Building Energy Modeling and Control Methods for Optimization and Renewables Integration

by

Eric M. Burger

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Assistant Professor Scott J. Moura, Chair
Associate Professor Duncan S. Callaway
Professor Raja Sengupta

Spring 2017

Building Energy Modeling and Control Methods for Optimization and Renewables Integration

Abstract

Building Energy Modeling and Control Methods
for Optimization and Renewables Integration

by

Eric M. Burger

Doctor of Philosophy in Engineering - Civil and Environmental Engineering

University of California, Berkeley

Assistant Professor Scott J. Moura, Chair

This dissertation presents techniques for the numerical modeling and control of building systems, with an emphasis on thermostatically controlled loads. The primary objective of this work is to address technical challenges related to the management of energy use in commercial and residential buildings. This work is motivated by the need to enhance the performance of building systems and by the potential for aggregated loads to perform load following and regulation ancillary services, thereby enabling the further adoption of intermittent renewable energy generation technologies. To increase the generalizability of the techniques, an emphasis is placed on recursive and adaptive methods which minimize the need for customization to specific buildings and applications.

The techniques presented in this dissertation can be divided into two general categories: modeling and control. Modeling techniques encompass the processing of data streams from sensors and the training of numerical models. These models enable us to predict the energy use of a building and of sub-systems, such as a heating, ventilation, and air conditioning (HVAC) unit. Specifically, we first present an ensemble learning method for the short-term forecasting of total electricity demand in buildings. As the deployment of intermittent renewable energy resources continues to rise, the generation of accurate building-level electricity demand forecasts will be valuable to both grid operators and building energy management systems. Second, we present a recursive parameter estimation technique for identifying a thermostatically controlled load (TCL) model that is non-linear in the parameters. For TCLs to perform demand response services in real-time markets, online methods for parameter estimation are needed. Third, we develop a piecewise linear thermal model of a residential building and train the model using data collected from a custom-built thermostat. This model is capable of approximating unmodeled dynamics within a building by learning from sensor data.

Control techniques encompass the application of optimal control theory, model predictive control, and convex distributed optimization to TCLs. First, we present the alternative control trajectory (ACT) representation, a novel method for the approximate optimization of non-convex discrete systems. This approach enables the optimal control of a population of non-convex agents using distributed convex optimization techniques. Second, we present a distributed convex optimization algorithm for the control of a TCL population. Experimental results demonstrate the application of this algorithm to the problem of renewable energy generation following.

This dissertation contributes to the development of intelligent energy management systems for buildings by presenting a suite of novel and adaptable modeling and control techniques. Applications focus on optimizing the performance of building operations and on facilitating the integration of renewable energy resources.

To Mom and Dad

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor, Prof. Scott Moura, for his mentorship and thoughtful contributions in support of my research endeavors, my lab mate, Hector Perez, for his enthusiastic encouragement and counseling, my partner, Kylin Navarro, for her love, patience, and understanding during my years of graduate school, and my parents, Mike and Mary Burger, for teaching me to be kind and hardworking and for years of unwavering support in my academic pursuits.

Chapter 1

Introduction

In developed countries around the globe, commercial and residential buildings are responsible for between 20% and 40% of energy consumption [78]. In developing countries, particularly India and China, rising standards of living and the impacts of climate change are expected to significantly increase the energy use of buildings. Studies suggest that global energy demand for air conditioning may increase by 70% over the next century due to climate change effects. Within the U.S., commercial and residential buildings account for 40% of primary energy consumption, more than either the transportation sector or the industrial sector (29% and 30%, respectively) [94]. In particular, heating, ventilation, and air-conditioning (HVAC) compose 43% of commercial and 54% of residential building site energy end-use. Space heating alone accounts for 45% of residential energy use. Additionally, it is estimated that nearly 10% of all greenhouse gas emissions within the U.S. are due to HVAC [94].

HVAC systems are an integral part of buildings responsible for regulating temperature, humidity, carbon dioxide, and airflow, conditions which directly impact occupant health and comfort. Estimates suggest that component upgrades and advanced HVAC control systems could reduce building energy usage by up to 30% [11]. Such advanced systems can improve the efficiency of building operations and better regulate indoor conditions to improve air quality and occupant comfort.

Advances in HVAC control, as well as the control of other electric loads, would also enable buildings to participate in demand response markets. Maintaining a continuous and instantaneous balance between generation and load is a fundamental requirement of the electric power system [34]. The variability of renewable energy resources, particularly wind and solar, poses a challenge for power system operators [45]. Namely, as renewable penetration increases it will be necessary for operators to procure more ancillary services, such as regulation and load following, to maintain balance between generation and load [60, 107]. Responsive thermostatically controlled loads (TCLs) have a high potential for providing such ancillary services [87, 18]. By shifting loads, building energy management systems will improve power grid stability and reduce energy related carbon emissions [5, 13].

The study of building energy use is an ever expanding research topic which crosses the fields of engineering, architecture, computer science, and mathematics. This dissertation contributes to the ongoing development of monitoring and control systems for building energy use. By its very nature, the study of building energy use is also the study of people and their needs and habits. Accordingly, we seek to understand and optimize a building and its systems as observed in operation as opposed to by design. We place an emphasis on the development of theory as well as validation through experimentation. Whenever possible, we incorporate sensory data collected from systems in operation. In this way, we seek to advance theory as well as practice.

The techniques presented in this dissertation can be divided into two general categories: modeling and control. Modeling techniques are presented in Part I and encompass the processing of data streams from sensors and the training of mathematical models. These models enable us to predict the energy use of a building and of sub-systems, such as a HVAC unit. Control techniques are presented in Part II and encompass the application of optimal control theory, model predictive control, and convex distributed optimization to TCLs. This allows us to control the energy use of a building to meet local or grid objectives.

The novel contributions of this dissertation include:

- *Ensemble learning method for the electricity demand forecasting of buildings*: The method combines the predictions from multiple minimally-customized forecasting models to produce a single short-term prediction of electricity demand. We demonstrate that the proposed method produces accurate electricity demand forecasts and that by continuously updating the forecaster's parameters, responds to changes in electricity demand patterns.

- *Kalman filter-based recursive parameter estimation technique for non-linear TCL model*: Using the Kalman filter and unscented Kalman filter, we develop four filter methods (single, joint, dual, and triple) for recursively estimating a discrete-time TCL model that is non-linear in the parameters. The analysis of the experimental results reveals that each method successfully converges to comparable parameter estimates and is capable of adapting to changes in the TCL characteristics.

- *Piecewise linear thermal model of a residential building*: We develop a model capable of capturing the predominant dynamics and disturbance patterns of a forced-air residential heating system. Experimental results demonstrate the potential of the model and parameter estimation method to produce accurate forecasts of the air temperature within a conditioned space.

- *Alternative control trajectory (ACT) representation*: The ACT representation enables the control of a non-convex discrete system to be represented as a convex program. The solution to this program can be interpreted stochastically for implementation. The significant contribution of this approach is that it allows for the approximate optimal control of a population of non-convex agents using distributed convex optimization techniques.

- *Distributed convex optimization of a TCL population*: Using the ACT representation, we developed a convex program that enables the distributed optimization of a TCL population. Experimental results demonstrate the potential for TCLs to help maintain a continuous and instantaneous balance between generation and load by participating in real-time ancillary service markets.

In Chapter 2, we present an ensemble learning method for the short-term forecasting of total electricity demand in buildings. Before attempting to change the electricity demand of a building, it is important to be able to accurately predict the demand over a short time horizon (on the order of hours or days). Over the past 3 decades, extensive research has focused on the task of predicting electricity demand in buildings using an array of machine learning techniques [3, 92, 41, 40]. Generally, these techniques must be tuned or customized to individual buildings, requiring a large time investment by an engineer before a monitoring system can begin producing electricity demand forecasts. In this dissertation, we present an ensemble learning method for electricity forecasting. Put simply, we combine the predictions from multiple minimally-customized forecasting methods to produce a single prediction. To improve the accuracy of this prediction, we learn, from past data, how the multiple forecasts should be combined. Rather than assuming that demand behaviors are time invariant, the proposed method responds to changes in electricity demand patterns by continuously updating the forecaster's parameters.

In Chapter 3, we present a recursive parameter estimation technique for identifying a thermostatically controlled load (TCL) model that is non-linear in the parameters. For a population of TCLs to provide ancillary services, it is necessary for each TCL to model its own behavior and to predict its energy demand. TCLs with poorly fit models will undermine the ability of the population to accurately perform ancillary services. Given that most TCLs experience regular changes to their physical characteristics (e.g. the contents of a refrigerator, the flow through a water heater, or the occupancy of a conditioned room), a linear time-invariant model is likely to prove inadequate. Also, for TCLs like radiant heaters and air conditioners, it is not possible for the manufacturer to predetermine the physical characteristics of the spaces that will be conditioned. Therefore, to improve the performance of distributed TCL control methods, it is necessary to employ recursive or online parameter estimation algorithms to fit and continuously update each TCL's model.

In Chapter 4, we develop a piecewise linear thermal model of a residential building and train the model using data collected from a custom-built thermostat. To effectively control the operation of an HVAC system, it is essential that a model predictive controller incorporate an accurate mathematical representation of a building's thermal dynamics. An ideal control-oriented model would capture the predominant dynamics and disturbance patterns within a building, enable accurate forecasting, adapt to future changes in building use, provide a model structure suitable for optimization, and be amenable to real-time data-driven model identification methods. The piecewise linear model and recursive parameter estimation method presented in this dissertation meets these characteristics by approximating unmodeled dynamics within a building based on observable patterns in the sensor data.

In Chapter 5, we present the alternative control trajectory (ACT) representation, a novel method for the approximate optimization of non-convex discrete systems. Energy systems like EVs and TCLs often have binary or discrete states due to hardware limitations and efficiency characteristics. Consequently, non-convex techniques are generally required for optimal control. This poses a challenge for load aggregation applications since distributed optimization methods generally require linearity or convexity in the agents. The ACT representation presented in this dissertation enables the control of a non-convex energy system to be represented as a convex program. The solution to this program can be interpreted stochastically for implementation. The significant contribution of this approach is that it allows for the approximate optimal control of a population of non-convex agents using distributed convex optimization techniques.

Finally, in Chapter 6, we present a distributed convex optimization algorithm for the control of a TCL population. Specifically, we examine the potential of TCLs, such as refrigerators and electric water heaters, to provide generation following services in real-time energy markets (1 to 5 minutes). Past literature on modeling and control of TCL populations has generally focused on aggregation methods with centralized control [61, 61, 70, 17]. In contrast, the approach presented in this dissertation employs a distributed control scheme with a centralized aggregator. Therefore, each TCL is controlled independently and the role of the aggregator is to enable coordination within the population. To perform distributed optimization across the population of TCLs, we apply a variation of the alternating direction method of multipliers (ADMM) algorithm. We numerically demonstrate the algorithm's potential for controlling a TCL population's total power demand within an error tolerance of 10 kW.

Each of the chapters in Parts I and II is self-contained and includes sections detailing the motivations, relevant literature, experimental results, and conclusions. Lastly, research that preceded or expands upon the work in Parts 1 and 2 is included in the Appendices. Appendix A presents a gated ensemble learning method for the short-term forecasting of total electricity demand in buildings. This method recursively trains multiple models for predicting the electricity demand of a building and employs a gating method to determine which model should be used to generate a forecast at a given time step. Appendix B presents work on the parameter estimation and model predictive control of an apartment with electric baseboard heaters and includes an analysis of how the forecast horizon impacts the optimality of the controller. Appendix C presents a simulation study demonstrating the capability of the parameter estimation method developed in Chapter 3 to quickly converge to new parameter estimates in response to changes in the system dynamics. Additionally, Appendix C presents simulation results for a population of refrigerators which optimize their power demand based on a demand response electricity price event. These studies show the advantage of using model predictive control with a recursive parameter estimation algorithm rather than employing a fixed set of model parameters.

Part I

Modeling

Chapter 2

Building Electricity Load Forecasting

This chapter presents a stacking ensemble method which addresses the need for a generalizable approach to building-level electricity demand forecasting. Rather than using a single model to predict electricity demand, the method uses a weighted linear combination of forecasts from multiple sub-models. By learning the model weights in real-time using electricity demand data streams and a moving horizon training technique, the method is more robust than a single model approach. Experimental results demonstrate the application of the method to electricity demand data sets for 8 different buildings.

2.1 Motivation & Background

Commercial and residential buildings account for 74.1% of U.S. electricity consumption, more than either the transportation sector or the industrial sector (0.2% and 25.7%, respectively) [96]. Maintaining a continuous and instantaneous balance between generation and load is a fundamental requirement of the electric power system [34]. To reliably match supply with demand, the forecasting of grid-level electricity loads has long been a central part of the planning and management of electrical utilities [3]. The accuracy of these forecasts has a strong impact on the reliability and cost of power system operations. Trends, such as vehicle electrification and distributed generation, are expected to pose new challenges for grid operators. In particular, traditionally centralized power flow and generator dispatch tasks are becoming increasingly decentralized, creating a critical need for local electricity forecasting.

To improve the accuracy of electricity demand forecasts and aid in power system management, recent attention has been placed on short-term building-level electricity demand forecasting using a wide range of models [92, 41]. Accurate and adaptive forecasting of demand-side loads will play a critical role in maintaining grid stability and enabling renewables integration. Additionally, many novel optimal control schemes, under research umbrellas such as demand response and microgrid management, require short-term building electricity demand forecasts to aid in decision making [40].

2.2 Literature Review

Supply-side and demand-side electricity forecasting has been a topic of research for many decades. The literature is filled with a variety of well-cited modeling approaches, each differing in algorithmic complexity, estimation procedure, and computational cost. Of particular note are the variants of Artificial Neural Networks (ANN) [3, 92, 41, 7, 33, 43, 68], Support Vector Regression (SVR) [62, 76, 44, 35] and Autoregressive Integrated Moving Average (ARIMA) models [3, 76, 44, 108, 21, 88, 71]. Lesser but nonetheless noteworthy attention has been given to approaches such as Multiple Linear Regression [3, 62, 81], Fuzzy Logic [3, 52], Decision Trees [92], and k-Nearest Neighbors (k-NN).

These studies provide a broad catalog of use-cases and demonstrate the performance of certain forecasting algorithms when applied to specific building types. In particular, [3, 92, 68, 88] provide a survey of electricity forecasting methods and a high-level comparison of techniques. In [33], the authors provide a detailed description of ANNs and their application to load forecasting, including data pre-processing and ANN architectures. The work in [41] details the development of a seasonal ANN approach and the advantage over a Seasonal ARIMA (SARIMA) model when applied to 6 building datasets. The focus of [71] is on the introduction of motion sensor data to improve the accuracy of an ARIMA model. In [43, 62, 108, 71, 52], the authors perform an in-depth analysis of the power demand patterns of a particular building in order to customize a forecasting model.

In papers with experimental results, the authors have generally applied their electricity demand forecasting technique to only a small number of datasets. Consequently, the literature is rich with forecasting algorithms tailored for individual buildings. This leads us to the following question: Is it possible to design a minimally-customized forecasting algorithm that is widely applicable across a diversity of building types, enabling scalability? We pursue this question by proposing a stacking ensemble learning method for electricity demand forecasting.

Specifically, due to unique building characteristics, occupancy patterns, and individual energy use behaviors, the literature demonstrates that no single "silver bullet" model structure can accurately forecast electricity demand across all buildings. For example, some forecasting models may produce accurate predictions under identifiable conditions, such as a seasonal trend, a morning routine, or an extended absence. Other models may be ideal for buildings with energy use behaviors that are periodic over long periods of time. For buildings with frequent changes in occupancy patterns, recursively trained models may yield the highest accuracy.

2.3 Contributions

A key contribution of this chapter is to develop an ensemble learning method that reduces the need for intensive model selection on a case-by-case basis. Rather, an engineer can select a set of different forecasting models that have proven effective in past case studies (e.g. the literature cited above). Once the models have been trained on building-specific electricity demand records, our ensemble method can learn, in real-time, which sub-models to favor and which to avoid for a particular building.

With our stacking ensemble method, we generate electricity demand forecasts using the weighted sum of predictions from multiple different forecasting sub-models. The sub-model weights are recursively learned using an electricity demand data stream and a moving horizon training technique. In this way, the ensemble method is able to learn in real-time and to produce short-term electricity demand forecasts that are automatically tailored to a particular building and instance in time. In addition to forecast accuracy, this chapter will place an emphasis on method adaptability and ease of use. While we have implemented certain forecasting sub-models in this chapter, the method is intended to allow the sub-models to be interchangeable.

Chapter Outline

This chapter is organized into three sections: Methods, Results, and Conclusions. In Section 2.4 Methods, we briefly present background theory for the two exemplary regression sub-models employed in this chapter, Ordinary (Linear) Least Squares with ℓ_2 Regularization (Ridge) and k-Nearest Neighbors (k-NN). These regression models will compose the sub-models in our ensemble method. Additionally, Section 2.4 presents the stacking ensemble learning method for electricity demand forecasting with a moving horizon training technique. In Section 2.5 Results, we apply and analyze the ensemble method to 8 commercial/university building electricity demand datasets. Key conclusions and future research directions are summarized in Section 2.6 Conclusions.

2.4 Methods

Regression Models

In this chapter, we will consider one parametric regression model, Ordinary (Linear) Least Squares with ℓ_2 Regularization (Ridge), and one nonparametric model, k-Nearest Neighbors with uniform weights and binary tree data structure (k-NN), for use as sub-models in our stacking ensemble method. The structure of both regression models are briefly described in the following subsections. While we have elected to employ relatively simple regression models, our ensemble method is such that these models could easily be replaced with more complex regression models, such as Artificial Neural Networks (ANNs) or Seasonal Autoregressive Integrated Moving Average (SARIMA) models.

