q_{mn}\right) && (7.10) \\ =& 2r_{mn}\sum_{l=1}^{N} P_{\mathcal{B}}^l + 2x_{mn}\sum_{l=1}^{N} Q_{\mathcal{B}}^l && (7.11) \end{aligned}$$

where $V_{mn}^2 = v_m^2 - v_n^2$ and v_m^2 is the squared voltage magnitude at node m. In (7.11), we express the power flows using (7.9), as sums of demand components within θ.

We now extend (7.10) for the difference in squared voltage of two nodes separated by an intermediate node and relate the new equation to θ. Consider the sequence of nodes 0, m, n that are successively connected via edges, and where 0 corresponds to a node within the substation. The difference in squared voltage magnitude between nodes 0 and n is

$$\begin{aligned} V_{0n}^2 =& 2(r_{0m}P_{0m} + r_{mn}P_{mn}) + 2(x_{0m}Q_{0m} + x_{mn}Q_{mn}) \\ =& 2\left(r_{0m}\left[P_{0m} - P_{mn} + P_{mn}\right] + r_{mn}P_{mn}\right) + && (7.12) \\ & 2\left(x_{0m}\left[Q_{0m} - Q_{mn} + Q_{mn}\right] + x_{mn}Q_{mn}\right) \\ =& 2\left(r_{0m}\left[P_{0m} - P_{mn}\right] + (r_{0m} + r_{mn})P_{mn}\right) + && (7.13) \\ & 2\left(x_{0m}\left[Q_{0m} - Q_{mn}\right] + (x_{0m} + x_{mn})Q_{mn}\right) \\ =& 2r_{0m}\sum_{l=1}^{N} P_{\mathcal{A}}^l + 2(r_{0m} + r_{mn})\sum_{l=1}^{N} P_{\mathcal{B}}^l + && (7.14) \\ & 2x_{0m}\sum_{l=1}^{N} Q_{\mathcal{A}}^l + 2(x_{0m} + x_{mn})\sum_{l=1}^{N} Q_{\mathcal{B}}^l. \end{aligned}$$

Above, we first use $V_{0n}^2 = V_{0m}^2 + V_{mn}^2$, we add and subtract the downstream power flow terms in (7.12), we rearrange terms in (7.13) to collect the resistances and reactances for the power flows, and we use (7.8) and the corresponding active power flow equation in (7.14) to express the power flows as sums of the demand components over portions of the feeder. We used the property that $\sum_{l=1}^{N} Q_{\mathcal{A}}^l = Q_{0m} - Q_{mn}$ by expressing both power flow terms on the right-hand side according to (7.8), then by noting that the terms left after subtracting Q_{mn} were a summation over the set of nodes and edges in $\mathcal{A}$. Because 0 and n correspond to the nodes used to define $\mathcal{A}$ and $\mathcal{B}$, and because no other power flow terms appeared in (7.13), we were able to define the power flows in terms of the portions of the feeder used in the definition of θ.

We now extend this to consider the difference in squared voltage magnitudes over an arbitrary sequence of nodes $1, \ldots, n$, and we describe the corresponding construction of θ. The above process can be extended to an arbitrary sequence of nodes by 1) expressing V_{1n}^2 in terms of the resistances, reactances, and power flows on each of the edges between 1 and n, 2) by looking at each power flow term and adding and subtracting power flows that appear within the equation and that are downstream from the power flow of interest, and 3) by rearranging terms to collect resistances and reactances for the power flows and power flow differences. The properties $P_{nm} = -P_{mn}$, $Q_{nm} = -Q_{mn}$, $r_{nm} = r_{mn}$, and $x_{nm} = x_{mn}$ can be used if quantities appear in the opposite direction that they were defined, i.e., if the edge is defined as (m, n) not (n, m). A portion of the feeder should be defined for each set of nodes and edges needed to compute each power flow term and each power flow difference term that appears in the manipulated equation for V_{1n}^2. Additional portions of the feeder should be defined, if needed, to form a partition of the feeder. The state θ should be constructed to include the N demand components for each of the feeder portions defined.

We construct the output matrix for V_{0n}^2 as the state for this corresponds to the original definition of θ, and then we describe the construction of an output matrix for the state resulting from the measurements across the arbitrary sequence of nodes $1, \dots, n$. For V_{0n}^2 we formulate an output equation as $V_{0n}^2 = C^{\mathrm{VM}}\theta$ where we must define the entries in C^{VM}. In (7.14), a resistance/reactance or a sum of resistances/reactances appears before each summation of load components over each of the portions of the network in θ. The scalars in C^{VM} defined in (7.6) would be set equal to these resistances/reactances and their sums if the network parameters, i.e., the resistances and reactances, were known. However, we assume that we only know the feeder topology and not the network parameters. As a result, we identify the parameters c, d, e, and f within C^{VM} via multiple linear regression using historical measurements of V_{0n}^2 and θ, where historical values of θ are estimated using the historical availability of both smart meter measurements and substation power flow measurements, where Section 7.2.3.4 details the process of estimating θ.

In a similar vein, once the state is constructed for measurements across the arbitrary sequence of nodes $1, \dots, n$, the parameters for the output matrix can be defined as the resistances/ reactances that appear as coefficients to the summations over the portions of the network within the state, or they can be identified if the resistances/reactances are not known. Future work should compare the disaggregation accuracy when the resistances and reactances within C^{VM} are known versus when they are identified.

We compute the measurement noise covariance R^{VM} as the covariance of the errors between the historical measurements of V_{0n}^2 and the values computed according to

$C^{\mathrm{VM}}\theta$. The measurement noise arises from the approximations used to form (7.10), and from the errors in estimating both C^{VM} and θ.

7.2.3.3 Voltage Angle Difference Measurements

We develop an output equation for measurements of the difference in voltage angles between two nodes separated by an intermediate node. We also describe how to develop an output equation for measurements of the difference in voltage angles over an arbitrary sequence of nodes. We start from the linear approximation for the voltage angle difference used in [134]. Following this, we use similarities between the voltage magnitude and voltage angle difference equations to describe the equation for the voltage angle difference over an arbitrary sequence of nodes. Finally, we describe the output matrix and measurement noise covariance for this measurement type.

The linear equation approximating δ_{mn}, the voltage angle difference between nodes m and n across (m, n), is

$$\delta_{mn} \approx x_{mn}P_{mn} - r_{mn}Q_{mn}. \tag{7.15}$$

Note that this has a similar form to (7.10), except that the resistance and reactance have been swapped and the resistance is negated. As a result, the equations generalizing the change in voltage angle over a sequence of nodes have the same form as those of the difference in squared voltage magnitude, but the resistances and reactances are swapped and the resistances are negated. We assume that voltage angle measurements are available at the same nodes as voltage magnitude measurements, and so no modifications to the state are needed to incorporate voltage angle differences as the same power flow terms (and resulting portions of the feeder) appear in both the voltage magnitude and angle equations.

The output equation for δ_{0n} is

$$\delta_{0n} = \frac{1}{2}\begin{bmatrix} e_N^{\mathrm{VM}} & f_N^{\mathrm{VM}} & -c_N^{\mathrm{VM}} & -d_N^{\mathrm{VM}} \end{bmatrix}\theta \tag{7.16}$$

where c_N^{VM}, d_N^{VM}, e_N^{VM}, and f_N^{VM} are the parameters within C^{VM}, which are identified via multiple linear regression. Future work should compare the disaggregation performance when using the identified parameters versus the actual network parameters, and it should also investigate identification of the matrix parameters using both voltage angle and voltage magnitude data (rather than only using voltage magnitude data).

The measurement noise covariance for voltage angle measurements, R^{VA}, is computed using the historical accuracy of the measured angle difference versus the predicted angle difference given historical estimates of the state vector θ, the computation of which are detailed in Section 7.2.3.4, and the corresponding output matrix. The measurement noise arises from the approximations used to form (7.16), and from the errors in estimating both the output matrix parameters and historical state values.

7.2.3.4 Smart Meter Measurements

We develop an output equation for real-time smart meter measurements. Real-time smart meter measurements along with substation power flow measurements allow the estimation of real-time active and reactive power measurements for the N demand components, where we have assumed that the smart meter data for a residence is disaggregated into the demand of the $N-1$ load types. If θ contains aggregations over portions of the network, then power measurements for $N-1$ load types can be computed by summing the disaggregated household data that pertains to the nodes within each portion of the network. As a result, θ does not need to be modified to incorporate real-time smart meter measurements.

We assume that measurements of the NW demand are not available for portions of the network as this would require power flow measurements at each end of each line/edge and metering of the power injections of the capacitor banks. As a result, we assume that the total NW demand of the feeder can be divided into the NW demand of the portions of the feeder based on the number of edges within the portion of the feeder (for a given phase). For example, $P_{\mathcal{A}}^{\mathrm{NW}} = P^{\mathrm{NW}}|\mathcal{E}_{0\backslash n+}|\left(|\mathcal{E}|\right)^{-1}$ and $P_{\mathcal{B}}^{\mathrm{NW}} = P^{\mathrm{NW}}|\mathcal{E}_{n+}|\left(|\mathcal{E}|\right)^{-1}$, where P^{NW} is the total NW demand of the feeder and where $|\cdot|$ is the number of elements in the argument set. Note that the NW demand contains the power injections of the capacitor banks, which introduces inaccuracies in the approximation. Also, in reality, the NW demand should be divided for the portions of the feeder as a function of the resistances and reactances of each line and the line length within the portion of the feeder.

Without adhering to the general output matrix structure in (7.6), the resulting output equation for real-time smart meter measurements is

$$y = \begin{bmatrix} I \end{bmatrix} \theta, \tag{7.17}$$

where I is an appropriately-sized identity matrix. Inaccuracies arise in the measurement from errors in disaggregating the smart meter data, but we assume this is negligible. As a result, we assume that the measurements are accurate, and so the measurement noise covariance for smart meter measurements is set as $R^{\text{SM}} \approx 0$. While there are known inaccuracies in the approximations of measurements of the NW demand for portions of the feeder, these inaccuracies cannot be reflected into the covariance as we assume no data is available to quantify the inaccuracy.

Future work should investigate improving the estimation of the NW demand for portions of the feeder. This could be done 1) by investigating whether a capacitor bank connecting/disconnecting to the feeder can be detected from the substation, 2) by investigating whether knowledge of the amount and location of power injected by the capacitor bank improves the estimation of the NW demand for portions of the feeder, and 3) by investigating methods of more accurately approximating active and reactive losses using line lengths and possibly line parameters. These three topics could improve disaggregation accuracy by improving the accuracy of the state measurements derived from real-time smart meter measurements as well as the accuracy of the identified parameters in the voltage magnitude and angle output matrices.

Throughout the subsections for the various measurement types we have described how to define θ based on the portions of the feeder used to compute the measured quantities. If measurements are only available at the substation, then θ does not need to include demand components for portions of the network, and we can relax the assumption that the feeder topology is known. Alternatively, if different types of measurements are available at different nodes, then it is necessary to define portions of the feeder that form a partition such that the portions can be combined into the set of nodes and edges used to compute the power flows that appear within the equations for the different measurements.

7.3 Aggregate Models for Energy Disaggregation

We create separate aggregate load models for the N demand components using multiple linear regression with historical smart meter and feeder measurements. The aggregate models are formed for each phase individually, and we do not include notation specifying the phase here for simplicity.

The regression models for a given phase are

$$\widehat{P}_t^l = \beta^l D_t^l \qquad l = 1, \ldots, N \tag{7.18}$$

$$\widehat{Q}_t^l = \gamma^l D_t^l \qquad l = 1, \ldots, N \tag{7.19}$$

where $\widehat{P}_t^l$ is the predicted active demand for load component l, $\widehat{Q}_t^l$ is the predicted reactive demand for l, β^l and γ^l are row vectors of regression parameters for l. The column vector D_t^l consists of the input features for l at time t:

$$D_t^l \triangleq \begin{bmatrix} D_t^{l,\mathrm{CAL}} & D_t^{l,\mathrm{WEA}} & D_t^{l,\mathrm{NW}} \end{bmatrix}^{\mathrm{T}},$$

where $D_t^{l,\mathrm{CAL}}$, $D_t^{l,\mathrm{WEA}}$, and $D_t^{l,\mathrm{NW}}$ are row vectors of calendar-based features, weather-based features, and features based on substation and network measurements, respectively.

Aggregate models for each of the N load component can be created for the entire feeder, or for portions of the feeder if the feeder topology is known. Historical data for the N load components are needed to compute the models, and we assume that historical values of the input features are available. To compute historical aggregate data of the $N-1$ load types, we assume that the historical smart meter data for each residence can first be disaggregated into the demand of the $N-1$ load types. If the feeder topology is known, the residence-level data for the $N-1$ load types can be summed over the set of residences connected to each node to compute aggregate historical nodal demand data for the $N-1$ load types. The aggregate nodal data can be summed over the nodes within each feeder portion to compute the aggregate demand of the load types for each portion of the feeder. If the feeder topology is not known, then historical data of the aggregate demand for the $N-1$ load types can be computed for the entire feeder by summing the disaggregated smart meter data for each load type over all of the residences.

Aggregate NW demand data, both active and reactive, can be computed for the entire feeder by subtracting the total smart meter demand from the substation measurements of the active and reactive power flow into the feeder. If the feeder topology is known, the total NW demand can be estimated for the portions of the feeder based on the number of lines/edges in that portion of the feeder (for the phase of interest) as in Section 7.2.3.4.

7.4 Case Studies

In this section, we perform case studies to evaluate the performance of the dissagregation algorithm using a variety of scenarios of real-time measurement availability and to evaluate the prediction accuracy of the models within the disaggregation algorithm when substation measurements are and are not included into the models. In these case studies, the power system entity disaggregates the total demand of a distribution feeder into three components: 1) the aggregate AC demand, 2) the aggregate non-air conditioning demand, or the OL demand (where OL stands for other load), and 3) the demand of the distribution feeder equipment, referred to as the network (NW) demand, which includes the losses and the power injections of the capacitor banks.

Subsection 7.4.1 details the plant used within the case studies, which includes a feeder model and load models that have active and reactive demand components. The time-step of the case studies is one minute as this is the frequency with which load data is available for the plant. Subsection 7.4.2 provides the details of the disaggregation algorithm. Subsection 7.4.3 details two sets of aggregate models of the three demand components used within the disaggregation algorithm, where the sets differ in the input features used. Subsection 7.4.4 details different cases, which are defined based on the real-time measurement availability and the aggregate models used within the disaggregation algorithm. Subsection 7.4.5 describes the data used in evaluating the disaggregation algorithm, and it defines the performance metric used to evaluate the disaggregation algorithm. Finally, Subsection 7.4.6 summarizes the results.

7.4.1 Plant Construction

The plant incorporates a feeder model, real-world weather data, real-world active power demand data for the loads connected to the feeder, and voltage-dependent load models with active and reactive power components. The feeder model is the IEEE 13-bus test feeder [135]. Figure 7.4 depicts the single-line diagram of the feeder and indicates the assumed boundaries of the substation. We use a three-phase, unbalanced feeder model, where we assume that all loads are wye-connected, we assume that the switch within the feeder is closed, we assume the data for the line from node 671 to 692 is the same as the line from node 692 to 675, and we assume that there is a load at node 680, which we set to $(290 + 212.0j)$, $(170 + 80.0j)$, and $(128 + 86.0j)$ kW for phases A, B, and C, respectively. We assign half of the distributed load to node 632 and half to node 671. The feeder contains a voltage regulator for each phase as well as capacitor banks, which we assume are always connected. We assume a lockout time of five minutes for tap changes on each phase of the voltage regulator.

Real-world outdoor temperature data and real-world data for the active power demand of residences and the devices/loads within those residences are taken from [136]. All data from [136] corresponds to Austin, TX. We linearly interpolate the hourly weather data down to the time-scale of the demand data of the loads, i.e., one minute time-steps. We connect houses to the phases at each node by randomly sampling with replacement from the set of available houses in [136]. Houses are drawn for each phase at a node until the total average active power demand over the course of August 31, 2017 is greater than the active demand for the phase load of the feeder model. We incorporate the real-world data of the loads' active power demand into load models to determine the voltage-dependent active demand as well as the voltage-dependent reactive power demand of the loads.

We use a "performance model" based on [137,138] to compute the reactive power draw of the ACs connected to each phase at each node, and we use ZIP model parameters from [139, 140] to compute the active and reactive power demand of the OLs connected to each phase at each node. Note that the following discussion pertains to each phase at node n, but we do not include notation specifying the phase for simplicity.

The performance model captures the voltage and temperature dependence of the reactive power demand of the ACs, where the active power model is approximately a constant power model that is unaffected by voltage changes. To implement the performance model, we first sum the active power demand of the ACs connected to a given phase at node n: $P_t^{n,\mathrm{AC}} = \sum_{j \in \mathcal{J}^{n,\mathrm{AC}}} P_t^{j,\mathrm{AC}}$, where $P_t^{n,\mathrm{AC}}$ is the total active power of ACs connected to node n, $P_t^{j,\mathrm{AC}}$ is the active power demand of AC j, $\mathcal{J}^{n,\mathrm{AC}}$ is the set of ACs connected to node n. Note that the real-world data used to calculate $P_t^{n,\mathrm{AC}}$ implicitly depends on outdoor temperature. Then, we use the outdoor temperature, the voltage at node n, and $P_t^{n,\mathrm{AC}}$ to compute the reactive power of the air conditioners at node n, $Q_t^{n,\mathrm{AC}}$, according to the following [137]:

$$Q_t^{n,\mathrm{AC}} = b_1 + \frac{b_2}{(v_t^n - b_0)} + b_3(v_t^n - b_0) + b_4(v_t^n - b_0) \tag{7.20}$$

where b_0, b_2, b_3, and b_4 are parameters that depend on the outdoor temperature, b_1 is a parameter the depends on $P_t^{n,\mathrm{AC}}$, and where v_t^n is the voltage of node n. We compute the parameters in (7.20) based on the curves of Fig. 16 in [138].

To determine $P_t^{n,\mathrm{OL}}$ and $Q_t^{n,\mathrm{OL}}$, which are the active and reactive power of the OL demand connected to a given phase at node n, we use active power data for these loads from [136], and we use voltage-dependent ZIP models. The ZIP model

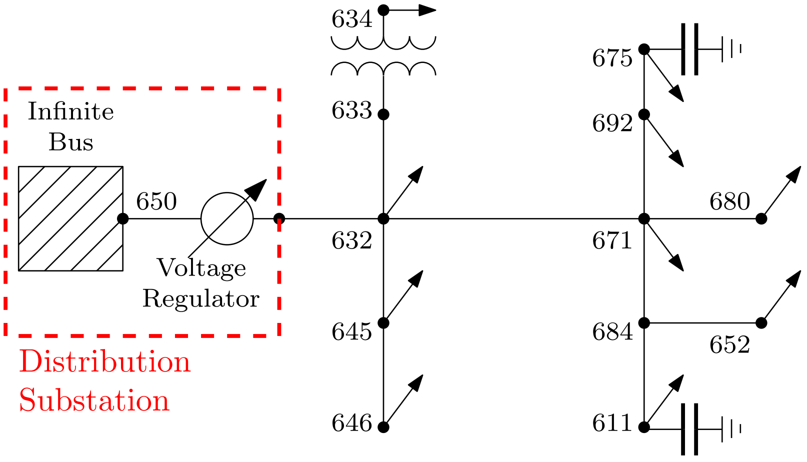


Figure 7.4: Single-line diagram of the IEEE 13-bus test feeder where the infinite bus corresponds to the distribution substation.

Table 7.2: Summary of ZIP Model Parameters for Different Load Types

Load Type	ZIP Parameters							Reference
	Q_0^j	Z_p^j	I_p^j	P_p^j	Z_q^j	I_q^j	P_q^j	
Coffee Maker	13.32	0.13	1.62	-0.75	3.89	-6.0	3.11	[140]
Drier Heater	0	0.96	0.05	-0.01	0	0	0	[139]
Drier Tumbler	459.91	1.91	-2.23	1.33	2.51	-2.34	0.83	[139]
Fan	83.28	-.47	1.71	-0.24	2.34	-3.12	1.78	[140]
Freezer	558.86	1.19	-0.26	0.07	0.59	0.65	-0.24	[139]
Heat Pump	426.37	0.72	-0.98	1.25	14.78	-23.72	9.93	[139]
Lighting	0.85	0.47	0.63	-0.1	0.55	0.38	0.07	[140]
Microwave	451.02	1.39	-1.96	1.57	50.07	-93.55	44.48	[140]
Miscellaneous	73.84	1.18	-1.64	1.47	8.29	-13.67	6.38	[140]
Office Equipment	0	0.34	-0.32	0.98	0	0	0	[139]
Refrigerator	52.47	1.17	-1.83	1.66	7.07	-10.94	4.87	[140]
Washing Machine	518.23	0.05	0.31	0.63	-0.56	2.2	-0.65	[139]
Water Heater	1.46	0.64	0.59	-0.23	0.13	0.75	0.12	[140]

from [139] for load $j \in \mathcal{J}^{n,\text{OL}}$ is

$$P_t^{j,\text{OL}} = P_0^j \left[Z_\text{p}^j \left(\frac{v_t^n}{v_0^j} \right)^2 + I_\text{p}^j \left(\frac{v_t^n}{v_0^j} \right) + P_\text{p}^j \right] \tag{7.21}$$

$$Q_t^{j,\text{OL}} = Q_0^j \left[Z_\text{q}^j \left(\frac{v_t^n}{v_0^j} \right)^2 + I_\text{q}^j \left(\frac{v_t^n}{v_0^j} \right) + P_\text{q}^j \right] \tag{7.22}$$

where $\mathcal{J}^{n,\text{OL}}$ is the set of OL loads connected to node n, where P_0^j, Z_p^j, I_p^j, P_p^j, and V_0^j are the ZIP parameters for the active power demand of load j, and where similar parameters are defined for $Q_t^{j,OL}$. The total OL active power demand at node n is $P_t^{n,\text{OL}} = \sum_{j \in \mathcal{J}^{n,\text{OL}}} P_t^{j,\text{OL}}$, and $Q_t^{n,\text{OL}}$ is computed similarly.

Table 7.2 summarizes the ZIP parameters. Note that the ZIP models include a parameter P_0^j for the nominal active power demand of load j; we set this parameter to the measured demand for load j at time t from [136], and so we do not include P_0^j in the table. The remaining ZIP parameters are taken from [139, 140] depending on the type of load. The lighting loads are assumed to be incandescent. Water heaters are assumed to be resistive heaters. Furnace/heater loads are assumed to correspond to heat pumps. Data from kitchen plugs for small appliances are assumed to correspond to a coffee maker. Miscellaneous loads that do not fall into one of the specific categories are summed into a household-level ZIP model whose parameters are also given in the "Miscellaneous" entry in Table 7.2. We assume that the rated voltage v_0^j for all ZIP models is 120 V.

At each time-step, the simulation computes the complex voltages and complex currents within the feeder using the previously determined assignments of loads to nodes and phases, the active power demand data and outdoor temperature data, the load models, and the feeder model. The residence connections are used to assign the loads to each phase at each node. The quantity $P_t^{n,\mathrm{AC}}$ is then calculated for each phase at each node, which is then used along with the outdoor temperature and the voltage at the node to determine $Q_t^{n,\mathrm{AC}}$. The ZIP models of the OL loads, the voltage of the respective nodes, and the active power demand data are used to compute $P_t^{n,\mathrm{OL}}$ and $Q_t^{n,\mathrm{OL}}$ at each node.