Ordinary Least Squares with ℓ_2 Regularization

Ordinary Least Squares with ℓ_2 Regularization (Ridge) fits a linear model with coefficients $w \in \mathbf{R}^n$ to minimize the sum of squared errors between the observed and predicted responses, while imposing a penalty on the size of coefficients measured by their ℓ_2-norm. The linear model of a system with univariate output is given by

$$\begin{aligned}\hat{y} &= w_0 x_0 + w_1 x_1 + \ldots + w_n x_n \\ &= \sum_k w_k x_k = w^T x\end{aligned} \tag{2.1}$$

with variables $x \in \mathbf{R}^n$, the model input, $\hat{y} \in \mathbf{R}$, the predicted response, n, the number of inputs or features in x, and $k = 1, \ldots, n$.

The linear model is trained on a set of inputs and observed responses by solving the quadratic program:

$$\min_w \sum_i (w^T x_i - y_i)^2 + \lambda \|w\|_2^2 \tag{2.2}$$

with variables $x_i \in \mathbf{R}^n$, the model input for the i-th data point, $y_i \in \mathbf{R}$, the i-th observed response, $w \in \mathbf{R}^n$, the weighting coefficients, and $i = 1, \ldots, N$, where N is the number of data samples and n is the number of features in x_i. Lastly, λ is a weighting term for the regularization penalty.

For a system with a multivariate output $\hat{y} \in \mathbf{R}^m$, we will treat the outputs as uncorrelated and define a set of coefficients $w_j \in \mathbf{R}^n$ for each predicted response $\hat{y}_j \in \mathbf{R}$ for $j = 1, \ldots, m$. Thus, the multivariate linear model is

$$\hat{y}_j = w_j^T x, \ \forall j = 1, \ldots, m \tag{2.3}$$

The weights of the multivariate model are determined by solving the quadratic program:

$$\min_w \sum_i \sum_j (w_j^T x_i - y_{i,j})^2 + \sum_j \lambda \|w_j\|_2^2 \tag{2.4}$$

with variables $x_i \in \mathbf{R}^n$, the model input, $y_{i,j} \in \mathbf{R}$, the j-th response observed response, $w_j \in \mathbf{R}^n$, the weighting coefficients of the j-th response, $i = 1, \ldots, N$, and $j = 1, \ldots, m$, where N is the number of data samples, n is the number of features in x_i, and m is the number of observations in y_i.

k-Nearest Neighbors Regression

In k-Nearest Neighbors Regression (k-NN), an input $x \in \mathbf{R}^n$ is mapped to a continuous output value according to the weighted mean of the k nearest data points or neighbors, as defined by the Euclidean distance. In this chapter, we will use uniform weights. In other words, each point in a neighborhood a contributes uniformly and thus the predicted univariate response $\hat{y} \in \mathbf{R}$ is the mean of the k-nearest neighbors.

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_{a,i} \tag{2.5}$$

with variable y_a, the set of k observed responses $y \in \mathbf{R}$ in neighborhood a. For a system with multivariate output $\hat{y} \in \mathbf{R}^m$, the model is defined as the mean of each observation j over the k-nearest neighbors.

$$\hat{y}_j = \frac{1}{k}\sum_{i=1}^{k} y_{a,i,j} \ \forall j = 1, \ldots, m \tag{2.6}$$

Given a new input x, it is possible to determine the neighborhood by computing the Euclidean distance (i.e. ℓ_2-norm of the difference) between the new input x and every data point in the training data set x_i for $i = 1, \ldots, N$ and then ordering the distances to identify the nearest neighbors. However, this brute-force search is computationally inefficient for large datasets.

To improve the efficiency of the neighborhood identification, the training data points are partitioned into a tree data structure. A commonly used approach for organizing points in a multi-dimensional space is the ball tree data structure, a binary tree in which every node defines a D-dimensional hypersphere or ball. At each node, data points are assigned to the left or right balls according to their distance from the ball's center. At each terminal node or leaf, the data points are enumerated inside the ball. We refer the reader to [73] for a description of ball tree construction algorithms.

Stacking Ensemble Learning

In this section, we develop a regression method that produces a prediction according to the weighted sum of predictions from multiple sub-models. Ensemble learning methods which linearly combine the predictions of multiple models are generally referred to as stacking or stacked generalization methods and can often outperform any one of the trained sub-models (see e.g. [104, 10]). To produce a multivariate prediction $\hat{y}_\Sigma \in \mathbf{R}^m$, the ensemble model is defined as the weighted sum of each prediction $\hat{y}_s \in \mathbf{R}^m$ from each sub-model s, as given by

$$\hat{y}_\Sigma = \sum_{s=1}^{M} \theta_s \hat{y}_s \tag{2.7}$$

with variable $\theta_s \in \mathbf{R}$, the weighting coefficient of sub-model s where M is the number of sub-models and subscript $s = 1, \ldots, M$ indexes the coefficients (i.e. $[\theta_1, \ldots, \theta_M] = \theta \in \mathbf{R}^M$). Note that we are not calculating the weighted mean of the sub-models. Therefore, we are not requiring that the values of the weighting coefficients sum to 1 or that the individual weights are positive.

We will employ an Ordinary Least Squares with ℓ_2 Regularization (Ridge) approach for learning the weighting coefficients θ. By solving a quadratic optimization problem, we can identify the weighting coefficients that minimize the error between the observations and the weighted sum of the sub-model predictions. The optimization problem and stacking ensemble model at time step t are given by

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N}\sum_{j=1}^{m}\left(y_{i,j} - \sum_{s=1}^{M}\theta_s\hat{y}_{s,i,j}\right)^2 + \lambda\sum_{s=1}^{M}\theta_s^2 \tag{2.8}$$

$$\hat{y}_{\Sigma,t} = \sum_{s=1}^{M}\theta_s^\star\hat{y}_{s,t} \tag{2.9}$$

with variables $\theta^\star \in \mathbf{R^M}$, the optimal weighting coefficients, $y_i \in \mathbf{R^m}$, the i-th observed multivariate response, $\hat{y}_{s,i} \in \mathbf{R^m}$, the i-th prediction from sub-model s, $\hat{y}_{\Sigma,t} \in \mathbf{R^m}$, the ensemble model prediction at t, and $i = 1, \ldots, N$, where N is the number of data samples used for training and m is the length of y_i. Subscript $j = 1, \ldots, m$ indexes the $j-$th response. Lastly, λ is a weighting term for the regularization penalty.

Stacking Ensemble with Moving Horizon Training

To enable the ensemble model to learn and adapt to changes in the observed system, we will utilize time-varying weights, $\theta_{t,s} \in \mathbf{R}$ for $s = 1, \ldots, M$ (i.e. $\theta_t \in \mathbf{R}^M$). These time-varying weights will be calculated by minimizing the error between the observations and the weighted sum of the sub-model predictions over a retrospective moving horizon of data samples. In other words, at each time step t, we will retrain the ensemble model using only the most recent T observations. Therefore, the observations used for calculating the weights θ_t will move with the current time step. The moving horizon optimization problem and stacking ensemble model at time step t are

$$\theta_t^\star = \underset{\theta_t}{\operatorname{argmin}} \sum_{i=1}^{T}\sum_{j=1}^{m}\left(y_{t-i,j} - \sum_{s=1}^{M}\theta_{t,s}\hat{y}_{s,t-i,j}\right)^2 + \lambda\sum_{s=1}^{M}\theta_{t,s}^2 \tag{2.10}$$

$$\hat{y}_{\Sigma,t} = \sum_{s=1}^{M}\theta_{t,s}^\star\hat{y}_{s,t} \tag{2.11}$$

with variables $\theta_t^\star \in \mathbf{R^M}$, the optimal weighting coefficients at time step t, $y_t \in \mathbf{R^m}$, the observation at t, $\hat{y}_{s,t} \in \mathbf{R^m}$, the prediction of sub-model s at t, and $\hat{y}_{\Sigma,t} \in \mathbf{R^m}$, the ensemble model prediction at t where T is the number of observations in the moving horizon.

Data

For experimentation, this chapter considers 2 years of metered hourly electricity demand (kW) data for 8 buildings on the University of California, Berkeley campus. This time-series data has been provided by the facilities team at the University of California, Berkeley and will be used as the observation data for the sub-models and ensemble model. Submetered electricity demand data and building operations data, such a occupancy measurements and mechanical system schedules, were not available. The 8 buildings were selected for their diversity. These buildings include classrooms, offices, libraries, and research facilities. We have also acquired hourly air temperature (°C) and relative humidity (%RH) data from a local weather station [86].

Ensemble Learning for Electricity Demand Forecasting

In this chapter, we will apply the stacking ensemble model above to the building electricity forecasting problem. Given the many unpredictable behaviors of occupants and the unique physical and mechanical characteristics of every building, a single model approach to electricity demand forecasting may perform very well in one case and very poorly in another. Furthermore, the incorporation of exogenous signals like regional weather conditions may improve a model's accuracy but such benefits cannot be guaranteed. Only through observation and experimentation can the best regression models and input types be identified for a particular building.

By employing our stacking ensemble learning method with moving horizon training technique, we seek to improve the robustness of our electricity demand forecaster. Before we test the ensemble model, we must first train and test the sub-models. In this chapter, we will use 8 sub-models, 4 using Ridge and 4 using k-NN. These models will be used to generate short-term multivariate electricity demand forecasts, specifically 6 consecutive hourly electricity demand predictions (i.e. $\hat{y} \in \mathbf{R}^6$).

The regression models will employ 4 different input types or feature sets: electricity demand (D), time (T), electricity demand and time (DT), and electricity demand, time, and exogenous weather data (DTE). Thus, there is 1 Ridge model and 1 k-NN model for each of the 4 input types. The electricity demand input type (D) consists of the 24 hourly records that precede the desired forecast ($x \in \mathbf{R}^{24}$). The time input type (T) is the current weekday and hour represented as a sparse binary vector ($x \in \{0,1\}^{31}$). The demand and time input type (DT) combines the demand and time inputs ($x \in \mathbf{R}^{55}$). The demand, time, and exogenous weather data input type (DTE) is the demand and time input plus current air temperature (°C) and relative humidity (%RH) data retrieved from a local weather station ($x \in \mathbf{R}^{57}$)[86].

In this study, we train a set of 8 sub-models for each building. Training data from one building is not used to fit the models of another building. The sub-models are trained in an off-line batch manner (i.e. trained once on a large dataset) using 18 months of hourly input data from January 1st, 2012, to July 1st, 2013 (i.e. 13,128 training data points). The remaining 6 months of hourly data, July 1st, 2013, to January 1st, 2014 are reserved for testing of the sub-models and the ensemble method (i.e. 4,416 testing data points).

Testing of the stacking ensemble method is done by repeating the following procedure for each time step $t = 1, \ldots, 4416$ where t represents the integer-valued hour between July 1st, 2013, and January 1st, 2014.

1. Using each of the 8 sub-models, generate a 6 hour electricity demand forecast, $\hat{y}_{s,t} \in \mathbf{R}^6$ for $s = 1, \dots, 8$, as given by either (2.3) or (2.6).
2. Learn the model weights $\theta^\star_{t,s} \in \mathbf{R}$ for $s = 1, \dots, 8$ by minimizing (2.10) over a moving horizon of the previous $T = 168$ observations (i.e. 7 days)
3. Generate the ensemble model's forecast, $\hat{y}_{\Sigma,t} \in \mathbf{R}^6$, as given by (2.11).

Once a forecast has been generated for every data point in the testing set, we calculate the errors between the observation y_t and the forecast $\hat{y}_{\Sigma,t}$ for $t = 1, \dots, 4416$. To evaluate the advantage of our ensemble method over a single model approach, we also calculate the errors between the observation y_t and the sub-model forecasts $\hat{y}_{\Sigma,t}$ for $s = 1, \dots, 8$ and $t = 1, \dots, 4416$. To enable the comparison of forecasting error between different buildings, the performance of the sub-models and the ensemble model will be reported as the mean absolute percent error (MAPE),

$$\text{MAPE} = \frac{100\%}{mN} \sum_{i=1}^{N} \sum_{j=1}^{m} \left| \frac{y_{i,j} - \hat{y}_{i,j}}{y_{i,j}} \right| \tag{2.12}$$

with variables $y_i \in \mathbf{R}^m$, the i-th observation, and $\hat{y}_i \in \mathbf{R}^m$, the i-th prediction, where m represents the number of outputs in the prediction ($m = 6$) and N, the number of predictions.

2.5 Results

The sub-model and ensemble model performances for each of the 8 building datasets are summarized in Figure 2.1. In the figure, the marker color indicates the regression technique used by each model (Ridge, k-NN, or Ensemble) and the marker shape indicates the data type (D, T, DT, DTE, or Ensemble). The sub-model results (Ridge and k-NN) denote the forecast MAPE produced from that particular sub-model, over the testing dataset. The ensemble model results indicate the forecast MAPE produced by minimizing the moving horizon optimization problem and a weighted linear combination of the sub-model forecasts. Examples of the multivariate electricity demand forecasts $\hat{y}_{\Sigma,t} \in \mathbf{R}^6$ produced by the ensemble model are presented in Figure 2.3. Note that the figure plots $\hat{y}_{\Sigma,t}$ starting at but excluding the most recent observation.

By comparing the results in Figure 2.1 for each building, we can distinguish sub-models that generally perform poorly (e.g. Ridge and k-NN with T input) from sub-models that generally perform well (e.g. Ridge and k-NN with DT input). We also observe dispersion among the results, particularly in Buildings E, F, and H. This dispersion represents a challenge for building level electricity forecasting. To produce the best results using a single model approach, an engineer must perform model selection for every deployment. This is difficult to scale. Just because a certain regression model and input type has performed well for one building does not guarantee it will do the same for another building.

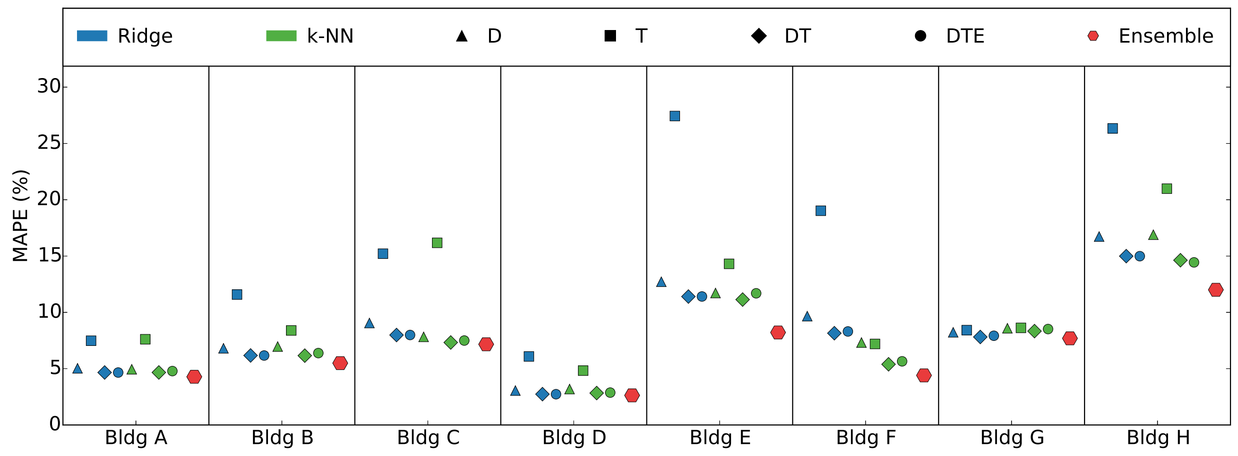

Figure 2.1: **Ensemble Model and Sub-Model Performance Results.** The mean absolute percent error (MAPE) of the ensemble model and sub-models of each building for every 6 hour forecast between July 1st, 2013, and January 1st, 2014.

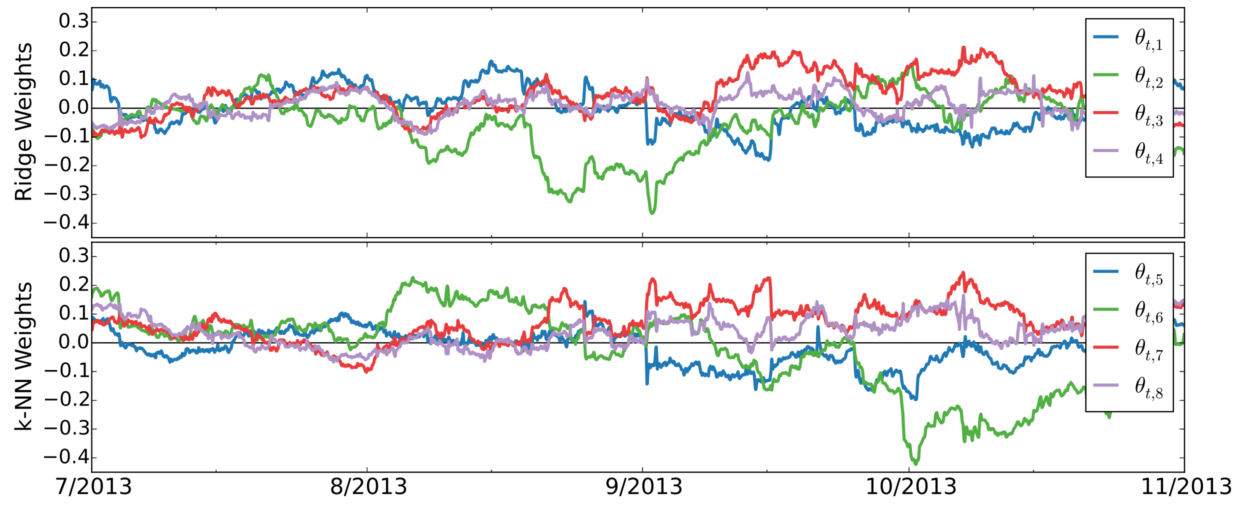

Figure 2.2: **Time-Varying Sub-Model Weights.** Examples of time varying weights $\theta_{t,s}$ for building E from July 1st to November 1st, 2013.