At each time-step in the simulation, the voltages at the nodes are initially assumed to be at their rated values, then a forward-backward sweep algorithm from [141] iterates to compute the complex voltages and complex currents within the feeder. During each iteration, the voltages within the performance and ZIP models are updated to compute the new value of the voltage-dependent load. We check the forward-backward sweep algorithm for convergence at node 671, where the algorithm is said to converge if the maximum change in voltage at the node is less than 1.0×10^{-6} in per-unit voltage. Once the algorithm converges at a given time-step, the tap positions of the voltage regulator are updated by checking the voltage magnitude at node 671, where the allowable voltage bandwidth is 121-123 V (for a 120 V base). If the tap position changes at a given time-step, the forward-backward sweep algorithm is restarted with the new tap positions and the process repeats. If the tap position does not change, the plant simulation advances to the next time-step.

7.4.2 Disaggregation Algorithm Details

This subsection provides the details of the disaggregation algorithm used within the case studies. We first define the state vector θ_t, then we select the user-defined functions in (7.1) to formulate the closed-form update. Finally, we summarize the parameter settings in the algorithm and detail the tuning process of these parameters.

The state vector θ_t is defined according to (7.5) with $\mathcal{B} = \{\mathcal{N}_n, \mathcal{E}_{n+}\}$ and $\mathcal{A} = \{\mathcal{N}_{0\backslash n}, \mathcal{E}_{0\backslash n+}\}$. We set node n to node 671 in Fig. 7.4, and we set 0 to the secondary side of the voltage regulator. As we have two load types in the disaggregation formulation of the case studies, $P_{\mathcal{A}}$, $P_{\mathcal{B}}$, $Q_{\mathcal{A}}$, and $Q_{\mathcal{B}}$ each contain two demand components, one for the AC demand and one for the OL demand in the respective portions of the feeder.

To formulate the closed-form update of (7.1), we set the Bregman divergence as $D(\theta\|\widehat{\kappa}_t) = \frac{1}{2}\|\widehat{\Pi}_t^{-\frac{1}{2}}(\theta - \widehat{\kappa}_t)\|_2^2$, and we choose the loss function to be $\ell_t(\widehat{\theta}_t) = \frac{1}{2}\|(\Pi_t^{\mathrm{y}})^{-\frac{1}{2}}(C_t\widehat{\theta}_t - y_t)\|_2^2$, where $\widehat{\Pi}_t$ is a user-defined, positive semi-definite matrix for the estimation error covariance, and where Π_t^{y} is a user-defined, positive definite matrix for the covariance of the output estimation error. Given these functions, the closed form solution of (7.1) is the following:

$$\widehat{\kappa}_{t+1} = \widehat{\kappa}_t + \eta^{\mathrm{s}}\widehat{\Pi}_t C_t^T (\Pi_t^{\mathrm{y}})^{-1}\left(y_t - C_t\widehat{\theta}_t\right). \tag{7.23}$$

We compute $\widehat{\Pi}_t$ from the historical covariance of the prediction errors of the aggregate models used within the disaggregation algorithm, where the prediction errors pertain to the historical data from August 14-30, 2017. We compute the covariance Π_t^{y} according to Section 7.2.2.

We tune the parameters within (7.23) by applying the disaggregation algorithm to the simulated plant using data for August 31,2017. We set $\eta^{\mathrm{s}} = 0.5$ when using model set $\mathcal{M}^{\mathrm{nc}}$ within the algorithm, and we set $\eta^{\mathrm{s}} = 0.2$ when using model set $\mathcal{M}^{\mathrm{c}}$. Note that these parameters were roughly tuned to optimize algorithm performance for the given day; additional tuning may result in performance gains, but we do not believe they would influence the results significantly. We set the measurement noise covariances R^{PQ} and R^{SM} to σI, where I is an appropriately-sized identity matrix in each case, and where $\sigma = 1 \times 10^{-8}$. Measurement noise covariances R^{VM} and R^{VA} are computed using the historical data from August 14-30, 2017.

7.4.3 Aggregate Model Details for the Disaggregation Algorithm

We define two sets of aggregate models used in the disaggregation algorithm that each contain separate models for the AC, OL, and NW demand. The state contains two portions of the feeder, and so two models for each demand type are computed in each set. We train the models, i.e., compute the regression parameters, using historical data obtained by simulating the plant with load and temperature data from August 14-30, 2017.

The first set of models include input features based on the complex current flowing into the feeder measured at the substation, and we denote the set of models as $\mathcal{M}^{\mathrm{c}}$. The vector $D_t^{\mathrm{AC}} = \begin{bmatrix} x_t^{\mathrm{tod}} & |i_t^0| & \mathrm{real}(i_t^0) & \mathrm{imag}(i_t^0) & T_t^{\mathrm{o}} \end{bmatrix}^{\mathrm{T}}$ are the input features for the AC demand models, where x_t^{tod} is an indicator vector indicating the time of day, where $|\cdot|$, $\mathrm{real}(\cdot)$, and $\mathrm{imag}(\cdot)$ are the magnitude, real component, and imaginary component of the argument, respectively, where i_t^0 is the complex current measured at the distribution substation, and where T_t^{o} is the outdoor temperature. The input features for the OL demand models are $D_t^{\mathrm{OL}} = \begin{bmatrix} x_t^{\mathrm{tod}} & |i_t^0| & \mathrm{real}(i_t^0) & \mathrm{imag}(i_t^0) \end{bmatrix}^{\mathrm{T}}$. The input features for the NW demand models are the same as for the OL demand, i.e., $D_t^{\mathrm{NW}} = D_t^{\mathrm{OL}}$.

The second set of models do not include input features based on the complex current measurements, and we denote the set of models as $\mathcal{M}^{\mathrm{nc}}$. In this case, the input features for the AC, OL, and NW demand models are $D_t^{\mathrm{AC}} = \begin{bmatrix} x_t^{\mathrm{tod}} & T_t^{\mathrm{o}} \end{bmatrix}^{\mathrm{T}}$ and $D_t^{\mathrm{NW}} = D_t^{\mathrm{OL}} = x_t^{\mathrm{tod}}$.

7.4.4 Case Definitions

To define the cases, we define five scenarios of real-time measurements from the substation and feeder, we describe notation used to indicate the additional availability of real-time smart meter measurements in a scenario, and we indicate which model sets are used within each scenario. In all scenarios, we assume that historical measurements of the active and reactive power flowing into the feeder and historical active and reactive power measurements from smart meters are available, which allows computation of the aggregate model parameters for $\mathcal{M}^{\mathrm{nc}}$ in any scenario. In addition, we assume that any measurements that are available in real-time are also available historically. For example, if real-time complex current measurements are available at the substation, historical values are available to compute $\mathcal{M}^{\mathrm{c}}$.

The measurement scenarios are the following:

- **Scenario 1:** This scenario does not include any real-time measurements from the substation, the feeder, or from smart meters. This scenario allows the use of the models in $\mathcal{M}^{\text{nc}}$ to compute predictions, but it does not allow measurement-based adjustments to the predictions within the disaggregation algorithm as there are no real-time measurements.

- **Scenario 2:** This scenario consists of real-time measurements of the complex current flowing into the feeder, measured at the substation. In this scenario, we use $\mathcal{M}^{\text{c}}$ to compute predictions as complex current measurements are available. We do not use measurement-based adjustments to the predictions in the disaggregation algorithm.

- **Scenario 3:** This scenario consists of real-time measurements of the active power flowing into the distribution feeder, measured at the substation. We use $\mathcal{M}^{\text{nc}}$ within this scenario as no real-time complex current measurements are available. We use measurement-based adjustments within the disaggregation algorithm using the real-time active power measurements.

- **Scenario 4:** This scenario consists of measurements of the active power, reactive power, and complex current flowing into the feeder, measured at the substation. We use both $\mathcal{M}^{\text{c}}$ and $\mathcal{M}^{\text{nc}}$ within this scenario. We also incorporate the active and reactive power measurements into the measurement-based adjustments of the disaggregation algorithm.

- **Scenario 5:** This scenario consists of the measurements of Scenario 4 plus voltage magnitude and voltage angle measurements at the secondary side of the voltage regulator and at node 671 within Fig. 7.4. While both model sets could be used within this measurement scenario, we only use $\mathcal{M}^{\text{c}}$. We incorporate the active and reactive power measurements as well as the voltage magnitude and angle measurements into the measurement-based adjustments in the disaggregation algorithm.

- **Smart Meter Measurements:** We indicate smart meter measurement availability in a scenario by adding a suffix to the measurement scenario that indicates the number of minutes between smart meter measurements. For example, 3-60 denotes measurement scenario 3 with smart meter measurements available every 60 minutes. If we do not incorporate smart meter measurements into the scenario, we do not add a suffix.

Scenario 3 is comparable to the measurement availability for the case studies in [96] and [86], detailed in the previous chapter, which benchmarks the results within this chapter against these prior works.

7.4.5 Performance Evaluation

We evaluate the algorithm performance by applying the disaggregation algorithm while simulating the plant using data from September 1-4, 2017. We use the RMS error (RMSE) of the AC, OL, and NW demand to evaluate the performance of the algorithm in the different cases. The RMSE for an arbitrary sequence of complex-valued predictions $\widehat{\psi}_t$ and the corresponding true values ψ_t over N^{ts} time-steps is defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N^{\text{ts}}} |\psi_t - \widehat{\psi}_t|^2}{N^{\text{ts}}}}$$

where $|\cdot|$ is the magnitude of the argument. While θ_t contains demand components for portions of the feeder, we evaluate the disaggregation algorithm's ability to estimate the total AC, OL, and NW demand, i.e., we add the estimates of the feeder portions together for each demand component and compare this to the total realized demand of that component.

7.4.6 Results

Table 7.3 summarizes the RMSE of the AC, OL, and NW demand components in each case. The RMSE values in the table are average values over the three phases, and the "models" column indicates the set of models used within the specific case study. The AC-P, AC-Q, and AC-S entries correspond to the active, reactive, and apparent power of the AC demand, and similar quantities are included for the OL and NW demand. In the table, we exclude entries for the reactive and apparent power for Scenario 3 and its variants as the disaggregation algorithm does not include reactive power in these cases. Figure 7.5 depicts time series of the actual AC, OL, and NW demand magnitude for phase A along with the corresponding aggregate model predictions for Scenario 1 and Scenario 2. Figure 7.6 depicts time series of the actual AC, OL, and NW demand magnitude for phase A along with the estimated demand components in scenario 4 using $\mathcal{M}^{\text{c}}$ and in Scenario 4-15, which also uses $\mathcal{M}^{\text{c}}$. The discussions below highlight aspects of Table 7.3 to draw comparisons between the different scenarios and to evaluate the impact different real-time measurement