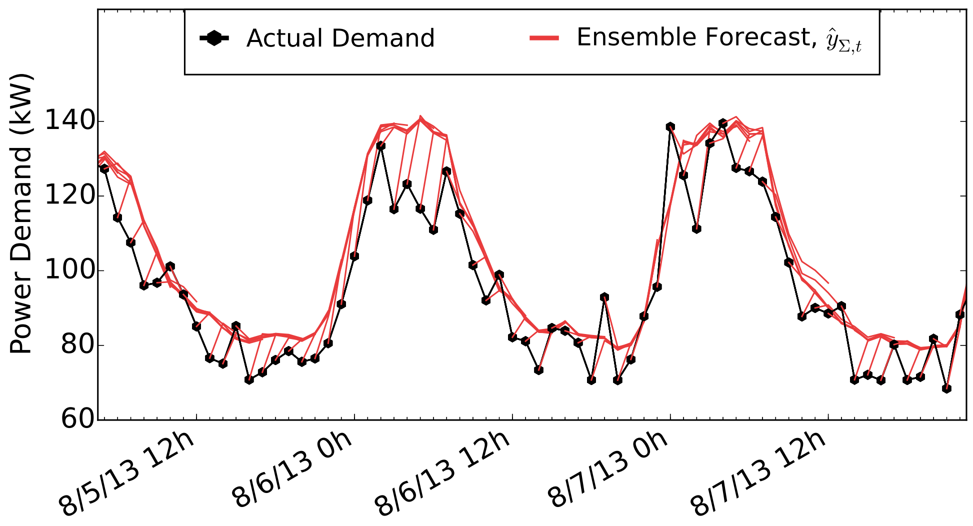

Figure 2.3: **Building E Ensemble Forecasts.** Examples of 6 hour electricity demand forecasts for building E using the stacking ensemble learning method.

As indicated by the results, the ensemble model performs comparable to or better than the best sub-model for each building. Therefore, by minimizing the moving horizon optimization problem, the ensemble model is able to (i) learn the sub-model weights in an online manner, and (ii) produce a linear combination of sub-model forecasts that is comparable to or better than the best sub-model forecast. This characteristic is valuable to grid operators and building-level applications. Specifically, engineers need only identify a set of sub-models which generally perform well for demand-side electricity demand forecasting. Then, after training each sub-model on data from a particular building, the stacking ensemble learning method with moving horizon training technique can adaptively identify the weighting of each sub-model for that building.

Figure 2.2 presents the sub-model weights $\theta_{t,s}$ of the Building E ensemble model from July 1st to November 1st, 2013. The weights $\theta_{t,1}$, $\theta_{t,2}$, $\theta_{t,3}$, and $\theta_{t,4}$ correspond to the Ridge models with D, T, DT, and DTE input types, respectively. Similarly, the weights $\theta_{t,5}$, $\theta_{t,6}$, $\theta_{t,7}$, and $\theta_{t,8}$ correspond to the k-NN models with D, T, DT, and DTE input types, respectively. As shown, the model weights do not converge but rather continuously evolve in time. Because the weights are determined by minimizing the moving horizon optimization problem, there are trends in the weighting values, but as the training data changes, so do the weights. Of particular note is the sharp change in the parameter values around September 1st, 2013. This can be attributed to the start of the fall academic semester at UC Berkeley and the corresponding change in electricity demand patterns.

2.6 Conclusions

This chapter presented a stacking ensemble learning method with a moving horizon training approach. We applied the method to the short-term building-level electricity demand forecasting problem. The experimental results demonstrate enhanced forecasting accuracy across a diversity of buildings due to two features out our approach: (i) employing a linear combination of sub-models, and (ii) adaptively learning the stacked model weights in real-time. The practical advantages are notable. Namely, the proposed method enables reliable forecasts over evolving use patterns across a wide diversity of buildings, in contrast to selecting and tailoring a single model for each building.

Additionally, the adaptability provided by the moving horizon training approach enables enhanced control applications. Rather than assuming that demand behaviors are time invariant, the proposed method responds to changes in electricity demand patterns. We have demonstrated this method on 8 buildings' datasets using 8 sub-models each. The results demonstrate that the stacking ensemble method produces equal or better accuracy than single models for multivariate electricity demand forecasts for building-level applications.

Chapter 3

Recursive Parameter Estimation of Thermostatically Controlled Loads

This chapter presents various unscented Kalman filter (UKF) algorithm variations for recursively identifying a thermostatically controlled load (TCL) model that is non-linear in the parameters. Experimental results demonstrate the parameter estimation of two residential refrigerators.

3.1 Motivation & Background

Large populations of thermostatically controlled loads (TCLs) hold great potential for performing ancillary services in power systems. The advantages of responsive TCLs over large storage technologies include: (i) they are already well-established technologies; (ii) they are spatially distributed around the power system; (iii) they employ simple and fast local actuation; (iv) they are unimpaired by the outage of individuals in the population; and (v) they - on the aggregate - can produce a quasi-continuous response despite the discrete nature of the individual controls [13, 18, 65].

Because TCLs are controlled according to a temperature setpoint and deadband range, customers are generally indifferent to precisely when electricity is consumed. The inherent flexibility of TCLs, such as refrigerators and electric water heaters, makes them promising candidates for provisioning power system services. In fact, direct load control (DLC) and demand response (DR) programs are increasingly controlling TCLs, among other electric loads, to improve power grid stability [25, 63].

For a population of TCLs to provide ancillary services, it is necessary for each TCL to model its own behavior and to predict its energy demand. TCLs with poorly fit models will undermine the ability of the population to accurately perform ancillary services. Given that most TCLs experience regular changes to their physical characteristics (e.g. the contents of a refrigerator, the flow through a water heater, or the occupancy of a conditioned room), a linear time-invariant model is likely to prove inadequate. Also, for TCLs like radiant heaters and air conditioners, it is not possible for the manufacturer to predetermine the physical characteristics of the spaces that will be conditioned. Therefore, to improve the performance of distributed TCL control methods, it is necessary to employ recursive or online parameter estimation algorithms to fit and continuously update each TCL's model.

Contributions

This chapter contributes to the development of recursive parameter estimation algorithms for TCLs by investigating various unscented Kalman filters for the estimation of a TCL model that is non-linear in the parameters. We present four closely related filter methods (single, joint, dual, and triple) employing both the standard Kalman filter (KF), and unscented Kalman filter (UKF) algorithms. Specifically, we consider: (i) a single filter approach in which one UKF estimates the TCL parameters; (ii) a joint filter approach in which one UKF simultaneously estimates both the parameters and the state; (iii) a dual filter approach in which one UKF estimates the parameters and one KF estimates the state; and (iv) a triple filter approach in which one UKF estimates the parameters, one KF estimates the state, and another KF estimates the model inputs. Finally, we present experimental parameter estimation results using real temperature data from two residential refrigerators.

Chapter Outline

This chapter is organized as follows. Section II discusses the TCL model and Section III overviews the parameter estimation problem. Sections IV and V provide background for the standard Kalman filter (KF) and the unscented Kalman filter (UKF), respectively. Section VI formulates four filter methods for recursive parameter estimation of a TCL. Section VII provides numerical examples of our proposed algorithms. Finally, Section VII summarizes key results.

3.2 Thermostatically Controlled Load Model

The predominant dynamics of a thermostatically controlled load (TCL) can be represented by the first order continuous time state equation

$$\dot{T}^t = \frac{T_\infty^t - T^t}{RC} + \frac{Pm^t}{C} \tag{3.1}$$

where $T^t \in \mathbf{R}$, $T_\infty^t \in \mathbf{R}$, and $m^t \in \{0, 1\}$ are the temperature of the conditioned mass (state, °C), the ambient air temperature (disturbance input, °C), and the discrete state of the mechanical system (control input, On/Off), respectively. The parameters R (°C$/kW$), C ($kJ/$°C), and P (kW) represent the thermal resistance, thermal capacitance, and rate of energy transfer, respectively.

The model can be expressed in the state-space form

$$\dot{T}^t = A_c T^t + B_c u^t \tag{3.2}$$

where

$$\begin{aligned} A_c &= \begin{bmatrix} \frac{-1}{RC} \end{bmatrix} \\ B_c &= \begin{bmatrix} \frac{1}{RC} & \frac{P}{C} \end{bmatrix} \\ u^t &= \begin{bmatrix} T^t_\infty \\ m^t \end{bmatrix} \end{aligned} \tag{3.3}$$

Assuming a zero-order hold on the input u, the model can be discretized using the transforms

$$\begin{aligned} A_d &= e^{A_c \Delta t} \\ B_d &= A_c^{-1}(A_d - I)B_c \end{aligned} \tag{3.4}$$

where Δt defines the length in hours between each time step. We define this as $\Delta t = 1/60$ (hours). Therefore, the state-space model becomes

$$T^{k+1} = A_d T^k + B_d u^k \tag{3.5}$$

where

$$\begin{aligned} A_d &= \begin{bmatrix} e^{-\frac{\Delta t}{RC}} \end{bmatrix} \\ B_d &= \begin{bmatrix} (1 - e^{-\frac{\Delta t}{RC}}) & (1 - e^{-\frac{\Delta t}{RC}})RP \end{bmatrix} \\ u^k &= \begin{bmatrix} T^k_\infty \\ m^k \end{bmatrix} \end{aligned} \tag{3.6}$$

and $k = 1, 2, \ldots, n$ denotes the integer-valued time step.

If we treat the control input m^k as a state with a piecewise update equation which enforces a temperature deadband range, the TCL can be expressed as the hybrid state discrete time model [69, 38, 17, 13, 14]

$$T^{k+1} = \theta_1 T^k + (1 - \theta_1)(T^k_\infty + \theta_2 m^k) + \theta_3 \tag{3.7a}$$

$$m^{k+1} = \begin{cases} 1 & \text{if } T^k > T_{set} + \frac{\delta}{2} \\ 0 & \text{if } T^k < T_{set} - \frac{\delta}{2} \\ m^k & \text{otherwise} \end{cases} \tag{3.7b}$$

where state variables $T^k \in \mathbf{R}$ and $m^k \in \{0, 1\}$ denote the temperature of the conditioned mass and the discrete state (on or off) of the mechanical system, respectively. Additionally, $T^k_\infty \in \mathbf{R}$ is the ambient temperature (°C), $T_{set} \in \mathbf{R}$ the temperature setpoint (°C), and $\delta \in \mathbf{R}$ the temperature deadband width (°C).

The parameter θ_1 represents the thermal characteristics of the conditioned mass as defined by $\theta_1 = \exp(-\Delta t/RC)$, θ_2 the energy transfer to or from the mass due to the systems operation as defined by $\theta_2 = RP$, and θ_3 an additive noise process accounting for energy gain or loss not directly modeled. Note that the sign conventions in (3.7) assume that the TCL is providing a cooling load and that P (and thus θ_2) is negative.

3.3 Nonlinear Parameter Estimation Background

A fundamental machine learning problem involves the identification of a nonlinear mapping

$$y^k = G(x^k, \theta) \tag{3.8}$$

where variable $x^k \in \mathbf{R}^X$ is the input, $y^k \in \mathbf{R}^Y$ is the output, and the nonlinear map G is parameterized by $\theta \in \mathbf{R}^\Theta$. Additionally, k denotes the integer-valued time step and X, Y, and Θ are the number of inputs, outputs, and parameters, respectively.

Batch Parameter Estimation

Learning can be performed in a batch manner by producing estimates of the parameters $\hat{\theta}$ given a training set of observed inputs and desired outputs, $\{x, y\}$. The goal of a parameter estimation algorithm is to minimize some function of the error between the desired and estimated outputs as given by $e^k = y^k - G(x^k, \hat{\theta})$.

Recursive Parameter Estimation

The parameter estimation problem can be expressed in a recursive form using a discrete-time state-space model representation

$$\theta^k = \theta^{k-1} + n^k \tag{3.9a}$$
$$y^k = G(x^k, \theta^k) + e^k \tag{3.9b}$$

where θ^k represents the parameter estimates at time step k and $n^k \in \mathbf{R}^\Theta$ corresponds to the parameter update noise (i.e. change in parameter values). The goal of a recursive parameter estimation algorithm is to produce $\hat{\theta}^k$ so as to minimize some function of the error e^k.

3.4 Kalman Filter Background

The Kalman filter (KF) is a recursive estimator for linear models such as the discrete-time state-space model

$$x^k = Ax^{k-1} + Bu^k + v^k \tag{3.10a}$$
$$y^k = Cx^k + Du^k + w^k \tag{3.10b}$$

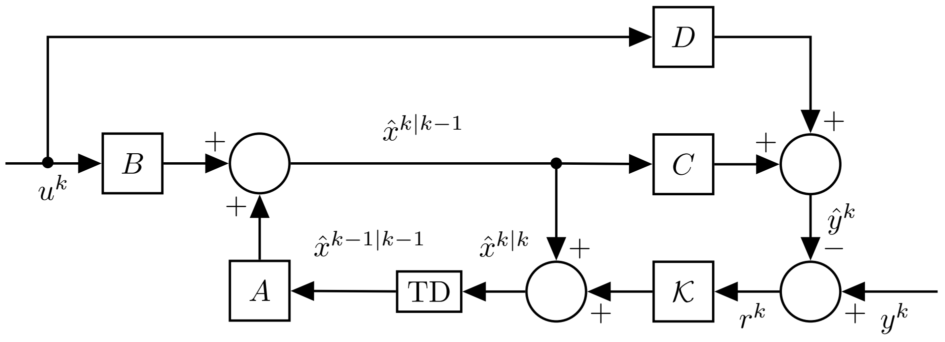

Figure 3.1: Kalman Filter Diagram

where variable $x^k \in \mathbf{R}^X$ is the state of the system, $u^k \in \mathbf{R}^U$ is the known exogenous input, and $y^k \in \mathbf{R}^Y$ is the observed measurement signal. The state transition model is given by $A \in \mathbf{R}^{X\times X}$ and the control-input model by $B \in \mathbf{R}^{X\times U}$. The process noise $v^k \in \mathbf{R}^X$ has covariance $Q_v \in \mathbf{R}^{X\times X}$, $v^k \sim N(0, Q_v)$. The observation model is given by $C \in \mathbf{R}^{Y\times X}$ and the feedthrough model by $D \in \mathbf{R}^{Y\times U}$. The measurement noise $w^k \in \mathbf{R}^Y$ has covariance $Q_w \in \mathbf{R}^{Y\times Y}$, $w^k \sim N(0, Q_w)$. The variances of v^k and w^k (i.e. diagonal elements of Q_v and Q_w, respectively) must be known in order to implement a Kalman filter.

The Kalman filter (KF) algorithm consists of a prediction step and an update/correction step. The KF will model x^k as a Gaussian random variable (GRV) with estimated mean $\hat{x}^k \in \mathbf{R}^X$ and covariance $Q_x^k \in \mathbf{R}^{X\times X}$. To provide clarity, it is helpful to expand the k notation to distinguish between the state estimates produced before and after the KF correction step. Therefore, at each time step k, the predicted (a priori) state estimate, denoted as $\hat{x}^{k|k-1}$, is the mean estimate of x^k given measurements $y^0, \ldots, y^{k-1}$. The corrected (a posterior) state estimate, $\hat{x}^{k|k}$, is the mean estimate of x^k given measurements $y^0, \ldots, y^k$. To reiterate, throughout this chapter, the uncorrected predictions (a priori) are denoted by $k|k-1$ or $k+1|k$ whereas the corrected predictions (a posterior) are denoted by $k|k$, $k-1|k-1$, or $k+1|k+1$.

The KF prediction step is given by

$$\hat{x}^{k|k-1} = A\hat{x}^{k-1|k-1} + Bu^k \tag{3.11a}$$

$$Q_x^{k|k-1} = AQ_x^{k-1|k-1}A^T + Q_v \tag{3.11b}$$

and the update/correction step by

$$\hat{y}^k = C\hat{x}^{k|k-1} + Du^k \tag{3.12a}$$

$$Q_y = CQ_x^{k|k-1}C^T + Q_w \tag{3.12b}$$

$$\mathcal{K} = Q_x^{k|k-1} C^T Q_y^{-1} \tag{3.13a}$$
$$r^k = y^k - \hat{y}^k \tag{3.13b}$$
$$\hat{x}^{k|k} = \hat{x}^{k|k-1} + \mathcal{K} r^k \tag{3.13c}$$
$$Q_x^{k|k} = Q_x^{k|k-1} - \mathcal{K} Q_y \mathcal{K}^T \tag{3.13d}$$

Figure 3.1 illustrates the KF algorithm. The block TD represents a time delay (commonly denoted in controls literature by z^{-1} or $1/z$, the Z-transform of the delay operator). To simplify notation in this chapter, we will express the Kalman filter algorithm with the following 3 operator expressions

$$\begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \end{bmatrix} = KF_x\left(\begin{bmatrix} A \\ B \end{bmatrix}, \begin{bmatrix} \hat{x}^{k-1|k-1} \\ Q_x^{k-1|k-1} \end{bmatrix}, u^k, Q_v \right) \tag{3.14a}$$
$$\begin{bmatrix} \hat{y}^{k} \\ Q_y^{k} \end{bmatrix} = KF_y\left(\begin{bmatrix} C \\ D \end{bmatrix}, \begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \end{bmatrix}, u^k, Q_w \right) \tag{3.14b}$$
$$\begin{bmatrix} \hat{x}^{k|k} \\ Q_x^{k|k} \\ r^k \end{bmatrix} = KF_c\left(\begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \end{bmatrix}, \begin{bmatrix} \hat{y}^{k} \\ Q_y^{k} \end{bmatrix}, C, y^k \right) \tag{3.14c}$$

where (3.14a) corresponds to (3.11a-3.11b), (3.14b) to (3.12a-3.12b), and (3.14c) to (3.13a-3.13d).

3.5 Unscented Kalman Filter Background

The UKF is an extension to the standard Kalman filter that utilizes a deterministic sampling approach known as the *unscented transform* (UT) to characterize states which undergo a *nonlinear* transformation. The UKF builds on the intuition that it is easier to approximate a probability distribution than to approximate an arbitrary nonlinear transformation [42].

Like the Kalman filter, the UKF includes a prediction step and an update/correction step. However, with the UKF, a state distribution is approximated by a Gaussian random variable (GRV) and specified using a minimal set of sample, or sigma, points around the mean. These sigma points are selected such that they capture the true mean and covariance of the GRV. When propagated through a nonlinear transform, the sigma points accurately capture the a posterior mean and covariance of the estimated state.

In other words, rather than simply passing the previous state estimate $\hat{x}^{k-1}$ through a nonlinear transform to produce a predicted state estimate $\hat{x}^k$, the UKF transforms the set of sigma points. The predicted state estimate $\hat{x}^k$ is then recovered as a weighted mean of the transformed points. With the UT, approximations of Gaussian states are accurate to the third order for any nonlinearities [100]. For non-Gaussian states, approximations are accurate to at least the second-order for any nonlinearities.

In this chapter, we employ the UKF algorithm as presented by Wan and van der Merwe [100, 98, 99] and summarized in the following section. Specifically, see Tables 7.3.1 and 7.3.2 in [98] for the algorithm employed in this work. We direct the reader to [42] for the original presentation of the UT and UKF. A discussion of dual estimation can be found in [100, 99].

In the following subsections, we summarize the UKF algorithm as presented by Wan and van der Merwe [98].

Sigma Points and Unscented Transform

To detail the UKF algorithm, we begin by describing the generation of sigma points and the execution of the unscented transform (UT). Consider a random variable $s \in \mathbf{R}^L$ with mean $\bar{s} \in \mathbf{R}^L$ and covariance $Q_s \in \mathbf{R}^{L\times L}$ that is propagated through a nonlinear function f such that $z = f(s)$ where $z \in \mathbf{R}^Z$. To calculate the statistics of z, we form a matrix $\mathcal{S} \in \mathbf{R}^{L\times(2L+1)}$ consisting of $2L+1$ sigma points $\mathcal{S}_i$ given by

$$\begin{aligned} \mathcal{S}_0 &= \bar{s} \\ \mathcal{S}_i &= \bar{s} + \left(\sqrt{(L+\lambda)Q_s}\right)_i, \quad i = 1,\dots,L \\ \mathcal{S}_{i+L} &= \bar{s} - \left(\sqrt{(L+\lambda)Q_s}\right)_i, \quad i = 1,\dots,L \end{aligned} \tag{3.15}$$

where $\left(\sqrt{(L+\lambda)Q_s}\right)_i$ is the ith column of the matrix square root of $(L+\lambda)Q_s$ and $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter. Constant α determines the spread of the sigma points (usually $10^{-4} \le \alpha \le 1$) and constant κ is a secondary scaling parameter (usually $\kappa = 0$ or $3 - L$).

The sigma points are propagated through the nonlinear function

$$\mathcal{Z}_i = f(\mathcal{S}_i) \quad i = 0,\dots,2L \tag{3.16}$$

and the mean and covariance of z are approximated as a weighted mean and covariance of the a posterior sigma points

$$\bar{z} \approx \sum_{i=0}^{2L} \mathcal{W}_{m,i}\mathcal{Z}_i \tag{3.17}$$

$$Q_z \approx \sum_{i=0}^{2L} \mathcal{W}_{c,i}(\mathcal{Z}_i - \bar{z})(\mathcal{Z}_i - \bar{z})^T \tag{3.18}$$

with weights $\mathcal{W}_m$, corresponding to the a posterior mean of the sigma points, given by

$$\begin{aligned} \mathcal{W}_{m,0} &= \lambda/(L+\lambda) \\ \mathcal{W}_{m,i} &= \lambda/(2(L+\lambda)), \quad i = 1,\dots,2L \end{aligned} \tag{3.19}$$

and weights $\mathcal{W}_c$, corresponding to the a posterior covariance of the sigma points, given by

$$\begin{aligned} \mathcal{W}_{c,0} &= \lambda/(L+\lambda) + (1-\alpha+\beta) \\ \mathcal{W}_{c,i} &= \mathcal{W}_{m,i}, \quad i = 1, \dots, 2L \end{aligned} \tag{3.20}$$

where constant β incorporates prior knowledge of the distribution of s (for Gaussian distributions, $\beta = 2$ is optimal).

To simplify notation, we will denote the generation of sigma points and the execution of the unscented transform ((3.15)-(3.18)) with the following operator expressions

$$\mathcal{S} = UT_s(\bar{s}, Q_s) \tag{3.21a}$$

$$\mathcal{Z}_i = f(\mathcal{S}_i) \quad i = 0, \dots, 2L \tag{3.21b}$$

$$\bar{z} = UT_m(\mathcal{Z}) \tag{3.21c}$$

$$Q_z = UT_c(\mathcal{Z}) \tag{3.21d}$$

where (3.21a) corresponds to (3.15), (3.21c) to (3.17), and (3.21d) to (3.18).

Unscented Kalman Filter Algorithm

The unscented Kalman filter (UKF) is a straightforward application of the UT to recursive estimation. To present the UKF algorithm, we will consider the state estimation of a discrete-time nonlinear dynamic system with non-additive noise given by the state-space model

$$x^k = F(x^{k-1}, u^k, v^k) \tag{3.22a}$$

$$y^k = H(x^k, u^k, w^k) \tag{3.22b}$$

where variable $x^k \in \mathbf{R}^X$ is the state of the system, $u^k \in \mathbf{R}^U$ is the known exogenous input, and $y^k \in \mathbf{R}^Y$ is the observed measurement signal. Function F is the transition model and the process noise $v^k \in \mathbf{R}^X$ has covariance $Q_v \in \mathbf{R}^{X \times X}$, $v^k \sim N(0, Q_v)$. Function H is the observation model and the measurement noise $w^k \in \mathbf{R}^Y$ has covariance $Q_w \in \mathbf{R}^{Y \times Y}$, $w^k \sim N(0, Q_w)$.

The UKF will model x^k as a GRV with estimated mean $\hat{x}^k$ and covariance Q_x^k. At each time step k, the UKF will generate sigma points for the previous state estimate, $\hat{x}^{k-1|k-1}$. For systems with non-additive noise, the state estimate and covariance is *augmented* with the process and measurement noise, as given by

$$\bar{s}^{k-1} = \begin{bmatrix} \hat{x}^{k-1|k-1} \\ \bar{v}^k \\ \bar{w}^k \end{bmatrix} \tag{3.23a}$$

$$Q_s^{k-1} = \begin{bmatrix} Q_x^{k-1|k-1} & 0 & 0 \\ 0 & Q_v & 0 \\ 0 & 0 & Q_w \end{bmatrix} \tag{3.23b}$$

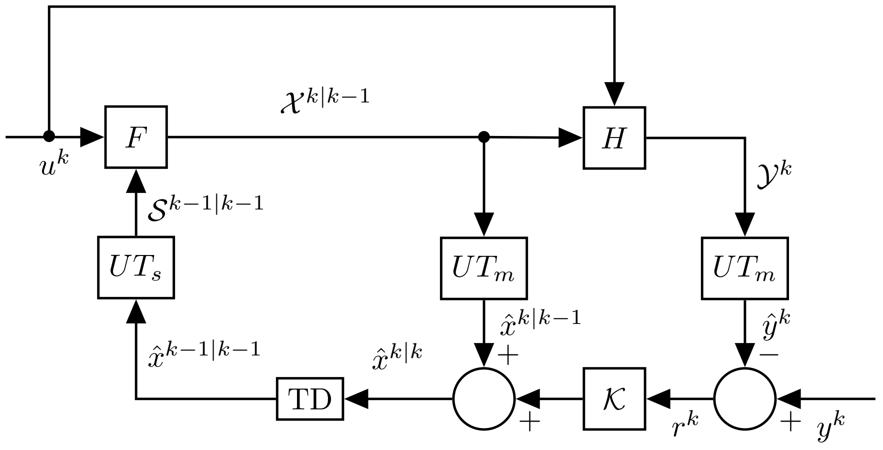

Figure 3.2: Additive Unscented Kalman Filter Diagram

where $\bar{v}^k$ and $\bar{w}^k$ are the mean of the process and measurement noises, respectively. In other words, if Gaussian, $\bar{v}^k \in \{0\}^X$ and $\bar{w}^k \in \{0\}^Y$. The dimensionality of $\bar{s}$ is therefore $L = 2X + Y$.

Next, the UKF will generate the sigma points. Because we are using the augmented state, we will introduce $\mathcal{S}_x$, $\mathcal{S}_v$, and $\mathcal{S}_w$, the sigma points associated with the state estimate, process noise, and measurement noise, respectively.

$$\mathcal{S}^{k-1} = UT_s(\bar{s}^{k-1}, Q_s^{k-1}) \tag{3.24a}$$
$$\mathcal{S}_{x,j}^{k-1} = \mathcal{S}_j^{k-1} \quad j = 0, \ldots, X-1 \tag{3.24b}$$
$$\mathcal{S}_{v,j}^{k-1} = \mathcal{S}_j^{k-1} \quad j = X, \ldots, 2X-1 \tag{3.24c}$$
$$\mathcal{S}_{w,j}^{k-1} = \mathcal{S}_j^{k-1} \quad j = 2X, \ldots, 2X+Y-1 \tag{3.24d}$$

where j refers to the rows of the L by $2L+1$ matrix $\mathcal{S}$.

In the prediction step, the UKF will propagate the sigma points through the process model and generate the a priori sigma points $\mathcal{X}_i^{k|k-1}$, state estimate $\hat{x}^{k|k-1}$, and covariance $Q_x^{k|k-1}$ as follows,

$$\mathcal{X}_i^{k|k-1} = F(\mathcal{S}_{x,i}^{k-1}, u^k, \mathcal{S}_{v,i}^{k-1}) \quad i = 0, \ldots, 2L \tag{3.25a}$$
$$\hat{x}^{k|k-1} = UT_m(\mathcal{X}^{k|k-1}) \tag{3.25b}$$
$$Q_x^{k|k-1} = UT_c(\mathcal{X}^{k|k-1}) \tag{3.25c}$$

In the correction step, the UKF will propagate the a priori sigma points through the measurement model to generate the measurement sigma points $\mathcal{Y}_i^k$, estimate $\hat{y}^k$, and covariance

Q_y as follows,

$$\mathcal{Y}_i^k = H(\mathcal{X}_i^{k|k-1}, u^k, \mathcal{S}_{w,i}^{k-1}) \quad i = 0, \ldots, 2L \tag{3.26a}$$

$$\hat{y}^k = UT_m(\mathcal{Y}^k) \tag{3.26b}$$

$$Q_y = UT_c(\mathcal{Y}^k) \tag{3.26c}$$

These are used to calculate the cross-covariance Q_{xy}, the Kalman gain $\mathcal{K}$, and the observation error r^k. Finally, the state estimate and covariance are corrected, producing the a posterior estimate $\hat{x}^{k|k}$ and covariance $Q_x^{k|k}$.

$$Q_{xy} = \sum_{i=0}^{2L} \mathcal{W}_{c,i}(\mathcal{X}_i^{k|k-1} - \hat{x}^{k|k-1})(\mathcal{Y}_i^k - \hat{y}^k)^T \tag{3.27a}$$

$$\mathcal{K} = Q_{xy} Q_y^{-1} \tag{3.27b}$$

$$r^k = y^k - \hat{y}^k \tag{3.27c}$$

$$\hat{x}^{k|k} = \hat{x}^{k|k-1} + \mathcal{K} r^k \tag{3.27d}$$

$$Q_x^{k|k} = Q_x^{k|k-1} - \mathcal{K} Q_y \mathcal{K}^T \tag{3.27e}$$

where $r^k \in \mathbf{R}$ is the error between the measurement y^k and the estimate $\hat{y}^k$ at time step k.

To simplify notation in this chapter, we will express the non-additive unscented Kalman filter algorithm with augmented state using the following 4 operator expressions

$$\begin{bmatrix} \mathcal{S}_x^{k-1} \\ \mathcal{S}_v^{k-1} \\ \mathcal{S}_w^{k-1} \end{bmatrix} = UKF_s\left(\begin{bmatrix} \hat{x}^{k-1|k-1} \\ Q_x^{k-1|k-1} \end{bmatrix}, \begin{bmatrix} \bar{v}^k \\ Q_v \end{bmatrix}, \begin{bmatrix} \bar{w}^k \\ Q_w \end{bmatrix}\right) \tag{3.28a}$$

$$\begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \\ \mathcal{X}^{k|k-1} \end{bmatrix} = UKF_x\left(F, \mathcal{S}_x^{k-1}, u^k, \mathcal{S}_v^{k-1}\right) \tag{3.28b}$$

$$\begin{bmatrix} \hat{y}^k \\ Q_y^k \\ \mathcal{Y}^k \end{bmatrix} = UKF_y(H, \mathcal{X}^{k|k-1}, u^k, \mathcal{S}_w^{k-1}) \tag{3.28c}$$

$$\begin{bmatrix} \hat{x}^{k|k} \\ Q_x^{k|k} \\ r^k \end{bmatrix} = UKF_c\left(\begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \\ \mathcal{X}^{k|k-1} \end{bmatrix}, \begin{bmatrix} \hat{y}^k \\ Q_y^k \\ \mathcal{Y}^k \end{bmatrix}, y^k\right) \tag{3.28d}$$

where (3.28a) corresponds to (3.23-3.24), (3.28b) to (3.25), (3.28c) to (3.26), and (3.28d) to (3.27).

Additive Unscented Kalman Filter

Consider the discrete-time nonlinear dynamic system with additive noise,

$$x^k = F(x^{k-1}, u^k) + v^k \tag{3.29a}$$

$$y^k = H(x^k, u^k) + w^k \tag{3.29b}$$

In this (very common) case, the UKF algorithm can be simplified. Specifically, $\bar{s}^{k-1} = \hat{x}^{k-1|k-1}$, $Q_s^{k-1} = Q_x^{k-1|k-1}$, and there are no sigma points for the process and measurement noise. This reduces the computational complexity of each iteration of the UKF from $O((2X+Y)^3)$ to $O(X^3)$ where X is the dimensionality of the state space and Y the dimensionality of the observation space. The complete additive UKF algorithm for model (3.29) is therefore,

$$\mathcal{S}_x^{k-1} = UT_s(\hat{x}^{k-1|k-1}, Q_x^{k-1|k-1}) \tag{3.30a}$$

$$\mathcal{X}_i^{k|k-1} = F(\mathcal{S}_{x,i}^{k-1}, u^k) \quad i = 0, \ldots, 2X \tag{3.30b}$$

$$\hat{x}^{k|k-1} = UT_m(\mathcal{X}^{k|k-1}) \tag{3.30c}$$

$$Q_x^{k|k-1} = UT_c(\mathcal{X}^{k|k-1}) + Q_v \tag{3.30d}$$

$$\mathcal{Y}_i^k = H(\mathcal{X}_i^{k|k-1}, u^k) \quad i = 0, \ldots, 2X \tag{3.30e}$$

$$\hat{y}^k = UT_m(\mathcal{Y}^k) \tag{3.30f}$$

$$Q_y = UT_c(\mathcal{Y}^k) + Q_w \tag{3.30g}$$

$$Q_{xy} = \sum_{i=0}^{2L} \mathcal{W}_{c,i}(\mathcal{X}_i^{k|k-1} - \hat{x}^{k|k-1})(\mathcal{Y}_i^k - \hat{y}^k)^T \tag{3.30h}$$

$$\mathcal{K} = Q_{xy} Q_y^{-1} \tag{3.30i}$$

$$r^k = y^k - \hat{y}^k \tag{3.30j}$$

$$\hat{x}^{k|k} = \hat{x}^{k|k-1} + \mathcal{K} r^k \tag{3.30k}$$

$$Q_x^{k|k} = Q_x^{k|k-1} - \mathcal{K} Q_y \mathcal{K}^T \tag{3.30l}$$

Figure 3.2 illustrates the additive UKF algorithm. The block TD represents a time delay (commonly denoted in controls literature by z^{-1} or $1/z$, the Z-transform of the delay operator). To simplify notation in this chapter, we will express the additive unscented Kalman filter algorithm using the following 4 operator expressions

$$\begin{bmatrix} \mathcal{S}_x^{k-1} \end{bmatrix} = UKF_s^+ \left(\begin{bmatrix} \hat{x}^{k-1|k-1} \\ Q_x^{k-1|k-1} \end{bmatrix} \right) \tag{3.31a}$$

$$\begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \\ \mathcal{X}^{k|k-1} \end{bmatrix} = UKF_x^+ \left(F, \mathcal{S}_x^{k-1}, u^k, Q_v \right) \tag{3.31b}$$

$$\begin{bmatrix} \hat{y}^k \\ Q_y^k \\ \mathcal{Y}^k \end{bmatrix} = UKF_y^+ (H, \mathcal{X}^{k|k-1}, u^k, Q_w) \tag{3.31c}$$

$$\begin{bmatrix} \hat{x}^{k|k} \\ Q_x^{k|k} \\ r^k \end{bmatrix} = UKF_c^+ \left(\begin{bmatrix} \hat{x}^{k|k-1} \\ Q_x^{k|k-1} \\ \mathcal{X}^{k|k-1} \end{bmatrix}, \begin{bmatrix} \hat{y}^k \\ Q_y^k \\ \mathcal{Y}^k \end{bmatrix}, y^k \right) \tag{3.31d}$$

where (3.31a) corresponds to (3.30a), (3.31b) to (3.30b-3.30d), (3.31c) to (3.30e-3.30g), and (3.31d) to (3.30h-3.30l).