Table 7.3: RMSE (kW) of the Demand Components in Different Cases

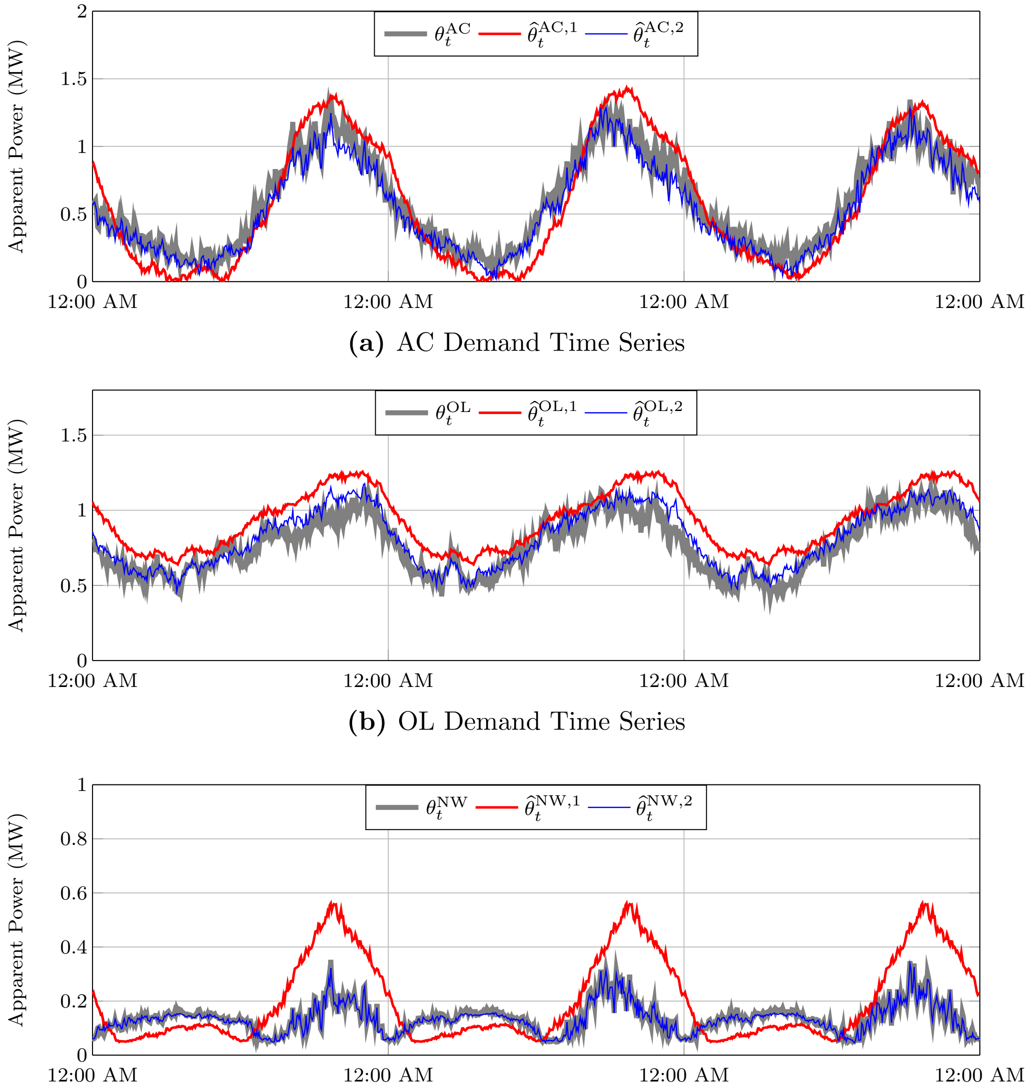
Scenario	Models	Demand Component								
		AC-P	AC-Q	AC-S	OL-P	OL-Q	OL-S	NW-P	NW-Q	NW-S
1	$\mathcal{M}^{\text{nc}}$	181.4	45.4	187.0	179.2	52.7	186.8	50.0	157.0	165.4
2	$\mathcal{M}^{\text{c}}$	64.1	16.0	66.1	68.0	20.0	70.9	5.6	8.2	9.9
3	$\mathcal{M}^{\text{nc}}$	129.6	-	-	114.0	-	-	23.0	-	-
3-60	$\mathcal{M}^{\text{nc}}$	77.5	-	-	63.0	-	-	16.8	-	-
3-30	$\mathcal{M}^{\text{nc}}$	69.6	-	-	55.7	-	-	13.8	-	-
3-15	$\mathcal{M}^{\text{nc}}$	61.1	-	-	47.0	-	-	10.9	-	-
4	$\mathcal{M}^{\text{nc}}$	103.5	25.9	106.7	97.0	33.1	102.5	8.7	21.4	23.1
4	$\mathcal{M}^{\text{c}}$	64.1	16.0	66.1	64.8	19.6	67.7	5.7	8.2	10.0
4-60	$\mathcal{M}^{\text{c}}$	46.9	11.7	48.3	48.7	15.6	51.1	5.6	7.7	9.5
4-30	$\mathcal{M}^{\text{c}}$	45.7	11.4	47.1	47.5	14.9	49.8	5.2	7.1	8.8
4-15	$\mathcal{M}^{\text{c}}$	41.2	10.3	42.4	43.1	13.4	45.2	5.0	6.4	8.1
5	$\mathcal{M}^{\text{c}}$	61.7	15.4	63.6	62.2	20.2	65.4	6.1	13.6	15.1

availability.

To evaluate the prediction accuracy for model sets $\mathcal{M}^{\text{nc}}$ and $\mathcal{M}^{\text{c}}$ we compare the RMSE of Scenarios 1 and 2. Scenario 2, which uses $\mathcal{M}^{\text{c}}$, achieves an average reduction in RMSE of 73.1% across the demand components compared to Scenario 1, which uses $\mathcal{M}^{\text{nc}}$. Specifically, the RMSE of the active power components of the AC, OL, and NW demand are reduced by 64.7%, 62.1%, and 88.8%, respectively. Figure 7.5 presents time series depicting this improvement in prediction accuracy. These results indicate that including real-time substation measurements of the complex current flowing into the distribution feeder significantly improves the prediction accuracy of the aggregate models used within the disaggregation algorithm.

To evaluate the impact of additional real-time smart meter measurements at increasing frequency of availability, we compare the RMSE of Scenarios 3, 3-60, 3-30, and 3-15. Comparing Scenario 3 and Scenario 3-60 shows that including smart meter measurements results in an average reduction in RMSE of 37.3% across the three active power demand components. The average RMSE reductions for Scenarios 3-30 and 3-15 versus Scenario 3 are 45.8% and 54.7%, respectively. These results indicate that real-time smart meter measurements can significantly improve disaggregation accuracy, and that increasing the frequency with which real-time smart meter measurements are available can further improve disaggregation accuracy.

To evaluate 1) the impact of substation reactive power measurements in addition to the active power measurements, and 2) the impact of substation complex current measurements in addition with the real and reactive power measurements, we compare Scenario 3, Scenario 4 using $\mathcal{M}^{\text{nc}}$, and Scenario 4 using $\mathcal{M}^{\text{c}}$. Scenario 4 using

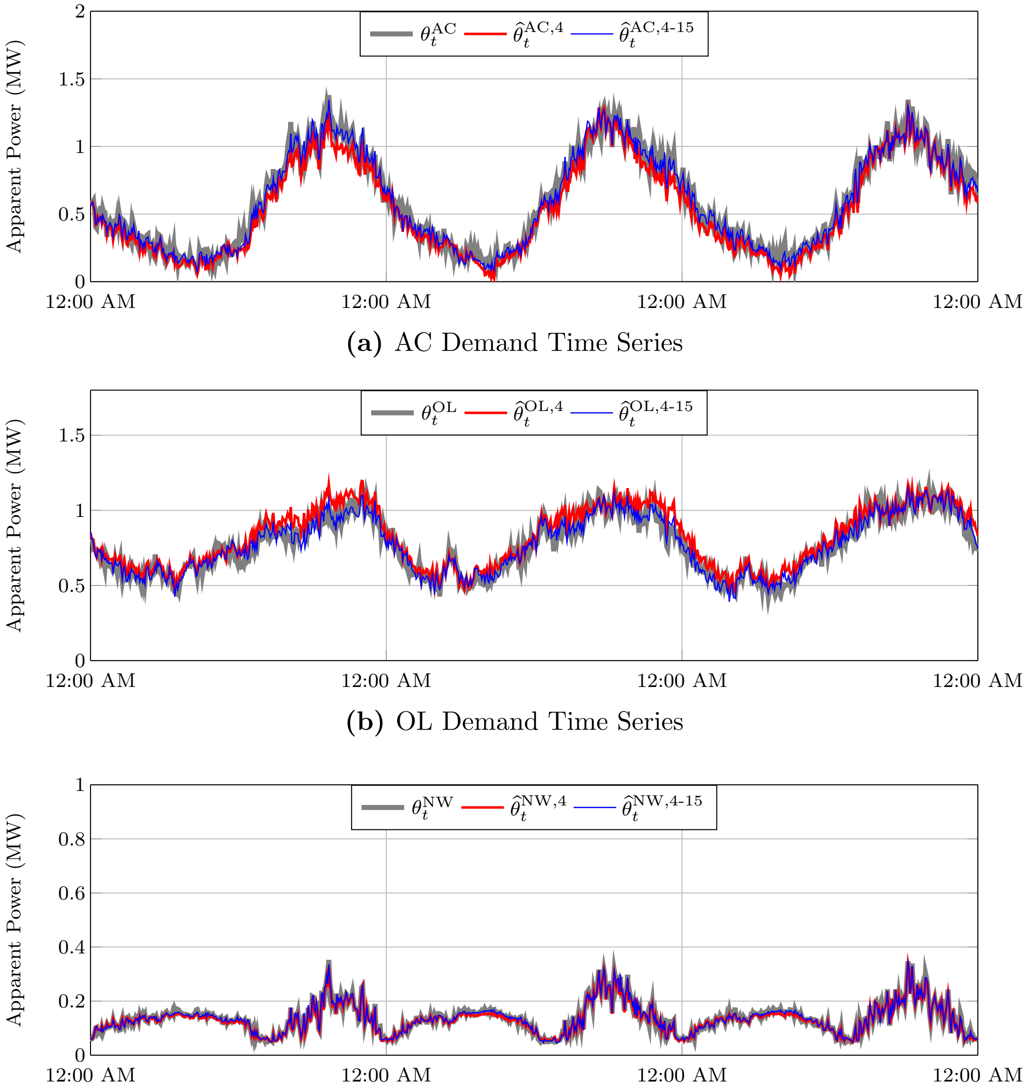


(a) AC Demand Time Series

(b) OL Demand Time Series

(c) NW Demand Time Series

Figure 7.5: Time series of the actual AC, OL, and NW demand magnitude, the magnitude of the aggregate model predictions in Scenario 1, denoted with a “1” in the superscript, and the magnitude of the aggregate model predictions in Scenario 2, denoted with a “2” in the superscript. The time series correspond to phase A.



(a) AC Demand Time Series

(b) OL Demand Time Series

(c) NW Demand Time Series

Figure 7.6: Time series of the actual AC, OL, and NW demand magnitude for phase A, the magnitude of the corresponding estimates produced in Scenario 4 using $\mathcal{M}^c$, and the magnitude of the corresponding estimates produced by Scenario 4-15. Estimates for Scenario 4 are denoted with a "4" in the superscript after the demand component, and estimates for Scenario 4-15 are denoted with a "4-15" in the superscript.

$\mathcal{M}^{\text{nc}}$ results in average RMSE reductions compared to Scenario 3 of 20.1%, 14.9%, 62.2%, for the active demand of the AC, OL, and NW demand, respectively. These results indicate that substation reactive power measurements significantly improves the disaggregation algorithm accuracy. Scenario 4 using $\mathcal{M}^{\text{c}}$ results in significant RMSE reductions versus Scenario 4 using $\mathcal{M}^{\text{nc}}$. However, this RMSE reduction is mainly due to the improvement in aggregate model accuracy within the disaggregation algorithm as the RMSEs for Scenario 4 using $\mathcal{M}^{\text{c}}$ are very similar to those of Scenario 2, but are slightly lower. These results indicate 1) that including reactive power flow measurements from the substation can significantly improve disaggregation accuracy and 2) that substation power flow measurements and substation current measurements may convey similar information as performance gains from real-time power flow measurements were small after complex current measurements were included into the models.