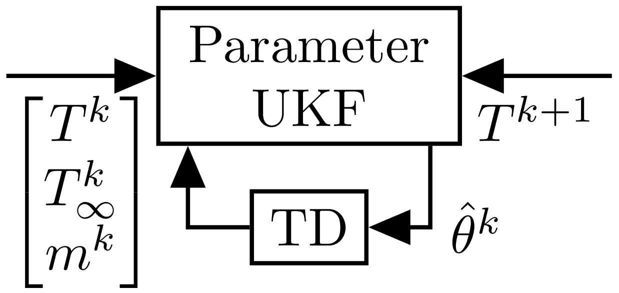

Figure 3.3: Single Filter Method Diagram

3.6 Recursive TCL Parameter Estimation

In this section, we will present 4 closely related approaches for parameter estimation of a thermostatically controlled load (TCL) using the Kalman filter (KF) algorithm in (3.14) and unscented Kalman filter (UKF) algorithm in (3.31). In this chapter, we will consider: (i) a single filter approach in which one UKF is used to estimate the parameters θ^k; (ii) a joint filter approach in which one UKF simultaneously estimates both θ^k and T^k; (iii) a dual filter approach in which one UKF estimates the parameters θ^k and one KF estimates the state T^k; and (iv) a triple filter approach in which we use one UKF to estimate θ^k, one KF to estimate T^k, and another KF to estimate the inputs, T^k_∞ and m^k.

In each case, we define the function G according to the TCL model (3.7)

$$\begin{aligned} T^{k+1} &= \theta_1^k T^k + \begin{bmatrix} 1-\theta_1^k & 1-\theta_1^k\theta_2^k & \theta_3^k \end{bmatrix} \begin{bmatrix} T^k_\infty \\ m^k \\ 1 \end{bmatrix} \\ &= G(T^k, T^k_\infty, m^k, \theta^k) \end{aligned} \tag{3.32}$$

Single Filter Parameter Estimation

Using the function G given in (3.32), the TCL recursive parameter estimation problem can be expressed with the state-space model

$$\theta^k = \theta^{k-1} + n^k \tag{3.33a}$$

$$y^k = G(T^k, T^k_\infty, m^k, \theta^k) + v^k + w^k \tag{3.33b}$$

where (3.33b) combines (3.32) with observation model $y^k = T^{k+1} + w^k$.

Figure 3.3 illustrates the single filter method. The block TD represents a time delay (commonly denoted $1/z$, the Z-transform of the delay operator). By employing the additive UKF algorithm in (3.31) with the T^{k+1} observation as y^k, we can produce $\hat{\theta}^k$, an estimate of the model parameters at time step k. Note that in the single filter case, θ corresponds to x, the variable being estimated, and G to H, the observation model. Additionally, the

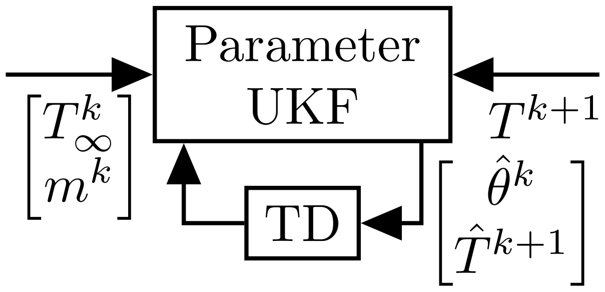
Parameter UKF
$\begin{bmatrix} T_\infty^k \\ m^k \end{bmatrix}$
T^{k+1}
$\begin{bmatrix} \hat{\theta}^k \\ \hat{T}^{k+1} \end{bmatrix}$
TD

Figure 3.4: Joint Filter Method Diagram

transition model F is given by $F(x^{k-1}, u^k) = x^{k-1}$ and T^k, T_∞^k, and m^k are effectively u^k, control and feed-through inputs at time step k.

Joint Filter State and Parameter Estimation

For system identification, it is often necessary to simultaneous perform state and parameter estimation from noisy observations [100]. There are two basic approaches, joint and dual estimation. In the joint estimation method, state and parameter estimation can be performed simultaneously with a single filter by estimating the state-space model

$$\begin{bmatrix} \theta^k \\ T^{k+1} \end{bmatrix} = \begin{bmatrix} \theta^{k-1} \\ G(T^k, T_\infty^k, m^k, \theta^{k-1}) \end{bmatrix} + \begin{bmatrix} n^k \\ v^k \end{bmatrix} \tag{3.34a}$$

$$y^k = T^{k+1} + w^k \tag{3.34b}$$

Figure 3.4 illustrates the joint filter method where the block TD represents a time delay. Just as in the single filter method, we employ the additive UKF algorithm in (3.31) with the T^{k+1} observation as y^k to recursively estimate the model. However, in the joint filter method, we produce estimates of both the state and the parameters ($\hat{T}^{k+1}$ and $\hat{\theta}^k$, respectively).

Dual Filter State and Parameter Estimation

In the dual estimation method, a separate state-space representation is used for the states and parameters. For a TCL, the state model is given by

$$T^{k+1} = G(T^k, T_\infty^k, m^k, \theta^k) + v^k \tag{3.35a}$$

$$y^k = T^{k+1} + w^k \tag{3.35b}$$

and the parameter model by (3.33).

Figure 3.5 illustrates the dual filter method where the block TD represents a time delay. Because the function G is linear in the states, we can estimate the state model (3.35) using the KF algorithm in (3.14) with the T^{k+1} observation as y^k to produce $\hat{T}^{k+1}$. Again, the

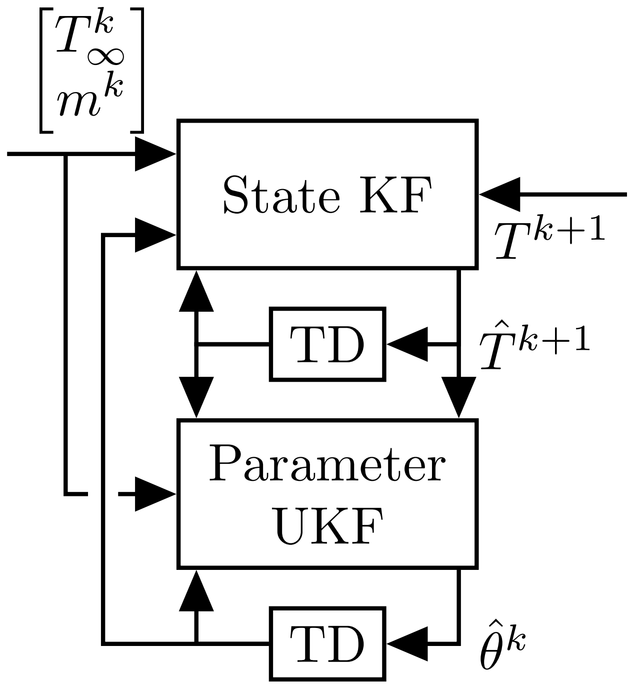

Figure 3.5: Dual Filter Method Diagram

parameter model (3.33) is estimated using the additive UKF algorithm in (3.31). We tie the two filters together by using the estimated state of one filter as the control input and/or observation in another filter. Specifically, for the state filter, we use the previous parameter estimate $\hat{\theta}^{k-1}$ in the transition model. For the parameter filter, we use the previous estimate $\hat{T}^k$ as input in the observation model and the current estimate $\hat{T}^{k+1}$ as the observation y^k (rather than the T^k and T^{k+1} observations, respectively).

Triple Filter Input, State, and Parameter Estimation

Lastly, we consider an estimation approach in which separate filters are used to estimate the inputs, states, and parameters. For simplicity and consistency, we will refer to this as the triple filter approach. The input model is given by

$$\begin{bmatrix} T_\infty^k \\ m^k \end{bmatrix} = \begin{bmatrix} T_\infty^{k-1} \\ m^{k-1} \end{bmatrix} + \begin{bmatrix} p_1^k \\ p_2^k \end{bmatrix} \tag{3.36a}$$

$$y^k = \begin{bmatrix} G(T^k, T_\infty^k, m^k, \theta^{k-1}) \\ T_\infty^k \\ m^k \end{bmatrix} + \begin{bmatrix} v^k + w^k \\ q_1^k \\ q_2^k \end{bmatrix} \tag{3.36b}$$

where $p \in \mathbf{R}^2$ and $q \in \mathbf{R}^2$ are process and measurement noises, respectively, associated with the inputs T_∞ and m. Again, the state model is given by (3.35) and the parameter model

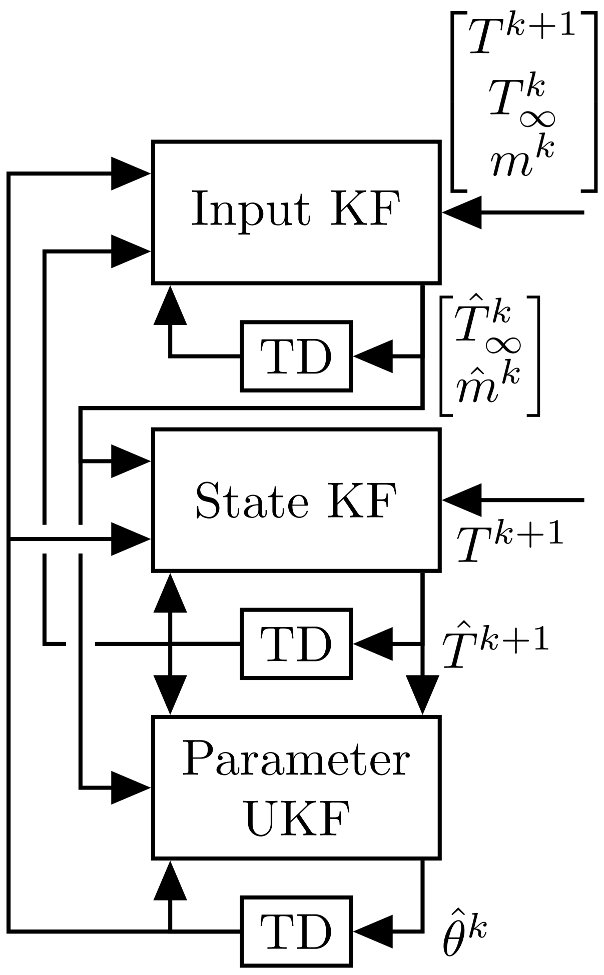
$\begin{bmatrix} T^{k+1} \\ T_\infty^k \\ m^k \end{bmatrix}$
Input KF
TD
$\begin{bmatrix} \hat{T}_\infty^k \\ \hat{m}^k \end{bmatrix}$
State KF
T^{k+1}
TD
$\hat{T}^{k+1}$
Parameter UKF
TD
$\hat{\theta}^k$

Figure 3.6: Triple Filter Method Diagram

by (3.33).

Figure 3.6 illustrates the triple filter method where the block TD represents a time delay. The input model (3.36) is estimated using the KF algorithm in (3.14) with the T^{k+1}, T_∞^k, and m^k observations as y^k to produce $\hat{T}_\infty^k$ and $\hat{m}^k$. To tie the three models together, the input estimates $\hat{T}_\infty^k$ and $\hat{m}^k$ are used in the transition model of the state filter and the observation model of the parameter filter. The previous parameter estimate $\hat{\theta}^{k-1}$ is used in the observation model of the input filter and the transition model of the state filter. Lastly, the state estimate $\hat{T}^k$ is used in the observation model of the input and parameter filters and $\hat{T}^{k+1}$ serves as the observation y^k in the parameter filter.

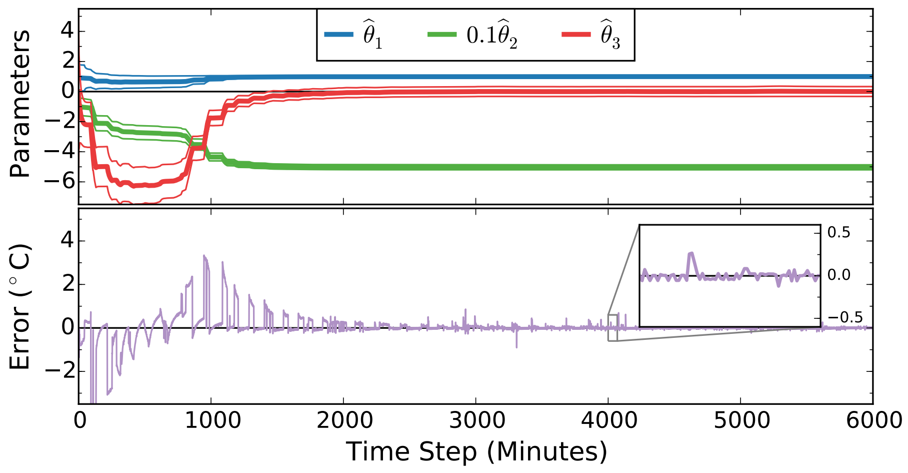

Figure 3.7: **Single Filter.** TCL_1 Parameter Estimation

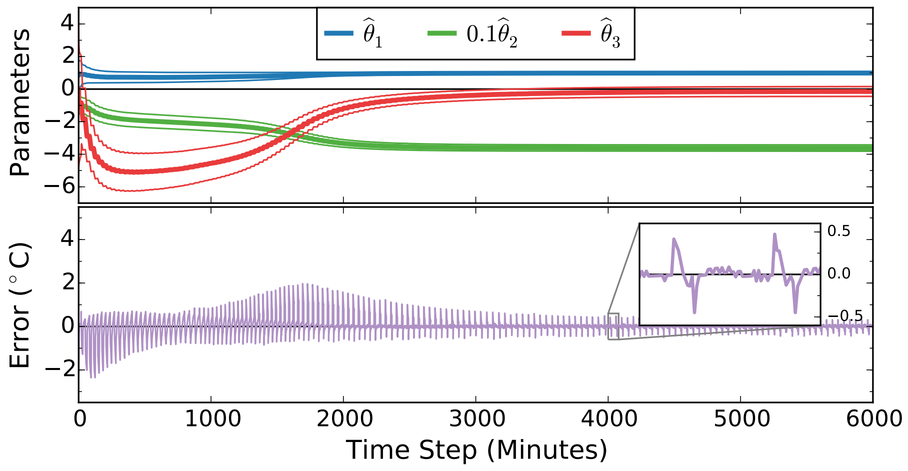

Figure 3.8: **Single Filter.** TCL_2 Parameter Estimation

3.7 TCL Estimation Experimental Results

In this section, we present parameter estimation results for TCL_1, a 500W residential refrigerator, and TCL_2, a 100W mini-fridge. Each TCL is instrumented with two DS18B20 digital temperature sensors to measure the ambient temperature T_∞ and internal refrigerator temperature T. The sensors have a -55°C to $+125$°C temperature range and a ±0.5°C accuracy from -10°C to $+85$°C. A current sensor is used to measure the state (on or off) of the compressor, m. Measurements were taken at 1 minute intervals for a period of 7 days. In this study, the temperature of the freezer is neither measured nor modeled.