To evaluate the impact of real-time smart meter measurements with increasing frequency when utilizing active power, reactive power , and current measurements at the substation, we compare Scenarios 4, 4-60, 4-30, and 4-15 (all using $\mathcal{M}^{\text{c}}$). Scenario 4-60 achieves an average reduction in RMSE versus to Scenario 4 using $\mathcal{M}^{\text{c}}$ of 18.1% across all demand components, and a reduction of 26.8%, 24.8%, and 1.8% for the active power components of the AC, OL, and NW demand, respectively. The average RMSE across all demand components is reduced by 21.9% and 28.8% when comparing Scenario 4-30 and 4-15 to Scenario 4 using $\mathcal{M}^{\text{c}}$. As in the previous discussion of real-time smart meter measurements, disaggregation accuracy improves as smart meter measurements are available more frequently, even though the algorithm is significantly more accurate in Scenario 4 than in the previous discussion of smart meter measurements. Figure 7.6 depicts the improved disaggregation accuracy of Scenario 4-60 versus Scenario 4. Figures 7.6a and 7.6b show improved accuracy in estimating the AC and OL demand components in Scenario 4-15 versus Scenario 4 (using $\mathcal{M}^{\text{c}}$) as bias in the estimation error is reduced. Figure 7.6c also shows improved estimation accuracy in estimating the NW demand, but the improvements are slight as the average RMSE is on the order of 1-10 kW.

Finally, to evaluate the impact of real-time voltage magnitude and angle measurements, we compare the RMSE for Scenario 4 and Scenario 5. The RMSE for the active power of the AC and OL demand is reduced by 3.7% and 4.0%,respectively, versus that of Scenario 4. The RMSE for the active power component of the NW demand increases by 7.0%. These modest improvements may be due to the process of identifying the parameters used in the output matrices, which may not accurately reflect the relationship between the state and the measurements.

7.5 Chapter VII Conclusions

In this chapter, we developed an energy disaggregation algorithm to separate the measured, real-time demand of a distribution feeder into N components, where one component consisted of the network losses and capacitor bank power injections. The algorithm was based on a modified version of DMD, an online learning algorithm, that used sensor fusion to allow measurements from multiple sources on different timescales to be used within the algorithm. Output matrices were developed to allow fusion of four real-time measurement types: active and reactive power flows, differences in squared voltage magnitudes, differences in voltage angles, and smart meter measurements. In addition, aggregate models were developed for the disaggregation algorithm that utilize real-time complex current measurements from the substation and real-time outdoor temperature measurements.

Case studies sought to disaggregate the real-time feeder demand into the AC demand, the OL demand, and the NW demand. The simulated plant within the case studies incorporated a three-phase, unbalanced distribution feeder model with wye-connected loads, a voltage and temperature dependent AC model that incorporates real-world active power demand data and outdoor temperature data, and voltage-dependent ZIP models of the non-air conditioning loads that also incorporate real-world active power demand data. All load models within the plant included an active and reactive power component.

Results indicated that 1) incorporating real-time measurements from the distribution substation into the aggregate models of the disaggregation algorithm can significantly improve the models' prediction accuracy, 2) including reactive power measurements at the substation can improve disaggregation accuracy, 3) including real-time smart meter measurements at higher frequencies increasingly improves the energy disaggreation algorithm's accuracy, and 4) including measurements of the complex voltage at points within the feeder and at the substation can further improve accuracy.

Future work should 1) incorporate delta-connected loads into the simulation framework and disaggregation algorithm, 2) compare the disaggregation performance when using the estimated/identified parameters versus the exact network parameters, 3) investigate whether capacitor banks connecting/disconnecting to the feeder can be detected from the substation, 4) investigate methods to better approximate the losses in each portion of the feeder. An additional avenue of future work should disaggregate the demand while controlling the ACs for demand response.

CHAPTER VIII

Dissertation Conclusions

To conclude this dissertation, a summary of the dissertation is presented, the main contributions of each chapter are summarized, and a various avenues of future research are discussed.

8.1 Dissertation Summary

This dissertation presented work in three main areas: 1) the development of controllers, state estimator, and models for demand response scenarios, 2) a comparison of state estimation algorithms from control theory and online learning, and 3) the disaggregation of the power demand of a distribution feeder into components.

Chapter II presented state estimation and control algorithms that addressed communication delays in a demand response scenario that operates on timescales of seconds to provide frequency regulation. This demand response scenario included 1) frequent, noisy measurements of the aggregate demand of the demand-responsive load population, 2) infrequent, noise-free measurements of the state of the individual demand-responsive loads, and 3) delays that were independently applied to each type of measurement as well as the inputs. It was found that incorporating delay statistics into the controller, incorporating the realized measurement delays into the state estimator, and allowing input selection logic at the demand-responsive loads can help mitigate the effects of communication delays.

Chapter III simplified the controller from the previous chapter into a linear control law. These controllers' performance was evaluated in a demand response scenario that included input delays, as in the previous chapter. In contrast with the previous chapter, it was assumed that state measurements were available at each time-step. It was found that using the simpler control law slightly reduced the accuracy of the frequency regulation provided in comparison with the more complex controller, but the simpler controller also greatly reduced the computation time of the inputs.

Chapter IV then investigated the prediction accuracy of three existing models in residential demand response that seek to model aggregate behavior of a population of thermostatically controlled loads (TCLs) on timescales of seconds; modifications were made to the models to make them applicable to more realistic demand response scenario. New models based on Markov chains were developed that incorporate the effects of a time-varying outdoor temperature by including both the outdoor temperature and its trend in computing the model. A new transfer function model was proposed based on data analysis that proved to be more accurate than the existing transfer function model. It was found that the more detailed Markov model, which computes discrete states from two temperatures within the individual TCL models (versus one temperature for the simpler Markov model), was the most accurate of the models. Including the outdoor temperature trend in computing the Markov models significantly reduced the difference in estimation accuracy between the two Markov models, and the simpler Markov model may be advantageous as it has reduced computational complexity with similar modeling accuracy.

Chapter V compared state estimation techniques from two fields, control theory and online learning. Two commonly used Kalman filtering methods were presented along with two online learning methods, Dynamic Mirror Descent (DMD) and Dynamic Fixed Share (DFS). It was shown that user-defined functions and parameters in DMD can be chosen such that DMD produces estimates identical to that of a Kalman filter. Following this, it was shown that a multiple model Kalman filter, which incorporates model uncertainty into the Kalman filter framework, can be manipulated to produce identical state estimates to that of DFS, which also addresses model uncertainty. Then, heuristics that are commonly included into the multiple model Kalman filter were incorporated into DFS. Finally, the estimation accuracy of DMD and a Kalman filter were compared in a simple example, and several variations of multiple model Kalman filters and DFS were compared in a demand response scenario.

Chapter VI presented the feeder-level energy disaggregation framework and investigated disaggregation accuracy using real-time measurements of the total active power demand of a distribution feeder. Feeder-level energy disaggregation seeks to separate/disaggregate the total demand of a distribution feeder into components, where the components correspond to different types of aggregate demand, e.g., for different types of loads. In this chapter, the demand was disaggregated into the residential air conditioning demand and the remaining demand on the feeder. A model of the distribution feeder was not included in the problem framework. DFS, which incorporates DMD, was used to perform disaggregation using a variety of models for the demand components, and two implementations of DFS were developed for the case study scenarios. The results investigated the impact on disaggregation accuracy of using different sets of models within DFS, of incorporating estimation error covariances into DMD/DFS, and of varying various parameters within DFS. It was found that DFS could disaggregate the air conditioning load on a feeder from the other loads on the feeder with reasonable accuracy, that careful model selection is necessary as including inaccurate models can greatly decrease the accuracy of DFS, and that incorporating accurate covariance information can greatly improve the performance of DFS.

Chapter VII then extended the feeder-level energy disaggregation framework of the previous chapter to include additional real-time measurements. A three-phase, unbalanced distribution feeder model was included in the problem framework. In this chapter, the total feeder demand was disaggregated into the aggregate air conditioning demand, the demand of the non-air conditioning loads, and the demand of the distribution network equipment. Methods were presented to incorporate a variety of real-time measurements on different timescales into a variation of DMD via sensor fusion. These measurements included active and reactive power flow measurements, voltage magnitude measurements at the substation and at points within the feeder, voltage angle measurements at the substation and at points within the feeder, and active and reactive smart meter measurements. Real-time measurements of the complex current flowing into the distribution feeder from the substation were also included into the aggregate demand models used within the variation of DMD. Case studies investigated the disaggregation accuracy under different scenarios of real-time measurement availability. It was found that including real-time reactive power flow measurements, in addition to active power flow measurements, within the disaggregation algorithm improved disaggregation accuracy. The availability of real-time smart meter measurements further improved disaggregation accuracy, and this improvement increased as real-time smart meter measurement were made available more frequently. It was also found that including the complex current measurements into the aggregate models improved their prediction accuracy.

8.2 Summary of Contributions

This dissertation has made a variety of contributions to the domains of residential demand response, state estimation, and energy disaggregation. Chapter II contributed to the field of residential demand response by adapting networked state estimation and control approaches to address communication delays and communication limitations in a residential demand response scenario operating on time-scales of seconds. Chapter III contributed to the field of residential demand response by developing a simplified, linear control law that incorporates input delay statistics. Chapter IV contributed to the field of residential demand response by developing modifications of several existing aggregate models for more realistic scenarios and by comparing these models and their modifications in a single, detailed simulation. Chapter V contributed to the field of state estimation by establishing that DMD can be made to produce identical updates to a Kalman filter and by exploring the connections of DFS to a multiple model Kalman filter. Chapter VI contributed to the field of energy disaggregation by formulating the feeder-level energy disaggregation problem and by showing that DFS can be effectively used perform this disaggregation task when using active power measurements. Finally, Chapter VII contributed to the field of energy disaggregation by extending the work of the previous chapter to include additional real-time measurements into the feeder-level disaggregation framework.

The work presented in this dissertation focused on overcoming practical issues in implementing cost effective residential demand response to provide fast-timescale frequency regulation to the power grid. Addressing communication delays and measurement limitations, e.g., from smart meters, provides a framework to perform state estimation and control under realistic communication scenarios and provides algorithmic tools to reduce the required quality of the sensing/communication infrastructure. Developing methods to perform feeder-level energy disaggregation addressed a gap in residential demand response literature, namely the question of how to obtain a real-time feedback signal of the aggregate demand-responsive load for control and estimation algorithms. The feeder-level energy disaggregation work was again formulated while considering realistic sensing capabilities, meaning that extensive upgrades to the communication and sensing infrastructure are not needed to implement the algorithms. By working within the constraints of realistic sensing and communication capabilities, the need to upgrade equipment or install costly, high-quality communication systems and pervasive sensing is mitigated. Thus, the overall cost of implementing a demand response program can be reduced, because key, practical problems in operating a residential demand response program have been addressed.

8.3 Avenues of Future Research

There are a number of additional tasks that can help make cost effective frequency regulation via aggregations of residential loads a reality. Avenues of future research in residential demand response and feeder-level energy disaggregation should include a variety of technical, economic, social science, and policy tasks. Additional avenues of future research in machine learning and state estimation are proposed that, while more widely applicable, would benefit residential demand response and feeder-level energy disaggregation.

The proposed technical tasks include tasks that address open questions within this dissertation and tasks that more generally apply to the field of residential demand response. The proposed tasks that address open questions within this dissertation include investigating the impact of large-scale changes within the energy disaggregation problem, investigating how the parameters should be set given the accuracy of the models, and investigating how the parameters should be set given the timescale of changes to the system. These tasks are detailed below:

1. *Investigating the impact of large-scale changes within the energy disaggregation problem:* The work on the feeder-level energy disaggregation within this dissertation assumed that the system was reasonably well behaved in the sense that there were no large-scale, abrupt changes to the system. These types of changes may occur within normal operation and could include feeder reconfiguration and sudden, dramatic changes in the weather, e.g., as a storm approaches. The energy disaggregation work in this dissertation used online learning algorithms, and these large-scale changes may be detectable if dramatic increases in the achieved losses of the online learning algorithm are observed. It would be beneficial to investigate the impact of these types of changes and to determine whether they are detectable within the algorithm.