TCL_1 is observed under typical operating conditions for a residential refrigerator. Therefore, the door is opened randomly and the contents of the refrigerator change regularly. By

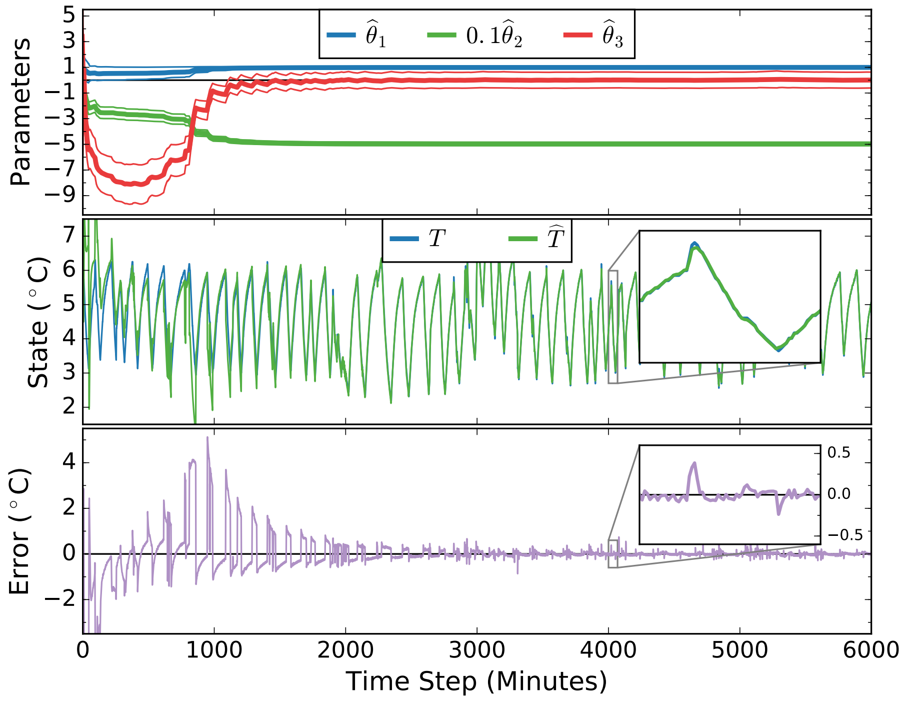

Figure 3.9: **Joint Filter.** TCL_1 Parameter Estimation

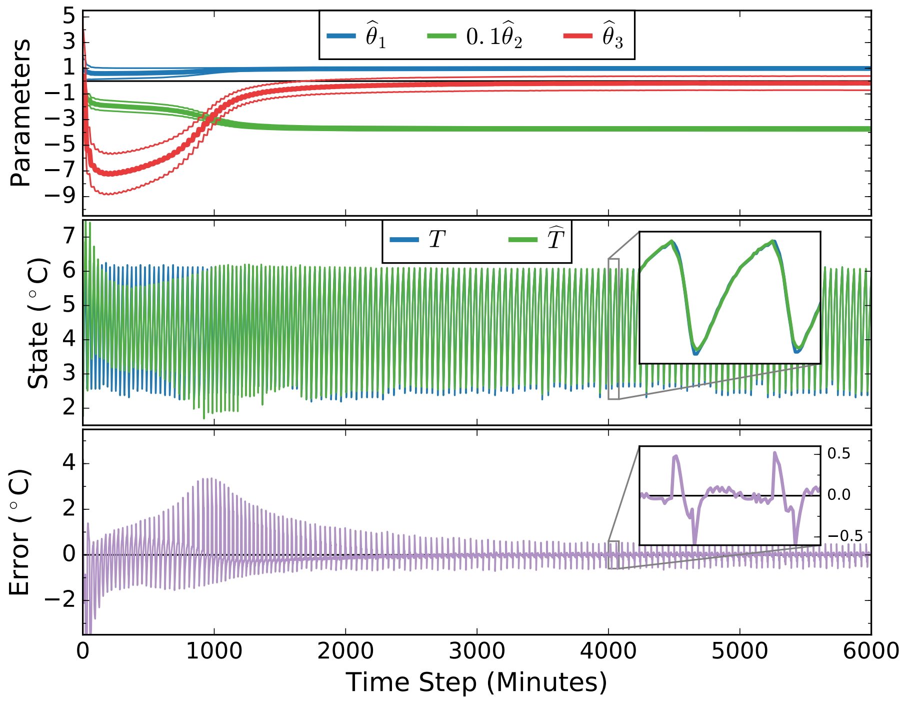

Figure 3.10: **Joint Filter.** TCL_2 Parameter Estimation

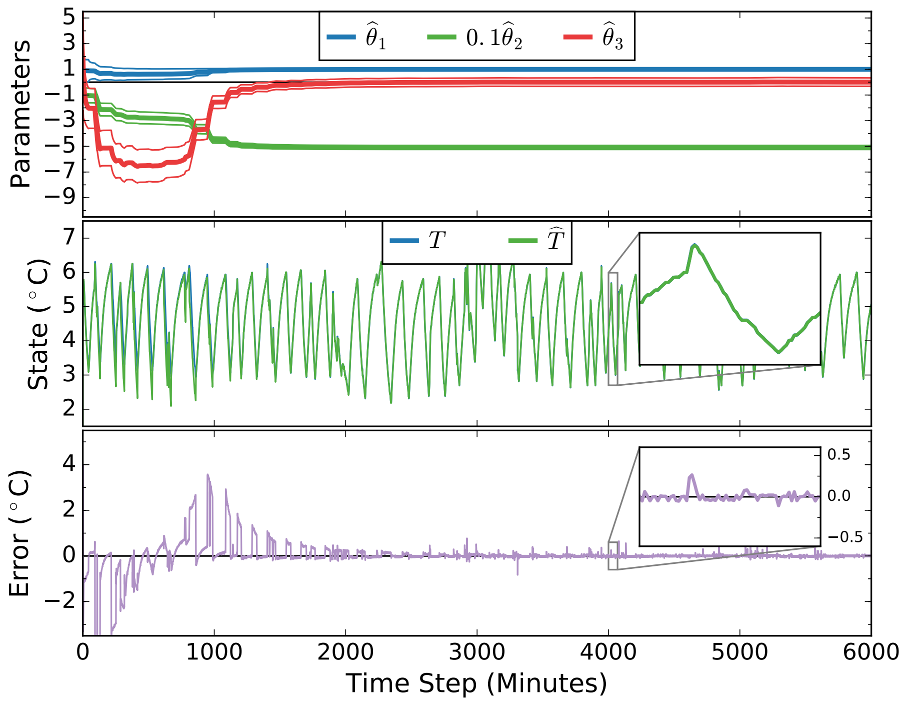

Figure 3.11: **Dual Filter.** TCL_1 Parameter Estimation

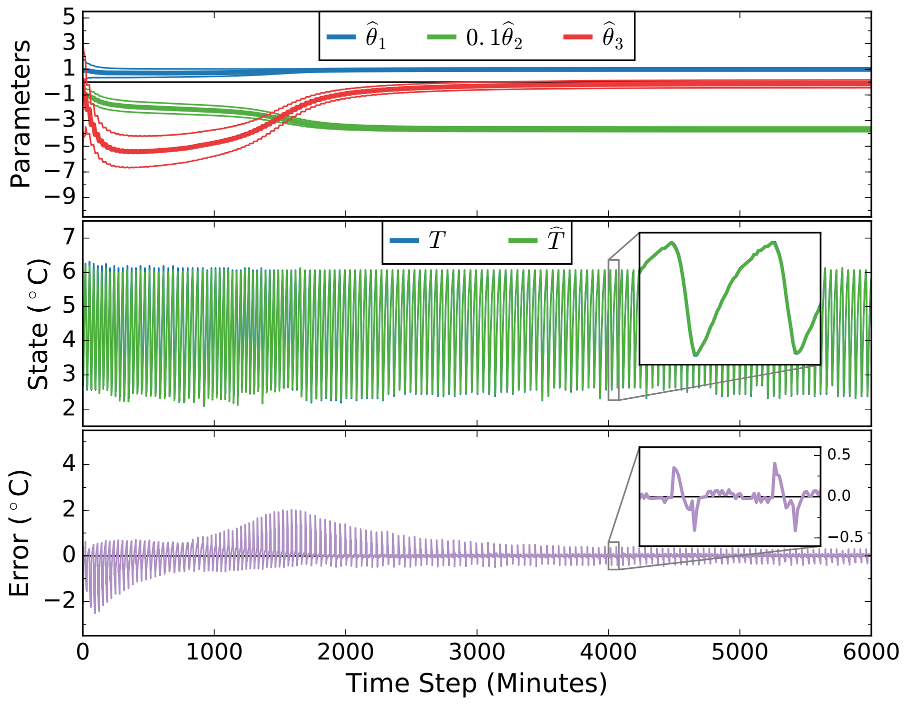

Figure 3.12: **Dual Filter.** TCL_2 Parameter Estimation

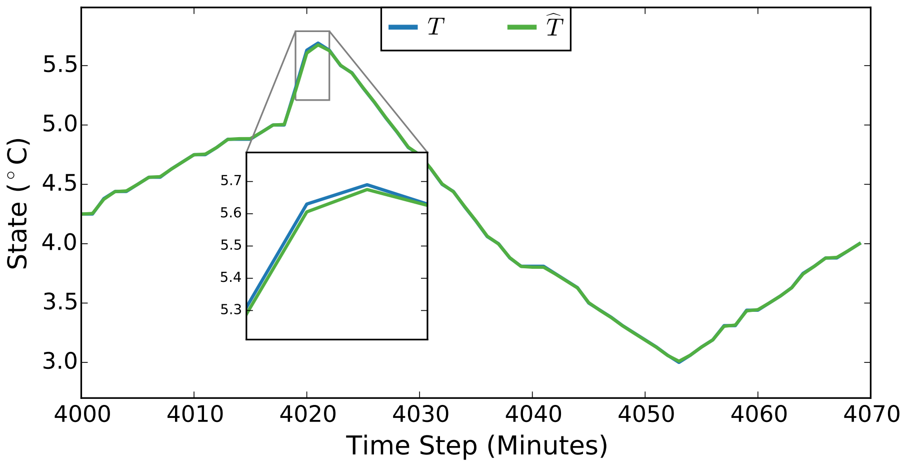

Figure 3.13: **Dual Filter.** TCL_1 Temperature State Estimate

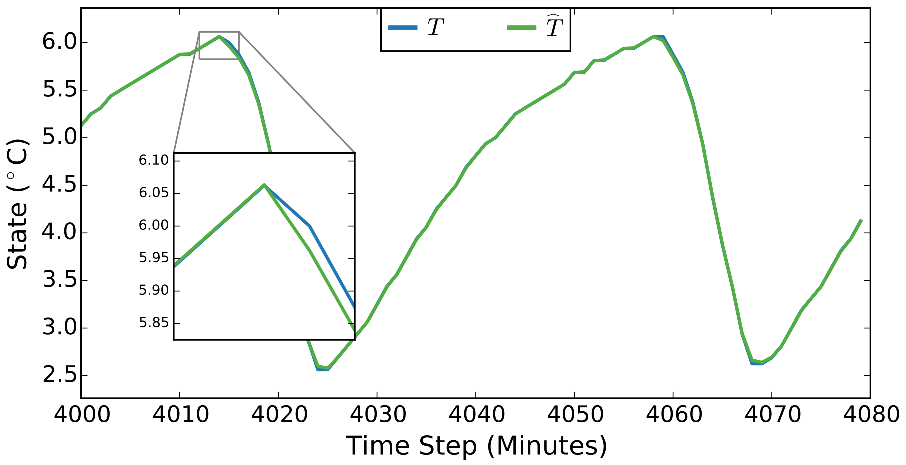

Figure 3.14: **Dual Filter.** TCL_2 Temperature State Estimate

contrast, TCL_2 is empty except for 18 liters of water, which compose the thermal mass being conditioned by the unit. The door of TCL_2 remains closed for the duration of the study.

For both TCLs, we have implemented four parameter estimation methods using the standard and unscented Kalman filters: single filter, joint filter, dual filter, and triple filter. The final parameter estimates $\hat{\theta}^f$ are presented in Table 3.1. Normally, for system identification, we would seek to measure the performance of each algorithm by first learning the parameters using a training dataset and then testing the parameters using a separate validation dataset. However, an advantage of recursive parameter estimation is that we can continuously improve the parameter estimates and potentially adapt to changes in the mechanical system. Therefore, for the single, dual, joint, and triple filter methods, we represent the performance as the root mean squared error (RMSE) over the last 300 time steps (i.e 5 hours). We

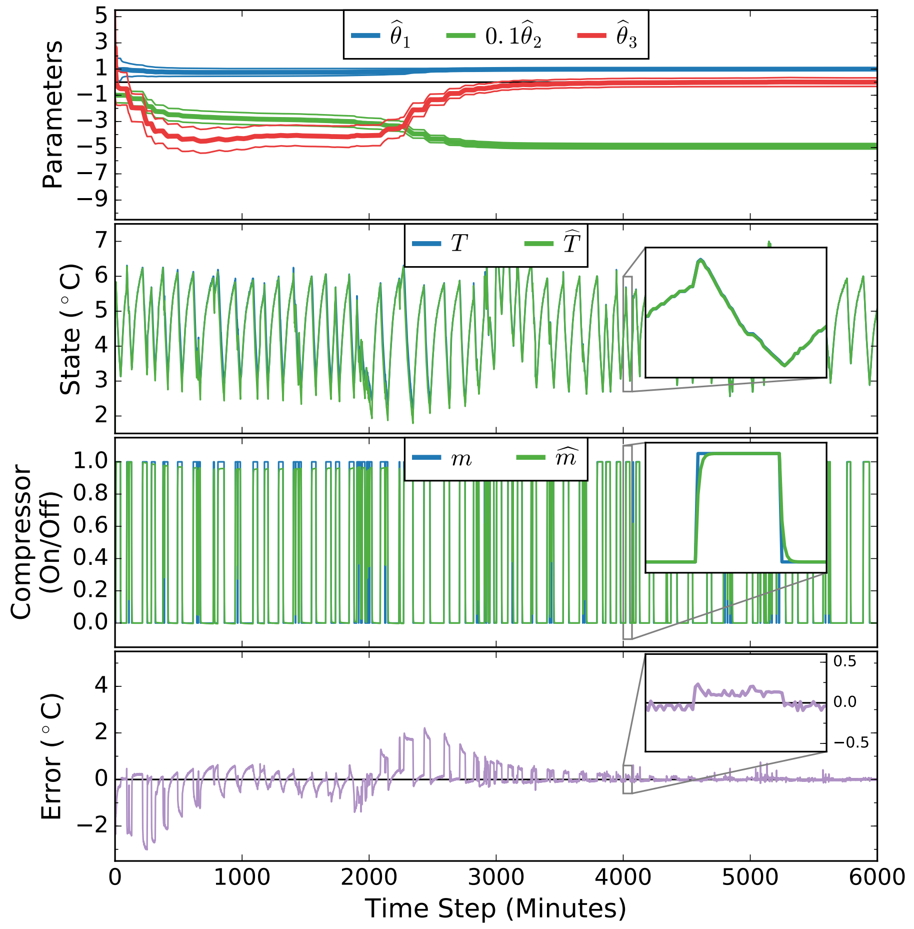

Figure 3.15: **Triple Filter.** TCL_1 Parameter Estimation

will denote this moving window RMSE as RMSE300. To measure the parameter estimation error, we employ the residual error r^k of the parameter filter for each of the four methods.

Figures 3.7 and 3.8 present the parameter estimation results using the single filter method. The top subplots show the parameter estimates $\hat{\theta}^k$ produced by the parameter filter at each time step k. For each parameter, the center line is the mean or expected value of the estimate and the top and bottom lines illustrate the variance relative to the mean. Eventually all parameter estimates converge and the variances decrease. The bottom subplots depict the residual error r^k. As shown, the parameter estimation for TCL_1 converges in about 1500 time steps while TCL_2 converges in about 3000 time steps.

Figures 3.9 and 3.10 present the parameter estimation results using the joint filter method and Figures 3.11 and 3.12 for the dual filter method. Again, the top subplots show the parameter estimates $\hat{\theta}^k$. The bottom subplots depict the residual error r^k. The state estimates $\hat{T}^k$ are presented in the center subplots. Both TCLs exhibit similar convergence characteristics compared to the single filter method. Upon convergence, the difference between the

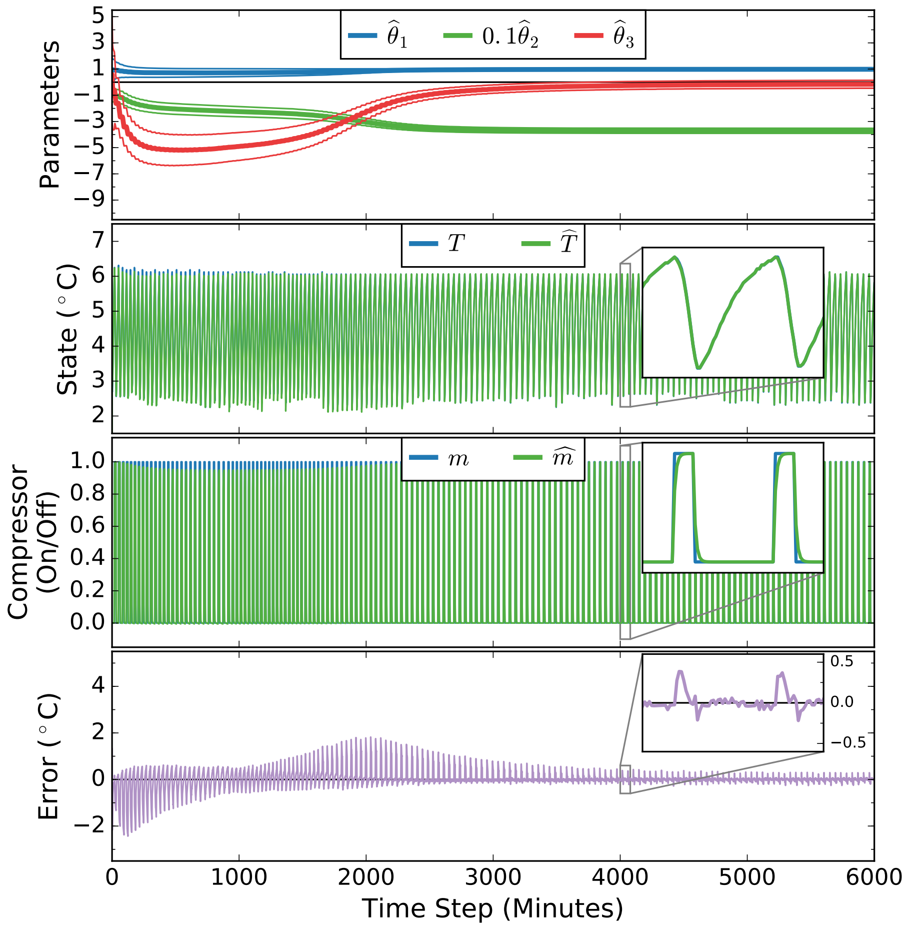

Figure 3.16: **Triple Filter.** TCL_2 Parameter Estimation

measured temperature T and estimated temperature $\hat{T}$ becomes negligible. Samples of the temperature estimates produced by the dual filter are shown in Figures 3.13 and 3.14.

As shown in Table 3.1, the differences in the final TCL_1 and TCL_2 parameter estimates for the single, joint, and dual filters are small enough to be considered negligible. Thus, with respect to parameter estimation, the joint and dual filters show little to no advantage over the single filter method. In other words, filtering the temperature measurement T does not appear to significantly improve the performance of our parameter estimation algorithm. Considering that we are estimating the parameters of two residential-sized refrigerators, this is not a surprising outcome. For a larger or noisier TCL, the joint and dual filters may yield a greater advantage, but for the TCLs used in this study, the estimation of T is simply unnecessary.