2. *Investigating how the parameters should be set given the accuracy of the models:* The accuracy of the online learning algorithms, both the single model variation and the multiple model variation, largely depends on the accuracy of the models and the parameter settings used within the algorithms. The accuracy of the models can be evaluated using historical data. However, the parameters within the algorithm were tuned to optimize the performance of the algorithms on

a set of tuning data without considering the underlying characteristics of the accuracy of the models. For example, models were used in Dynamic Fixed Share that were accurate for part of the day but inaccurate for the remainder of the day. It may have been possible to tune the parameters within Dynamic Fixed Share to take this behavior into account. Characterizing the accuracy of the models and how this accuracy changes over the period of interest may inform better parameter tuning. It would be beneficial to develop a formal relationship between accuracy of the models over the period of interest and the optimal parameter settings within the online learning algorithms.

3. *Investigating how the parameters should be set given the timescale of changes to the system:* Similar to the previous point, the parameters within the online learning algorithm were tuned without paying attention to the timescales of changes in the underlying demand components and the changes in weather-related characteristics driving those demand changes. It would be beneficial to derive optimal settings for the parameters within the online learning algorithm based on the timescales of the dynamics within (or driving) the components of the measured signal. This can help inform reasonable parameter settings in different scenarios based on the behavior of the underlying system.

The proposed technical tasks that more generally apply to the field of demand response include developing of a standard simulation framework, hardware-based testing of developed methods, investigating the performance of the collective control tasks, investigating the impact of demand response participation levels, and investigating a virtual power plant approach. These tasks are detailed below:

1. *Developing a standard simulation framework:* Developing a standard, realistic, and detailed simulation framework to evaluate the existing and future methods (e.g., models, controllers, state estimators, and other inference algorithms) within demand response literature would allow the direct comparison of methods. Presently, these methods are often developed in disparate scenarios that each have their own assumptions, making it difficult to compare their effectiveness against other techniques. Establishing a common framework, in a similar vein to the IEEE test networks for power systems, would allow a fair comparison of approaches to begin determining the most effective of the available methods in residential demand response. In addition, if the simulation framework includes reasonable assumptions about the sensing and communication infrastructure, it may encourage the development of methods that are more implementable.

2. *Hardware-based testing of developed methods:* Creating hardware-based evaluation methods would illuminate practical issues in existing residential demand response methods. Hardware-based testing could occur within a test bench or in a real-world pilot program of demand response with a goal of providing frequency regulation. The implementation of a variety of existing models in the more detailed simulation framework of Chapter IV led to modifications to the models to make them more suitable to the more realistic scenario, and it led to additional adjustments to the models that made them more accurate in this scenario. In a similar light, continuing to add more realism to the scenarios in which residential demand response methods are tested may spur more developments and improvements as factors impacting their performance are better understood, and as practical issues in their usage arise.

3. *Investigating the performance of the collective control tasks:* It would be beneficial if the existing residential demand response control algorithms that address feeder-level energy disaggregation, state estimation, control, and parameter identification of the demand-responsive devices were tested together to provide insight as to how accurate these algorithms are together. Algorithms for residential demand response are often developed while only looking at a particular portion of the overall problem. While this simplifies the process of developing the individual algorithms, an understanding of how these algorithms operate together is also valuable. In particular, the stability of the interconnected control tasks should be investigated to ensure that the approaches are implementable together in practice. In addition, it may be possible to integrate the previously independent control tasks to improve performance, e.g., by controlling the demand-responsive devices in a manner that benefits energy disaggregation or state estimation.

4. *Investigating the impact of demand response participation levels:* In the feeder-level energy disaggregation work, the goal was to disaggregate the air conditioning demand from the total demand of a distribution feeder. Additional work in feeder-level energy disaggregation could focus on disaggregating demand for a subset of the air conditioners on the feeder, where the subset of air conditioners could be assumed to be participating in demand response and the remaining air conditioners could be assumed to be operating normally. In addition, analysis of energy disaggregation accuracy versus the portion of air conditioners connected to the feeder within this subset could be carried out. This would

provide information about the portion of demand-responsive loads on a distribution feeder that are necessary to effectively carry out the inference algorithms in a real-world demand response program.

5. *Investigating a virtual power plant approach:* A virtual power plant approach could look at a distribution feeder, including the loads and generation resources connected to it, as a single entity within the power system (i.e., as a virtual power plant). In this context, the net demand, or the demand minus the generation, of the feeder over the course of a day is scheduled, e.g., via markets, similar to a traditional power plant. The control task in this setting is to coordinate the controllable loads, generation resources, uncontrollable loads, and network losses to follow the scheduled net demand. Measurements at the distribution substation of the power flow into the feeder, i.e., the net feeder demand, could be used to validate the realized versus scheduled net demand and to provide a feedback signal for carrying out the necessary control tasks. In addition, it would be beneficial to investigate the advantages and drawbacks of the virtual power plant approach versus an approach that treats demand-responsive loads separately.

Additional proposed tasks for future work include cost-benefit analyses to determine the net benefit of adding more sensing, communication, and automation to a demand response program, an investigation of the social factors driving demand response participation, and development of standards and policies for home automation infrastructure and smart appliances. These tasks are detailed below:

1. *Cost-benefit analyses to determine the net benefit of adding more sensing, communication, and automation to a demand response program:* Cost-benefit analyses comparing the implementation costs of technical capabilities within the communication, sensing, and actuation infrastructure versus the potential performance benefits and revenue increases would be instrumental in helping to determine reasonable assumptions in the development of residential demand response methods. For example, understanding the cost-benefit trade-off associated with enabling finer temperature sensing in a residential air conditioner informs the temperature detail with which models can be developed, tested, and implemented. In addition, understanding the cost-benefit trade-off of implementing various communication technologies informs the frequency and reliability with which measurements and inputs can be communicated. Another

analysis could investigate the performance of feeder-level energy disaggregation and the resulting performance in providing frequency regulation versus the costs of obtaining the various measurements that can be used within the algorithm. This economic analysis of infrastructure capabilities could then be used to quantify the costs and revenues of a demand response program and to inform reasonable assumptions in developing residential demand response methods.

2. *Investigation of the social factors driving demand response participation:* An investigation of the social factors encouraging or discouraging participation in demand response would enable demand response providers to better tailor their recruitment and educational efforts to increase participation in residential demand response programs. For example, the collection of user data, e.g., in mobile phone application usage, is more common today than it has ever been. However, there are often privacy concerns about the installation of smart meters. In addition, household occupants may be hesitant to allow a demand response provider to control their air conditioner. Demand response providers could take actions in their handling of the pertinent data to alleviate privacy concerns, and they could take a non-disruptive control approach that seeks to operate within the normal range of the device, which is defined by the user. Educating potential participants about efforts to protect their privacy and ensure non-disruptive control may boost interest in demand response programs. Methods to alleviate barriers to participation in demand response programs can improve demand response participation rates in a given area.

3. *Development of standards and policies for home automation infrastructure and smart appliances:* Smart meters have been widely deployed, and they allow two-way communication between a demand response provider and the smart meters of end users. However, additional in-home communication infrastructure is needed to enable a demand response provider to communicate with a demand-responsive appliance via the smart meter, and the appliance must be upgraded to enable responsiveness to demand response control signals. Standards should be developed ensuring that appliances that are good candidates for demand response have the necessary capabilities, e.g., communication and control capabilities. In addition, policies encouraging the adoption of smart devices and home automation systems that have built-in demand response capabilities would provide more widespread ability to execute demand response actions.

Additional avenues of future research in machine learning and state estimation include the adaptation of performance bounds for online learning algorithms to their Kalman filter counterparts, the development of performance bounds for the modified Dynamic Mirror Descent algorithm, an investigation of convex loss functions for different error characteristics, and an investigation of identifiability and observability in energy disaggregation. These tasks are detailed below:

1. *The adaptation of performance bounds for online learning algorithms to their Kalman filter counterparts:* Performance bounds have been developed for Dynamic Mirror Descent, and this dissertation showed that Dynamic Mirror Descent can be constructed to produce identical estimates to a Kalman filter. However, the assumptions of Dynamic Mirror Descent are more general than those of a Kalman filter. Adapting the performance bounds of Dynamic Mirror Descent to the assumptions of a Kalman filter may lead to new bounds on the accuracy of a Kalman filter. Similar performance bounds could be established for a multiple model Kalman filter by adapting the performance bounds for Dynamic Fixed Share.

2. *The development of performance bounds for the modified Dynamic Mirror Descent algorithm:* A modified version of Dynamic Mirror Descent was developed in this dissertation that uses measurements to modify the output of a model, rather than adjusting the state within the model. While this makes the algorithm applicable to more model types, the performance bounds for Dynamic Mirror Descent do not apply to it. Additional performance bounds could be developed for this algorithm.

3. *An investigation of convex loss functions for different error characteristics:* Dynamic Mirror Descent relies on a convex optimization problem to compute adjusted state estimates based on the realized loss (and the newly arrived measurements). In this dissertation, loss functions were constructed to include the covariance of the state and output estimation errors into the convex program. Additional work could develop other convex loss functions that capture other characteristics of the prediction errors and reflect those characteristics into the optimization problem. This could help inform the selection of the loss function and Bregman divergence functions to provide more accurate estimates based on the underlying characteristics of the system.

4. *Investigation of identifiability and observability in energy disaggregation:* Iden-

tifiablility and observability are properties that describe whether parameters or states within a model can be inferred given observations/measurements produced by the system under consideration. These properties are rarely referred to in energy disaggregation literature, but they may be informative in characterizing the performance, i.e., disaggregation accuracy, for a given energy disaggregation scenario. It would be beneficial to use identifiability/observability, and how well these criteria are satisfied, to quantify the effects of 1) disaggregating additional demand components, 2) additional measurements, and 3) additional models used within the online learning algorithms. For example, one could quantify the change in disaggregation accuracy if additional demand/generation components are disaggregated from the measured signals, one could optimize the additional measurements used within an energy disaggregation scenario to maximally improve identifiability/observability, or one could use identifiability/observability to inform the selection of models used within a multiple model estimation algorithm.

BIBLIOGRAPHY

[1] K. H. LaCommare and J. H. Eto, "Cost of power interruptions to electricity consumers in the United States (US)," *Energy*, vol. 31, no. 12, pp. 1845–1855, 2006.

[2] A. Von Meier, *Electric power systems: a conceptual introduction.* John Wiley & Sons, 2006.

[3] J. P. S. Catalão, *Electric power systems: advanced forecasting techniques and optimal generation scheduling.* CRC Press, 2012.

[4] H. Farhangi, "The path of the smart grid," *IEEE Power and Energy Magazine*, vol. 8, no. 1, 2010.

[5] R. Walawalkar, S. Fernands, N. Thakur, and K. R. Chevva, "Evolution and current status of demand response (DR) in electricity markets: Insights from PJM and NYISO," *Energy*, vol. 35, no. 4, pp. 1553–1560, 2010.

[6] U.S. Department of Energy, "Benefits of demand response in electricity markets and recommendations for achieving them," U.S. Department of Energy, Tech. Rep., Feb. 2006.

[7] K. Bhattacharya, M. Bollen, and J. E. Daalder, *Operation of restructured power systems.* Springer Science & Business Media, 2012.

[8] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control.* McGraw-Hill New York, 1994, vol. 7.

[9] D. S. Kirschen and G. Strbac, *Fundamentals of power system economics.* John Wiley & Sons, 2004.

[10] "Forward capacity market," https://www.iso-ne.com/markets-operations/markets/forward-capacity-market, ISO New England Inc., accessed: 2017-09-05.

[11] "Day-ahead and real-time energy markets," https://www.iso-ne.com/markets-operations/markets/da-rt-energy-markets, ISO New England Inc., accessed: 2017-09-05.

[12] Z. Chen, L. Wu, and Y. Fu, "Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1822–1831, 2012.