Figures 3.15 and 3.16 present the parameter estimation results using the triple filter method. In addition to the parameter estimate $\hat{\theta}^k$, state estimate $\hat{T}^k$, and parameter filter residual error r^k, the figure displays the compressor state estimates $\hat{m}^k$. The input filters

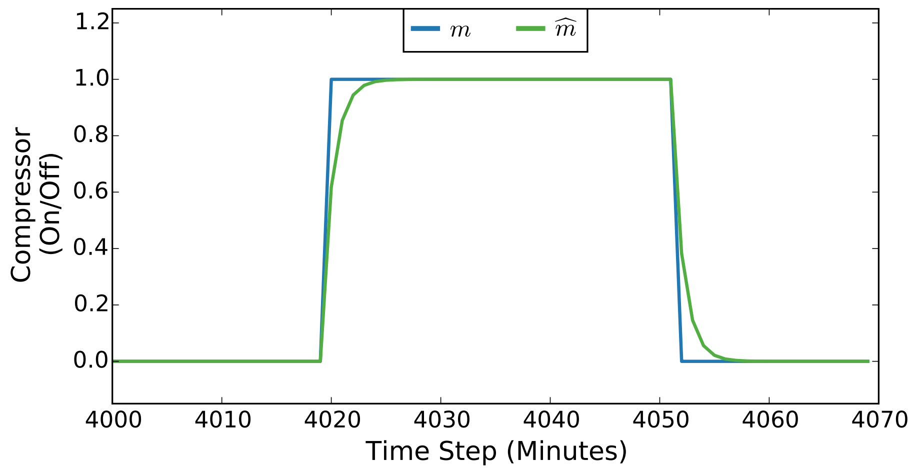

Figure 3.17: **Triple Filter.** TCL_1 Compressor State Estimate

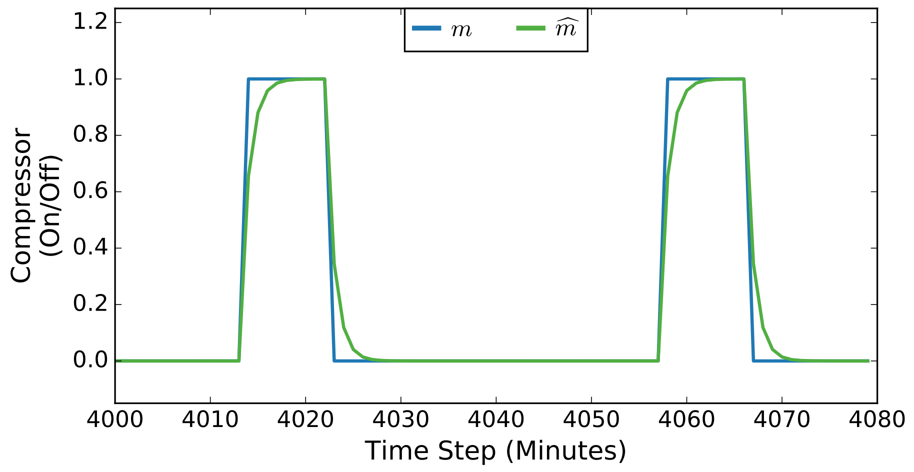

Figure 3.18: **Triple Filter.** TCL_2 Compressor State Estimate

	TCL_1			TCL_2		
	$\hat{\theta}_1^f$	$\hat{\theta}_2^f$	$\hat{\theta}_3^f$	$\hat{\theta}_1^f$	$\hat{\theta}_2^f$	$\hat{\theta}_3^f$
Single	0.998	-50.329	0.004	0.987	-36.193	-0.141
Joint	0.997	-49.762	0.005	0.985	-37.277	-0.159
Dual	0.998	-50.872	0.004	0.988	-36.822	-0.133
Triple	0.997	-52.268	0.005	0.986	-37.141	-0.153

Table 3.1: Final Parameter Estimates for TCL_1 and TCL_2

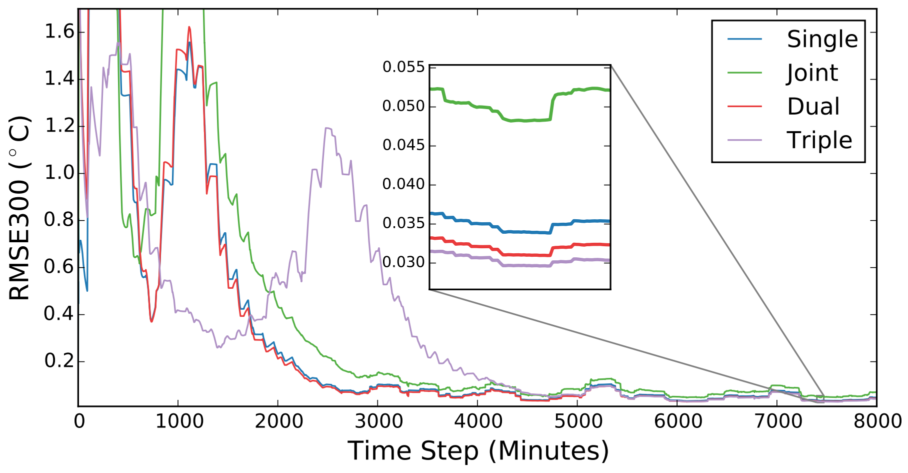

Figure 3.19: **Filter Error.** TCL_1 Parameter Estimation Moving Window RMSE

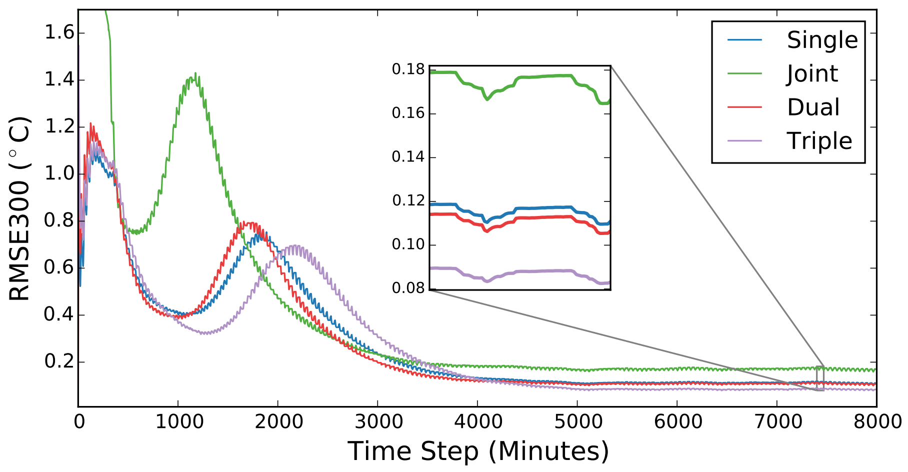

Figure 3.20: **Filter Error.** TCL_2 Parameter Estimation Moving Window RMSE

also produce ambient temperature estimates $\hat{T}_{\infty}^{k}$. However, due to the negligible difference between the ambient temperature observations and estimates, we have excluded the results from this chapter.

Sample of the compressor state estimates are plotted in Figures 3.17 and 3.18. As shown, the estimated compressor states $\hat{m}^k$ resemble a first-order system response rather than the discrete on/off state m^k given by the TCL model (3.7). One could argue that $\hat{m}^k$ better represents the thermodynamics of a TCL like the refrigerators used in this study. Specifically, while the compressor may instantaneously turn on or off (providing a step input), it takes some amount of time for the refrigeration cycle to start or stop removing heat (resulting in a first-order response). We could elect to model the refrigeration cycle as a first-order linear time-invariant system and to estimate the system parameters, however using the triple filter

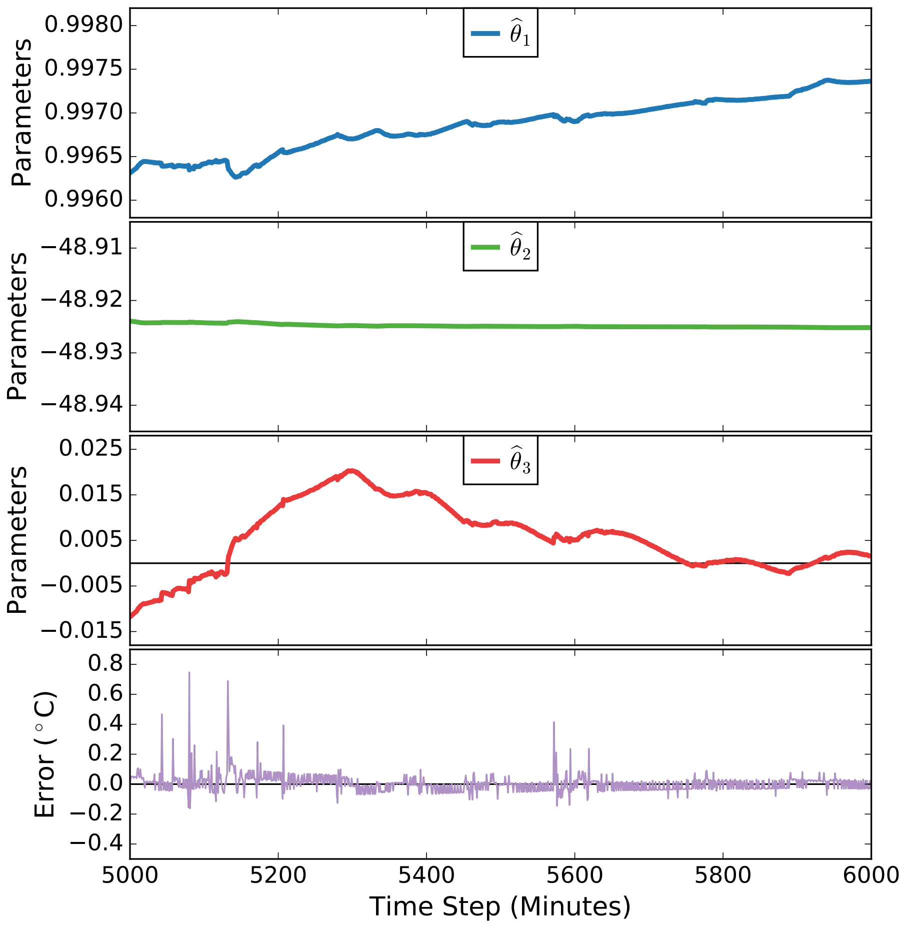

Figure 3.21: **Single Filter.** TCL_1 Adaptive Parameter Estimation

method to estimate m^k is a simple way to achieve a comparable result.

The moving window root mean square error (RMSE) results for each filter method are presented in Figures 3.19 and 3.20. As stated previously, the RMSE300 at each time step is a measure of the RMSE over the last 300 time steps (i.e 5 hours). In this way, the RMSE300 is a function of the parameter filter's residual error r^k but provides a clearer means of comparing the performance of each filter method. As shown, the triple filter is the slowest to converge but performs slightly better than the other methods. The single and joint filter performances are comparable in TCL_1 whereas the joint method has the worst performance for TCL_2. Overall, each filter method succeeds in performing parameter estimation and converges to comparable values. After convergence, the differences in the RMSE300 values are small enough to be considered negligible, suggesting that the refrigerators studied in this chapter are relatively low noise systems. The same results are not to be expected of larger or noisier TCLs. Nonetheless, this chapter presents a compiled collection of filtering methods for online learning of TCLs.

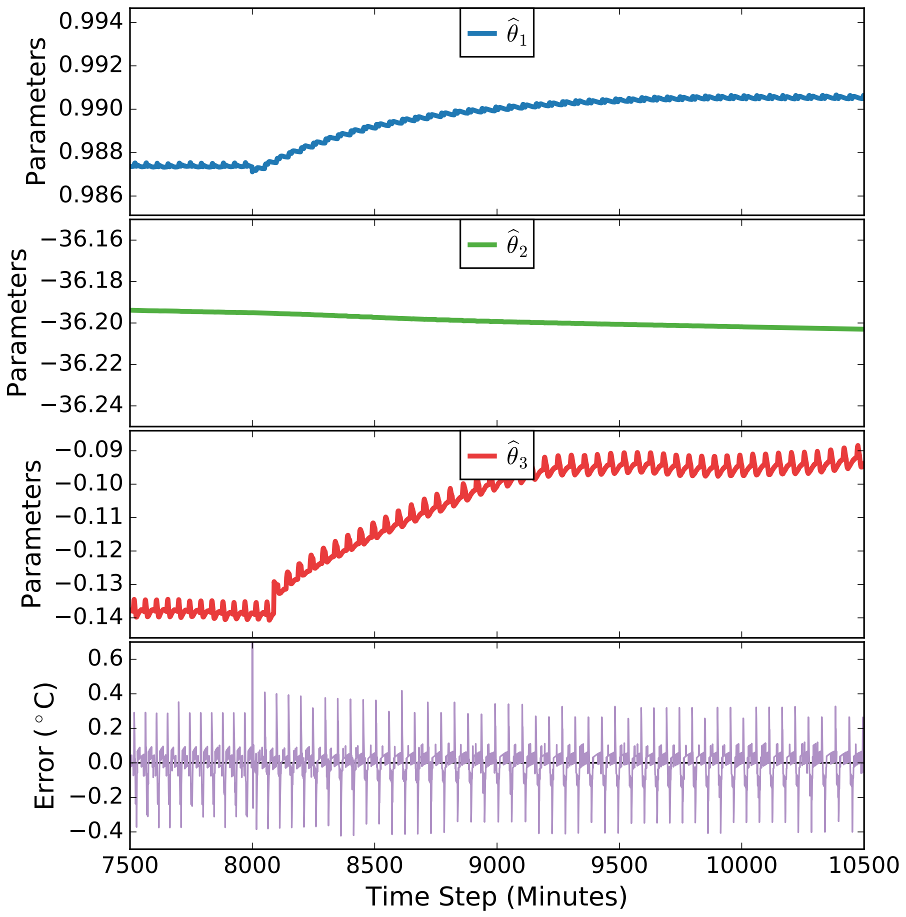

Figure 3.22: **Single Filter.** TCL_2 Adaptive Parameter Estimation

Utilizing a recursive system identification technique provides the added benefit of allowing the parameter estimates to continuously adapt to changes in the system. This point is illustrated in Figures 3.21 and 3.22. Because TCL_1 is subject to normal residential use, the capacitance of the refrigerator regularly changes by relatively small, random, and unobserved magnitudes. Figure 3.21 shows how the parameter estimates produced by the single filter method, particularly $\hat{\theta}_3$, respond to observations from the temperature sensors. In other words, the UKF updates the parameter estimates in order to minimize the residual error, thus allowing the model to dynamically adapt.

To more directly test the adaptive characteristics of our recursive system identification technique, we removed the 18 liters of water from TCL_2 at roughly time step $k = 8000$, thereby producing a step reduction in the thermal capacitance of the system. Figure 3.22 illustrates how the single UKF method responded to this step change by increasing both $\hat{\theta}_1$ and $\hat{\theta}_3$. According to the TCL model (3.7), we expect that decreasing the capacitance in the TCL increases $\hat{\theta}_1$, which represents the thermal characteristics of the conditioned

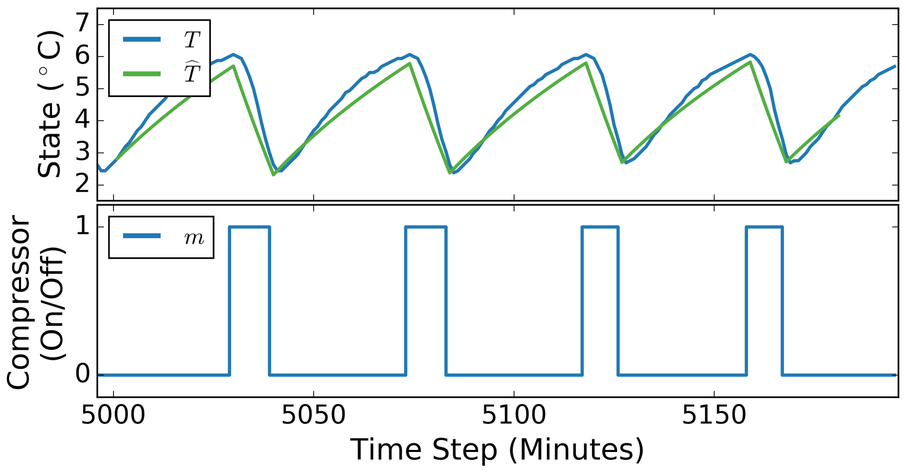

Figure 3.23: **Single Filter.** TCL_2 Forecast Example

mass ($\theta_1 = \exp(-h/RC)$). As stated previously, $\hat{\theta}_3$ represents a noise process accounting for energy gain or loss that is not directly modeled.

Figure 3.23 presents a 3 hour forecast $\hat{T}$ of the temperature T within TCL_2 using the model parameters as estimated at the start of the time horizon. As shown, the model is capable of accurately estimating the predominate dynamics within the TCL and therefore suitable for forecasting the temperature of the conditioned mass. However, the model is not capable of representing the higher order dynamics observed within each cycle. These dynamics could, in theory, be captured by a higher order model, but this would increase the model complexity and the need for temperature sensor measurements. In the next chapter, we present a model capable of approximating higher order dynamics using a piecewise linear approach.

In Appendix C, we present 2 simulation studies which employ the single UKF method presented in this chapter. The first study demonstrates the capability of the single UKF method to quickly converge to new parameter estimates in response to changes in the system dynamics. The second study presents simulation results for a population of refrigerators which optimize their power demand based on a demand response electricity price event. These studies show the advantage of using model predictive control with the single UKF method rather than employing a fixed set of model parameters.

3.8 Conclusions

This chapter examined online parameter estimation of thermostatically controlled loads (TCLs). We briefly discussed the Kalman filter (KF) and unscented Kalman filter (UKF) algorithms. Next, we presented four filter methods (single, joint, dual, and triple) for recursively estimating the parameters of a discrete-time thermostatically controlled load (TCL) model. Finally, we presented experimental results using real temperature data from a 500W and a 100W residential refrigerator. For each of the four filter methods, the algorithm successfully converged to comparable parameter estimates and adapted to changing TCL characteristics.