[13] A. J. Roscoe and G. Ault, "Supporting high penetrations of renewable generation via implementation of real-time electricity pricing and demand response," *IET Renewable Power Generation*, vol. 4, no. 4, pp. 369–382, 2010.

[14] M. H. Albadi and E. F. El-Saadany, "A summary of demand response in electricity markets," *Electric power systems research*, vol. 78, no. 11, pp. 1989–1996, 2008.

[15] H. P. Barker, "The centralized control of public lighting and off-peak loads by superimposed ripples," *Journal of the Institution of Electrical Engineers*, vol. 83, no. 504, pp. 823–836, 1938.

[16] A. J. Conejo, J. M. Morales, and L. Baringo, "Real-time demand response model," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 236–242, 2010.

[17] J. L. Mathieu, S. Koch, and D. S. Callaway, "State estimation and control of electric loads to manage real-time energy imbalance," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 430–440, 2013.

[18] S. Bashash and H. K. Fathy, "Modeling and control of aggregate air conditioning loads for robust renewable power management," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1318–1327, 2013.

[19] K. M. Tsui and S.-C. Chan, "Demand response optimization for smart home scheduling under real-time pricing," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1812–1821, 2012.

[20] N. Li, L. Chen, and S. H. Low, "Optimal demand response based on utility maximization in power networks," in *IEEE Power and Energy Society General Meeting*. IEEE, 2011.

[21] J. L. Mathieu, T. Haring, J. O. Ledyard, and G. Andersson, "Residential demand response program design: Engineering and economic perspectives," in *Conference on the European Energy Market (EEM)*. IEEE, 2013.

[22] W. J. Hausman and J. L. Neufeld, "Time-of-day pricing in the US electric power industry at the turn of the century," *The RAND Journal of Economics*, pp. 116–126, 1984.

[23] C. W. Gellings, "The concept of demand-side management for electric utilities," *Proceedings of the IEEE*, vol. 73, no. 10, pp. 1468–1470, 1985.

[24] A. M. Bruning, "Cold load pickup," *IEEE Transactions on Power Apparatus and Systems*, no. 4, pp. 1384–1386, 1979.

[25] C.-Y. Chong and A. S. Debs, "Statistical synthesis of power system functional load models," in *IEEE Conference on Decision and Control (CDC)*, 1979, pp. 264–269.

[26] R. Malhamé and C.-Y. Chong, "Electric load model synthesis by diffusion approximation of a high-order hybrid-state stochastic system," *IEEE Transactions on Automatic Control*, vol. AC-30, no. 9, pp. 854–860, 1985.

[27] S. Ihara and F. C. Schweppe, "Physically based modeling of cold load pickup," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 9, pp. 4142–4150, 1981.

[28] M.-L. Chan, E. N. Marsh, J. Y. Yoon, and G. B. Ackerman, "Simulation-based load synthesis methodology for evaluating load-management programs," *IEEE Transactions on Power Apparatus and Systems*, no. 4, pp. 1771–1778, 1981.

[29] S. H. Lee and C. L. Wilkins, "A practical approach to appliance load control analysis: a water heater case study," *IEEE Transactions on Power Apparatus and Systems*, no. 4, pp. 1007–1013, 1983.

[30] R. E. Mortensen and K. P. Haggerty, "A stochastic computer model for heating and cooling loads," *IEEE Transactions on Power Systems*, vol. 3, no. 3, pp. 1213–1219, 1988.

[31] S. Borenstein, "The trouble with electricity markets: understanding California's restructuring disaster," *The Journal of Economic Perspectives*, vol. 16, no. 1, pp. 191–211, 2002.

[32] Office of Electricity Delivery and Energy Reliability, "The American Recovery and Reinvestment Act Smart Grid Highlights," U.S. Department of Energy, Tech. Rep., Oct. 2004.

[33] Federal Energy Regulatory Commission, "Wholesale Competition in Regions with Organized Electric Markets," Federal Energy Regulatory Commission, Tech. Rep., Oct. 2008, Order 719.

[34] ——, "Demand Response Compensation in Organized Wholesale Energy Markets," Federal Energy Regulatory Commission, Tech. Rep., Mar. 2011, Order 745.

[35] ——, "Frequency Regulation Compensation in the Organized Wholesale Power Markets," Federal Energy Regulatory Commission, Tech. Rep., Oct. 2011, Order 755.

[36] ——, "Third-Party Provision of Ancillary Services; Accounting and Financial Reporting for New Electric Storage Technologies," Federal Energy Regulatory Commission, Tech. Rep., Jul. 2013, Order 784.

[37] D. S. Callaway and I. A. Hiskens, "Achieving controllability of electric loads," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 184–199, 2011.

[38] D. S. Callaway, "Tapping the energy storage potential in electric loads to deliver load following and regulation, with application to wind energy," *Energy Conversion and Management*, vol. 50, no. 5, pp. 1389–1400, 2009.

[39] W. Zhang, J. Lian, C.-Y. Chang, and K. Kalsi, "Aggregated modeling and control of air conditioning loads for demand response," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4655–4664, 2013.

[40] T. Borsche, F. Oldewurtel, and G. Andersson, "Minimizing communication cost for demand response using state estimation," in *PowerTech*, 2013.

[41] E. Vrettos, J. L. Mathieu, and G. Andersson, "Control of thermostatic loads using moving horizon estimation of individual load states," in *Proceedings of the Power Systems Computation Conference (PSCC)*, Wroclaw, Poland, Aug. 2014.

[42] A. Ghaffari, S. Moura, and M. Krstic, "PDE-based modeling, control, and stability analysis of heterogeneous thermostatically controlled load populations," *Journal of Dynamic Systems, Measurement, and Control*, 2015.

[43] J. H. Braslavsky, C. Perfumo, and J. K. Ward, "Model-based feedback control of distributed air-conditioning loads for fast demand-side ancillary services," in *Conference on Decision and Control (CDC)*, Dec. 2013, pp. 6274–6279.

[44] S. Koch, J. L. Mathieu, and D. S. Callaway, "Modeling and control of aggregated heterogeneous thermostatically controlled loads for ancillary services," in *Power Systems Computation Conference (PSCC)*, Stockholm, Sweden, Aug. 2011.

[45] A. Lesage-Landry and J. A. Taylor, "Learning to shift thermostatically controlled loads," in *Hawaii International Conference on System Sciences (HICSS)*, 2017.

[46] K. Kalsi, M. Elizondo, J. Fuller, S. Lu, and D. Chassin, "Development and validation of aggregated models for thermostatic controlled loads with demand response," in *Hawaii International Conference on Systems Science (HICSS)*, 2012.

[47] M. Lee, O. Aslam, B. Foster, D. Kathan, J. Kwok, L. Medearis, R. Palmer, P. Sporborg, and M. Tita, "Assessment of demand response and advanced metering," Federal Energy Regulatory Commission, Tech. Rep., 2013.

[48] A. McWilliams, "The U.S. market for home automation and security technologies," BCC Research, Tech. Rep., 2015, Report number: IAS031C.

[49] K. C. Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? The case of electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013.

[50] G. S. Ledva, E. Vrettos, S. Mastellone, G. Andersson, and J. L. Mathieu, "Managing communication delays and model error in demand response," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1299–1308, 2018.

[51] ——, "Applying networked estimation and control algorithms to address communication bandwidth limitations and latencies in demand response," in *Hawaii International Conference on System Sciences (HICSS)*, 2015, pp. 2645–2654.

[52] G. Strbac, "Demand side management: Benefits and challenges," *Energy policy*, vol. 36, no. 12, pp. 4419–4426, 2008.

[53] P. Siano, "Demand response and smart grids – a survey," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 461–478, 2014.

[54] SCE, "Southern California Edison company 2012-2014 demand response program portfolio," Southern California Edison, Tech. Rep. Application No. A.11-03-003 filed before the Public Utilities Commission of California, Mar. 2011.

[55] J. H. Eto, J. Nelson-Hoffman, E. Parker, C. Bernier, P. Young, D. Sheehan, J. Kueck, and B. Kirby, "The demand response spinning reserve demonstration–measuring the speed and magnitude of aggregated demand response," in *Hawaii International Conference on System Science (HICSS),*. IEEE, 2012.

[56] L. Zhang, H. Gao, and O. Kaynak, "Network-induced constraints in networked control systems-a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 403–416, 2013.

[57] K. Wada and A. Yokoyama, "Load frequency control using distributed batteries on the demand side with communication characteristics," in *Conference and Exhibition on Innovative Smart Grid Technologies*, Oct. 2012.

[58] N. Gatsis and G. B. Giannakis, "Residential load control: Distributed scheduling and convergence with lost AMI messages," *IEEE Transactions on Smart Grid*, vol. 3, no. 2, pp. 770–786, 2012.

[59] H. Hao, B. M. Sanandaji, K. Poolla, and T. L. Vincent, "Frequency regulation from flexible loads: Potential, economics, and implementation," in *American Control Conference (ACC)*, 2014, pp. 65–72.

[60] R. C. Sonderegger, "Dynamic models of house heating based on equivalent thermal parameters," Ph.D. dissertation, Princeton University, 1978.

[61] N. W. Wilson, B. S. Wagner, and W. G. Colborne, "Equivalent thermal parameters for an occupied gas-heated house," *ASHRAE Transactions*, vol. 91, no. CONF-850606-, 1985.

[62] K. Ogata, *Discrete-time control systems.* Prentice Hall Englewood Cliffs, NJ, 1995, vol. 2.

[63] (2015, Apr.) GridLAB-D House Class Documentation. Online. [Online]. Available: http://gridlab-d.sourceforge.net/wiki/index.php/House

[64] L. Schenato, "Optimal sensor fusion for distributed sensors subject to random delay and packet loss," in *46th IEEE Conference on Decision and Control*, 2007, pp. 1547–1552.

[65] J. Fischer, A. Hekler, and U. D. Hanebeck, "State estimation in networked control systems," in *15th International Conference on Information Fusion (FUSION)*, 2012, pp. 1947–1954.

[66] A. Shapiro and A. Philpott, "A tutorial on stochastic programming," *Manuscript. Available at www2. isye. gatech. edu/ashapiro/publications. html*, vol. 17, 2007.

[67] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip

[68] PJM. (2015, Mar.) Ancillary Services. reg-data-external-may-2014.xls. [Online]. Available: https://www.pjm.com/markets-and-operations/ancillary-services.aspx

[69] *PJM Manual 12: Balancing Operations*, PJM, 2014, revision 31.

[70] G. S. Ledva and J. L. Mathieu, "A linear approach to manage input delays while supplying frequency regulation using residential loads," in *The American Control Conference (ACC).* IEEE, 2017, pp. 741–747.

[71] S. Moura, J. Bendtsen, and V. Ruiz, "Observer design for boundary coupled PDEs: Application to thermostatically controlled loads in smart grids," in *IEEE Conference on Decision and Control (CDC)*, Florence, Italy, Dec. 2013.

[72] M. Alizadeh, A. Scaglione, and R. J. Thomas, "From packet to power switching: Digital direct load scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 6, pp. 1027–1036, 2012.

[73] D. Niyato, P. Wang, Z. Han, and E. Hossain, "Impact of packet loss on power demand estimation and power supply cost in smart grid," in *IEEE Wireless Communications and Networking Conference (WCNC).* IEEE, 2011, pp. 2024–2029.

[74] L. Zheng, S. Parkinson, D. Wang, L. Cai, and C. Crawford, "Energy efficient communication networks design for demand response in smart grid," in *Conference on Wireless Communications and Signal Processing.* IEEE, 2011.

[75] L. Zheng, N. Lu, and L. Cai, “Reliable wireless communication networks for demand response control,” *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 133–140, 2013.

[76] W. Zhang, J. Lian, C.-Y. Chang, K. Kalsi, and Y. Sun, “Reduced-order modeling of aggregated thermostatic loads with demand response,” in *IEEE Conference on Decision and Control (CDC)*, 2012, pp. 5592–5597.

[77] J. S. Bay, *Fundamentals of linear state space systems.* WCB/McGraw-Hill, 1999.

[78] H. Hao, T. Middelkoop, P. Barooah, and S. Meyn, “How demand response from commercial buildings will provide the regulation needs of the grid,” in *Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, 2012.

[79] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods.* Englewood Cliffs, NJ, USA: Prentice-Hall, 1990.

[80] G. S. Ledva, S. Peterson, and J. L. Mathieu, “Benchmarking of aggregate residential load models used for demand response,” in *Power & Energy Society General Meeting (PESGM).* IEEE, 2018 (In Press).

[81] C. Perfumo, E. Kofman, J. H. Braslavsky, and J. K. Ward, “Load management: Model-based control of aggregate power for populations of thermostatically controlled loads,” *Energy Conversion and Management*, vol. 55, pp. 36–48, 2012.

[82] N. Mahdavi, J. H. Braslavsky, and C. Perfumo, “Mapping the effect of ambient temperature on the power demand of populations of air conditioners,” *IEEE Transactions on Smart Grid*, 2016.

[83] “GridLAB-D Residential module user’s guide,” http://gridlab-d.shoutwiki.com/wiki/Residential_module_user%27s_guide, Accessed: 2017-11-01.

[84] J. L. Mathieu, M. Kamgarpour, J. Lygeros, G. Andersson, and D. S. Callaway, “Arbitraging intraday wholesale energy market prices with aggregations of thermostatic loads,” *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 763–772, 2015.

[85] Pecan Street Inc., “Dataport,” 2016.

[86] G. S. Ledva, Z. Du, L. Balzano, and J. L. Mathieu, “Disaggregating load by type from distribution system measurements in real-time,” in *Energy Markets and Responsive Grids*, S. Meyn, T. Samad, I. Hiskens, and J. Stoustrup, Eds. New York: Springer, 2018, pp. 413–437.

[87] G. S. Ledva, L. Balzano, and J. L. Mathieu, “Exploring connections between a multiple model kalman filter and dynamic fixed share with applications to demand response,” in *Conference on Control Technology and Applications (CCTA).* IEEE, 2018, pp. 217–223.

[88] D. Simon, "Optimal state estimation," in *Kalman, H Infinity, and Nonlinear Approaches, Wiley & Sons*, 2006.

[89] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software.* John Wiley & Sons, 2004.

[90] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 647–662, 2015.

[91] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[92] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 50–61, 2010.

[93] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.

[94] G. Welch and G. Bishop, "An introduction to the Kalman filter," http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf, August 2001, SIGGRAPH 2001 Course 8.

[95] R. E. Mortensen and K. P. Haggerty, "Dynamics of heating and cooling loads: models, simulation, and actual utility data," *IEEE Transactions on Power Systems*, vol. 5, no. 1, pp. 243–249, 1990.

[96] G. S. Ledva, L. Balzano, and J. L. Mathieu, "Real-time energy disaggregation of a distribution feeder's demand using online learning," *IEEE Transactions on Power Systems*, vol. 33, no. 4, pp. 4730 –4740, 2018.

[97] ——, "Inferring the behavior of distributed energy resources with online learning," in *Allerton Conference on Communication, Control, and Computing (Allerton).* IEEE, 2015, pp. 187–194.

[98] GTM Research/SEIA: U.S. Solar Market Insight, "Solar market insight report 2015 Q1," 2015, http://www.seia.org/research-resources/solar-market-insight-report-2015-q1.

[99] Navigant Research, "Direct load control and dynamic pricing programs, DR markets, and DR management systems for residential customers: Global market analysis and forecasts," 2015, https://www.navigantresearch.com/research/residential-demand-response.

[100] M. P. Lee, O. Aslam, B. Foster, S. Hou, D. Kathan, C. Pechman, and C. Young, "Assessment of Demand Response & Advanced Metering," FERC, Staff Report, Dec. 2015, https://www.ferc.gov/legal/staff-reports/2015/demand-response.pdf.

[101] M. E. Berges, E. Goldman, H. S. Matthews, and L. Soibelman, "Enhancing electricity audits in residential buildings with nonintrusive load monitoring," *J. Industrial Ecology*, vol. 14, no. 5, pp. 844–858, 2010.

[102] J. Z. Kolter and T. S. Jaakkola, "Approximate inference in additive factorial HMMs with application to energy disaggregation," in *Artificial Intelligence and Statistics*, vol. 22, 2012, pp. 1472–1482.

[103] R. Dong, L. J. Ratliff, H. Ohlsson, and S. S. Sastry, "Energy disaggregation via adaptive filtering," in *Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2013, pp. 173–180.

[104] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[105] T. Hong, "Short term electric load forecasting," Ph.D. dissertation, North Carolina State University, 2010.

[106] T. Byers, "How Comverge is using machine learning to improve demand response forecasts," Feb. 2016, http://blog.comverge.com/expert-insights/how-comverge-is-using-machine-learning-to-improve-demand-response-forecasts/.

[107] J. W. Taylor, "An evaluation of methods for very short-term load forecasting using minute-by-minute British data," *International Journal of Forecasting*, vol. 24, no. 4, pp. 645–658, 2008.

[108] N. Hatziargyriou, "Machine learning applications to power systems," in *Machine Learning and Its Applications*. Springer, 2001, pp. 308–317.

[109] M. Negnevitsky, P. Mandal, and A. K. Srivastava, "Machine learning applications for load, price and wind power prediction in power systems," in *IEEE Conference on Intelligent System Applications to Power Systems*. IEEE, 2009.

[110] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2531–2539, 2010.

[111] M.-G. Zhang, "Short-term load forecasting based on support vector machines regression," in *IEEE Conference on Machine Learning and Cybernetics*, vol. 7. IEEE, 2005, pp. 4310–4314.

[112] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting solar generation from weather forecasts using machine learning," in *IEEE Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2011, pp. 528–533.

[113] T. T. Teo, T. Logenthiran, and W. L. Woo, "Forecasting of photovoltaic power using extreme learning machine," in *IEEE Conference on Innovative Smart Grid Technologies-Asia*. IEEE, 2015.

[114] J. Taylor and J. L. Mathieu, “Index policies for demand response,” *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1287–1295, 2014.

[115] D. Kalathil and R. Rajagopal, “Online learning for demand response,” in *Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015.

[116] F. Ruelens, B. Claessens, S. Vandael, B. De Schutter, R. Babuska, and R. Belmans, “Residential demand response of thermostatically controlled loads using batch reinforcement learning,” *IEEE Transactions on Smart Grid*, 2016.

[117] K. Khezeli and E. Bitar, “Risk-sensitive learning and pricing for demand response,” *IEEE Transactions on Smart Grid*, 2017.

[118] K. P. Schneider, Y. Chen, D. P. Chassin, R. G. Pratt, D. W. Engel, and S. E. Thompson, “Modern grid initiative distribution taxonomy final report,” Pacific Northwest National Laboratory, Tech. Rep., 2008, PNNL-18035.

[119] NOAA, “NNDC Climatic Data Online,” 2009, Satellite and Information Service National Climate Data Center. [Online]. Available: http://www7.ncdc.noaa.gov/CDO/dataproduct

[120] J. L. Mathieu, A. J. Gadgil, D. S. Callaway, P. N. Price, and S. Kiliccote, “Characterizing the response of commercial and industrial facilities to dynamic pricing signals from the utility,” in *International Conference on Energy Sustainability.* Phoenix, AZ: ASME, May 2010.

[121] M. S. Grewal and A. P. Andrews, *Kalman filtering.* Wiley, 2015.

[122] The Federal Energy Regulatory Commission, “Distributed energy resources: Technical considerations for the bulk power system,” FERC, Staff Report AD18-10-000, Feb. 2018, https://www.ferc.gov/CalendarFiles/20180215112833-der-report.pdf.

[123] B. Foster, D. Burns, J. Grove, D. Kathan, M. P. Lee, S. Peirovi, and C. Schilling, “Assessment of Demand Response & Advanced Metering,” FERC, Staff Report, Dec. 2017, https://www.ferc.gov/legal/staff-reports/2017/DR-AM-Report2017.pdf.

[124] D. Atanackovic and V. Dabic, “Deployment of real-time state estimator and load flow in BC Hydro DMS-challenges and opportunities,” in *Power and Energy Society General Meeting (PESGM).* IEEE, 2013.

[125] F. de León, M. Diaz-Aguiló, and A. Raza, “Conservation voltage reduction,” *Smart Grid Handbook*, vol. 3, 2016.

[126] A. Von Meier, D. Culler, A. McEachern, and R. Arghandeh, “Micro-synchrophasors for distribution systems,” in *Innovative Smart Grid Technologies Conference (ISGT).* IEEE, 2014.

[127] E. C. Kara, C. M. Roberts, M. Tabone, L. Alvarez, D. S. Callaway, and E. M. Stewart, “Disaggregating solar generation from feeder-level measurements,” *Sustainable Energy, Grids and Networks*, vol. 13, pp. 112–121, 2018.

[128] Y. Xu and J. V. Milanovic, “Artificial-intelligence-based methodology for load disaggregation at bulk supply point,” *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 795–803, 2015.

[129] M. Domanovic, G. Dobric, and N. Rajakovic, “Direct method for estimation of demand composition at supply point,” in *Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion (MedPower).* IET, 2016.

[130] D. Chakravorty, B. Chaudhuri, and S. Hui, “Estimation of aggregate reserve with point-of-load voltage control,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4649–4658, 2018.

[131] Q. Gan and C. J. Harris, “Comparison of two measurement fusion methods for kalman-filter-based multisensor data fusion,” *IEEE Transactions on Aerospace and Electronic systems*, vol. 37, no. 1, pp. 273–279, 2001.

[132] S. Bolognani, N. Bof, D. Michelotti, R. Muraro, and L. Schenato, “Identification of power distribution network topology via voltage correlation analysis,” in *Conference on Decision and Control (CDC).* IEEE, 2013, pp. 1659–1664.

[133] M. E. Baran and F. F. Wu, “Network reconfiguration in distribution systems for loss reduction and load balancing,” *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, 1989.

[134] R. Dobbe, D. Arnold, S. Liu, D. Callaway, and C. Tomlin, “Real-time distribution grid state estimation with limited sensors and load forecasting,” in *International Conference on Cyber-Physical Systems (ICCPS).* IEEE, 2016.

[135] W. H. Kersting, “Radial distribution test feeders,” in *IEEE Power Engineering Society Winter Meeting*, vol. 2, 2001, pp. 908–912.

[136] Pecan Street Inc., “Dataport,” https://dataport.cloud/, 2018.

[137] R. Bravo, R. Yinger, D. Chassin, H. Huang, N. Lu, I. Hiskens, and G. Venkataramanan, “Final project report load modeling transmission research,” *Lawrence Berkeley National Laboratory (LBNL)*, 2010.

[138] North American Electric Reliability Corporation, “Technical reference document: Dynamic load modeling,” NERC, Tech. Rep., 12 2016.

[139] L. M. Hajagos and B. Danai, “Laboratory measurements and models of modern loads and their effect on voltage stability studies,” *IEEE Transactions on Power Systems*, vol. 13, no. 2, pp. 584–592, 1998.

[140] A. Bokhari, A. Alkan, R. Dogan, M. Diaz-Aguiló, F. de León, D. Czarkowski, Z. Zabar, L. Birenbaum, A. Noel, and R. E. Uosef, "Experimental determination of the ZIP coefficients for modern residential, commercial, and industrial loads," *IEEE Transactions on Power Delivery*, vol. 29, no. 3, pp. 1372–1381, 2014.

[141] W. H. Kersting, *Distribution system modeling and analysis.* CRC press, 2012.