Chapter 4

Building Thermal Modeling

This chapter presents a piecewise linear thermal model of a building. To learn the model, a Kalman filter based approach for estimating the parameters is described. Experimental results employ data collected from a residential building with a forced-air heating and ventilation system to train and validate the piecewise model.

4.1 Motivation & Background

Heating, ventilation, and air-conditioning (HVAC) account for 43% of commercial and 54% of residential energy consumption [94]. Space heating alone accounts for 45% of all residential energy use. HVAC systems are an integral part of buildings responsible for regulating temperature, humidity, carbon dioxide, and airflow, conditions which directly impact occupant health and comfort. Estimates suggest that component upgrades and advanced HVAC control systems could reduce building energy usage by up to 30% [11]. Such intelligent systems can improve the efficiency of building operations, better regulate indoor conditions to improve air quality and occupant comfort, and enable buildings to participate in demand response services to improve power grid stability and reduce energy related carbon emissions [13].

To effectively control the operation of an HVAC system, it is essential that a model predictive controller incorporate an accurate mathematical representation of a building's thermal dynamics. The processes that determine the evolution of temperatures within a building are complex and uncertain. A reliable model improves the ability of a controller to forecast conditions and meet efficiency and comfort objectives. Simulation software, such as EnergyPlus and TRNSYS, is capable of high fidelity modelling of building HVAC systems. These mathematical models play a crucial role in the architectural and mechanical design of new buildings, however, due to high dimensionality and computational complexity, are not suitable for incorporation into HVAC control systems [6].

The American Society of Heating, Refrigeration, and Air-Conditioning Engineers (ASHRAE) handbook [30] describes how to determine the thermal resistance values of a building surface given it materials and construction type. However, for existing buildings, details about the materials in and construction of walls and windows may be difficult to obtain or non-existent [53]. Additionally, modifications to the building or changes brought about by time and use (e.g. cracks in windows or walls) further diminish the potential for characterizing a building based on design or construction information.

Therefore, an ideal control-oriented model would capture the predominant dynamics and disturbance patterns within a building, enable accurate forecasting, adapt to future changes in building use, provide a model structure suitable for optimization, and be amenable to real-time data-driven model identification methods. For these reasons, low order RC models are widely employed for control-oriented thermal building models [53, 1, 80]. Such models trade complexity and accuracy for simplicity and efficiency.

In this chapter, we present a piecewise linear RC model for the thermostatic control of buildings and a recursive Kalman filter method for parameter estimation. The piecewise model structure enables the approximate identification of unmodeled dynamics, in particular higher-order dynamics and time delays related to changes in the mechanical state of the system. By employing a recursive parameter estimation technique, we are able to perform online data-driven learning of the model.

We do not model heating from solar gain, building occupants, or equipment. This does not restrict the applicability of this work because the model structure can be extended for such cases. By estimating these effects with a single time-varying gain, we produce a simpler model better suited for predictive control.

Chapter Outline

This chapter is organized as follows. Section 4.2 presents our piecewise thermal model and Section 4.3 formulates a Kalman filter-based method for recursive parameter estimation of the piecewise model. Section 4.4 provides numerical examples of our proposed model and algorithm for the parameter estimation of an apartment with a forced-air heating and ventilation system. Finally, Section 4.5 summarizes key results.

4.2 Building Thermal Model

Linear Thermal Model

The thermostatically controlled load model (3.7a) can be expressed in a form that is linear in both the states and the parameters.

$$T^{k+1} = \theta_a T^k + (1 - \theta_a) T_\infty^k + \theta_b m^k + \theta_c \tag{4.1}$$

where $T^k \in \mathbf{R}$, $T_\infty^k \in \mathbf{R}$, and $m^k \in \{0, 1\}$ are the indoor air temperature (state, °C), outdoor air temperature (disturbance input, °C), and heater state (control input, On/Off), respectively.

The parameter θ_a corresponds to the thermal characteristics of the conditioned space as defined by $\theta_a = \exp(-\Delta t/RC)$, θ_b to the energy transfer due to the systems mechanical state as defined by $\theta_b = (1 - \exp(-\Delta t/RC))RP$, and θ_c to an additive process accounting for energy gain or loss not directly modeled.

As noted in [17, 69], the discrete time model implicitly assumes that all changes in mechanical state occur on the time steps of the simulation. In this chapter, we assume that this behavior reflects the programming of the systems being modeled. In other words, we assume that the thermostat has a sampling frequency of $1/(3600\Delta t)$ Hz or once per minute.

Piecewise Linear Thermal Model

The linear discrete time model (4.1) is capable of representing the predominant thermal dynamics within a conditioned space. Unfortunately, because it does not capture any higher-order dynamics or time delays related to changes in the mechanical state of the system, the model is fairly inaccurate in practice. Research into higher-order models, in particular multi-zone network models and the modeling of walls as 2R-1C or 3R-2C elements, have shown potential for producing higher fidelity building models [53, 1, 80]. However, this comes at the cost of increasing the model complexity and the need for temperature sensing (in particular, within interior and exterior walls).

In this chapter, we present a piecewise linear model capable of approximating dynamics related to changes in the mechanical state of the system. Our piecewise modelling approach is related to linear parameter-varying (LPV) systems which employ a linear model whose parameters change according to a time-varying state. This parameter dependency enables LPV systems to approximate nonlinear dynamics.

In our piecewise thermal model, the number of time steps since the system turned on or off serves as the time-varying state with which the parameters are determined. Specifically, we define N_a models for when the mechanical system is off ($m^k = 0$) and N_b model for when the mechanical system is on ($m^k = 1$). Each of the $i = 1, \dots, N_a$ and $j = 1, \dots, N_b$ submodels describe a particular range of time steps after the mechanical system has switched from an on to an off state or vice versa. When the system is off, we define the length of each range as δ_a, the number of ranges as N_a, and the number of time steps since the system was last on before switching off as λ_a (i.e. if $m^{k-1} = 1$ and $m^k = 0$ then $\lambda_a = 1$). Likewise, when the system is on, we define the length of each range as δ_b, the number of ranges as N_b, and the number of time steps since the system was last off as λ_b (i.e. if $m^{k-1} = 0$ and $m^k = 1$ then $\lambda_b = 1$). Thus, the piecewise thermal model is given by

$$T^{k+1} = \begin{cases} \theta_{a,1}T^k + (1-\theta_{a,1})T_\infty^k + \theta_{c,1} \\ \quad \text{if } m^k = 0 \\ \quad \text{and } \lambda_a \le \delta_a \\ \theta_{a,2}T^k + (1-\theta_{a,2})T_\infty^k + \theta_{c,2} \\ \quad \text{if } m^k = 0 \\ \quad \text{and } \delta_a < \lambda_a \le 2\delta_a \\ \qquad \vdots \\ \theta_{a,N_a}T^k + (1-\theta_{a,N_a})T_\infty^k + \theta_{c,N_a} \\ \quad \text{if } m^k = 0 \\ \quad \text{and } \lambda_a > (N_a-1)\delta_a \\ \theta_{a,N_a}T^k + (1-\theta_{a,N_a})T_\infty^k \\ + \theta_{c,N_a} + \theta_{b,1} \\ \quad \text{if } m^k = 1 \\ \quad \text{and } \lambda_b \le \delta_b \\ \theta_{a,N_a}T^k + (1-\theta_{a,N_a})T_\infty^k \\ + \theta_{c,N_a} + \theta_{b,2} \\ \quad \text{if } m^k = 1 \\ \quad \text{and } \delta_b < \lambda_b \le 2\delta_b \\ \qquad \vdots \\ \theta_{a,N_a}T^k + (1-\theta_{a,N_a})T_\infty^k \\ + \theta_{c,N_a} + \theta_{b,N_b} \\ \quad \text{if } m^k = 1 \\ \quad \text{and } \lambda_b > (N_b-1)\delta_b \end{cases} \tag{4.2}$$

where $\theta_{a,i}$ and $\theta_{c,i}$ are the parameters for the i-th model $i = 1, \ldots, N_a$ and $\theta_{b,j}$ is the parameter for the j-th model $j = 1, \ldots, N_b$. When the system is on, we employ the θ_{a,N_a} and θ_{c,N_a} parameters regardless of λ_a. In the following sections, we describe a recursive method for estimating the parameters in (4.2) using a Kalman filter.

4.3 Kalman Filter-based Parameter Estimation of the Piecewise Thermal Model

In this section, we present a parameter estimation method for the piecewise linear thermal model (4.2) using the Kalman filter (KF) algorithm. To begin, we consider the recursive parameter estimation problem for the linear model (4.1), which can be expressed with the state-space model

$$\theta^k = \theta^{k-1} + n^k \tag{4.3a}$$
$$y^k = \begin{bmatrix} T^k - T_\infty^k & m^k & 1 \end{bmatrix} \theta^k + T_\infty^k + q^k \tag{4.3b}$$

where (4.3b) combines (4.1) with the observation model $y^k = T^{k+1} + w^k$. The noise term $q^k \in \mathbf{R}$ has covariance $Q_q \in \mathbf{R}$, $q^k \sim N(0, Q_q)$, and represents the sum of the process noise v^k and the measurement noise w^k. The parameter update noise $n^k \in \mathbf{R}^3$ has covariance $Q_n \in \mathbf{R}^{3\times3}$, $n^k \sim N(0, Q_n)$.

If we model (4.3) using the Kalman filter with

$$\begin{aligned} A &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & B &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ C &= \begin{bmatrix} T^k - T_\infty^k & m^k & 1 \end{bmatrix} & D &= \begin{bmatrix} 1 \end{bmatrix} \\ u^k &= \begin{bmatrix} T_\infty^k \end{bmatrix} & y^k &= \begin{bmatrix} T^{k+1} \end{bmatrix} \end{aligned} \tag{4.4}$$

then the estimated state $\hat{\theta}^{k|k} \in \mathbf{R}^3$ will be a time-varying estimate of the thermal model parameters θ_a, θ_b, and θ_c. Note that the values of the C matrix will also be time-varying.

To learn the piecewise thermal model, we define a Kalman filter for each of the $N_a + N_b$ models in (4.2). When the mechanical system is off ($m^k = 0$), the state-space model is

$$\theta_i^k = \theta_i^{k-1} + n_i^k \tag{4.5a}$$
$$y^k = \begin{bmatrix} T^k - T_\infty^k & 1 \end{bmatrix} \theta_i^k + T_\infty^k + q_i^k \tag{4.5b}$$

and thus the filter models take the form

$$\begin{aligned} A &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & B &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ C &= \begin{bmatrix} T^k - T_\infty^k & 1 \end{bmatrix} & D &= \begin{bmatrix} 1 \end{bmatrix} \\ u^k &= \begin{bmatrix} T_\infty^k \end{bmatrix} & y^k &= \begin{bmatrix} T^{k+1} \end{bmatrix} \end{aligned} \tag{4.6}$$

for each of the $i = 1, \ldots, N_a$ models. Therefore, the estimated state $\hat{\theta}_i^{k|k} \in \mathbf{R}^2$ is a time-varying estimate of the thermal model parameters $\theta_{a,i}$ and $\theta_{c,i}$. Additionally, $n_i^k \in \mathbf{R}^2$ and $q_i^k \in \mathbf{R}$ are noise terms for each model.

When the mechanical system is on ($m^k = 1$), the state-space model is

$$\theta_j^k = \theta_j^{k-1} + n_j^k \tag{4.7a}$$
$$y^k = \theta_j^k + (T^k - T_\infty^k)\theta_{a,N_a} + T_\infty^k + \theta_{c,N_a} + q_j^k \tag{4.7b}$$

and the filter models are given by

$$\begin{aligned} A &= \begin{bmatrix} 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 \end{bmatrix} \\ C &= \begin{bmatrix} 1 \end{bmatrix} \quad D = \begin{bmatrix} 1 \end{bmatrix} \\ u^k &= \begin{bmatrix} \theta_{a,N_a}(T^k - T_\infty^k) + T_\infty^k + \theta_{c,N_a} \end{bmatrix} \\ y^k &= \begin{bmatrix} T^{k+1} \end{bmatrix} \end{aligned} \tag{4.8}$$

for $j = 1, \ldots, N_b$ where θ_{a,N_a} and θ_{c,N_a} are the current parameter estimates of model $i = N_a$. Therefore, the estimated state $\hat{\theta}_j^{k|k} \in \mathbf{R}$ is an estimate of θ_b. Additionally, $n_j^k \in \mathbf{R}$ and $q_j^k \in \mathbf{R}$ are noise terms for each model.

When training the models, the temperature measurement T^{k+1} at each time step is used as an observation to train only one of the $N_a + N_b$ submodels as given by (4.2). In this way, each submodel is learning to represent a particular characteristic of the thermal dynamics of the system. For the remainder of the submodels, the parameter estimates are unchanged (i.e. $\hat{\theta}_i^k = \hat{\theta}_i^{k-1}$ for each filter i where i is not the observed model).

With respect to the covariances $Q_\theta^{k|k}$, there are two ways of updating the matrices. The first is to set each covariance matrix to the previous value. This expresses that, even though we did not observe the model in the current time step, we have not lost confidence in the parameter estimates. Alternatively, we can add the process noise covariance (as done in the Kalman filter prediction step (3.11b)), expressing a increasing loss of confidence in the parameter values. In this chapter, we assume the former and only alter the covariance matrix when the model is observed.

4.4 Residential Heating System Parameter Estimation Experiments

In this section, we present parameter estimation results for an 850 sq ft apartment with a forced-air heating and ventilation system. The apartment is located in Berkeley, California and equipped with a custom thermostat designed and built for this research. Therefore, we are able to control the operation of the heating system and to measure the indoor air temperature. Local weather data, specifically ambient air temperature, is retrieved from the Internet service, Weather Underground [101].

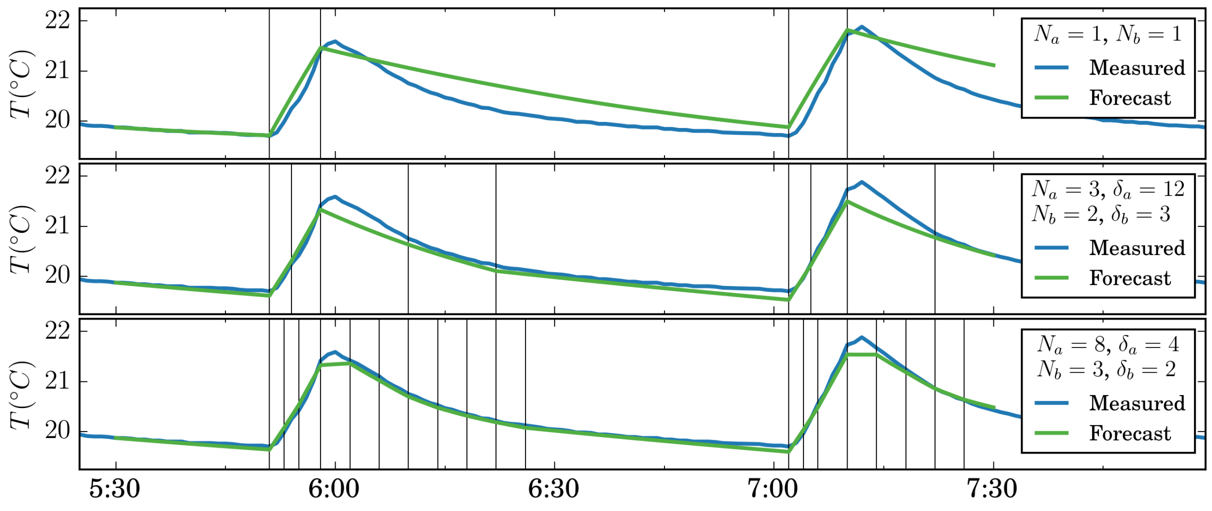

Figure 4.1: **Piecewise Thermal Models.** Examples with $N_a = N_b = 1$ (top), $N_a = 3$ and $N_b = 2$ (middle), and $N_a = 8$ and $N_b = 3$ (bottom) used to produce a 2 hour forecast.

Data was collect at a time-scale of one minute for 6 weeks during December and January of 2015-2016. With this data, we are able to perform recursive parameter estimation of the piecewise thermal model (4.2). The results presented in this section focus of quantifying and qualifying the advantages of the piecewise model and the Kalman filter based learning method.

Figure 4.1 presents a comparison of thermal models using varying numbers of N_a and N_b submodels. In each subplot, a 2 hour forecast is produced using the model parameters as estimated at the start of the time horizon. The vertical lines designate the start and end of each model's corresponding range. The top subplot shows the most basic case where $N_a = N_b = 1$, for a total of 2 submodels. As shown, the model is simply incapable of representing the evolution of the indoor air temperature. Most notably, the forecast poorly accounts for the thermal dynamics immediately after the heating system turns off. These dynamics are related to the interaction between the air and the other thermal masses (walls, furniture, etc.) within the conditioned space. These dynamics could, in theory, be captured by a higher order model, but this would increase the model complexity and the need for temperature sensor measurements.

By increasing the number of submodels, as shown in the second and third subplots of Figure 4.1, the piecewise thermal model is able to better approximate the dynamics of the apartment and heating system without significantly increasing the model complexity. Figure 4.2 presents a forecast produced by a piecewise model with $N_a = 8$, $\delta_a = 4$, $N_b = 3$, and $\delta_b = 2$. The top subplot shows the 2 hour forecast and the measured air temperature within the apartment. The remaining subplots show the θ_a, θ_b, and θ_c parameter values employed by the piecewise model at each time step of the forecast.

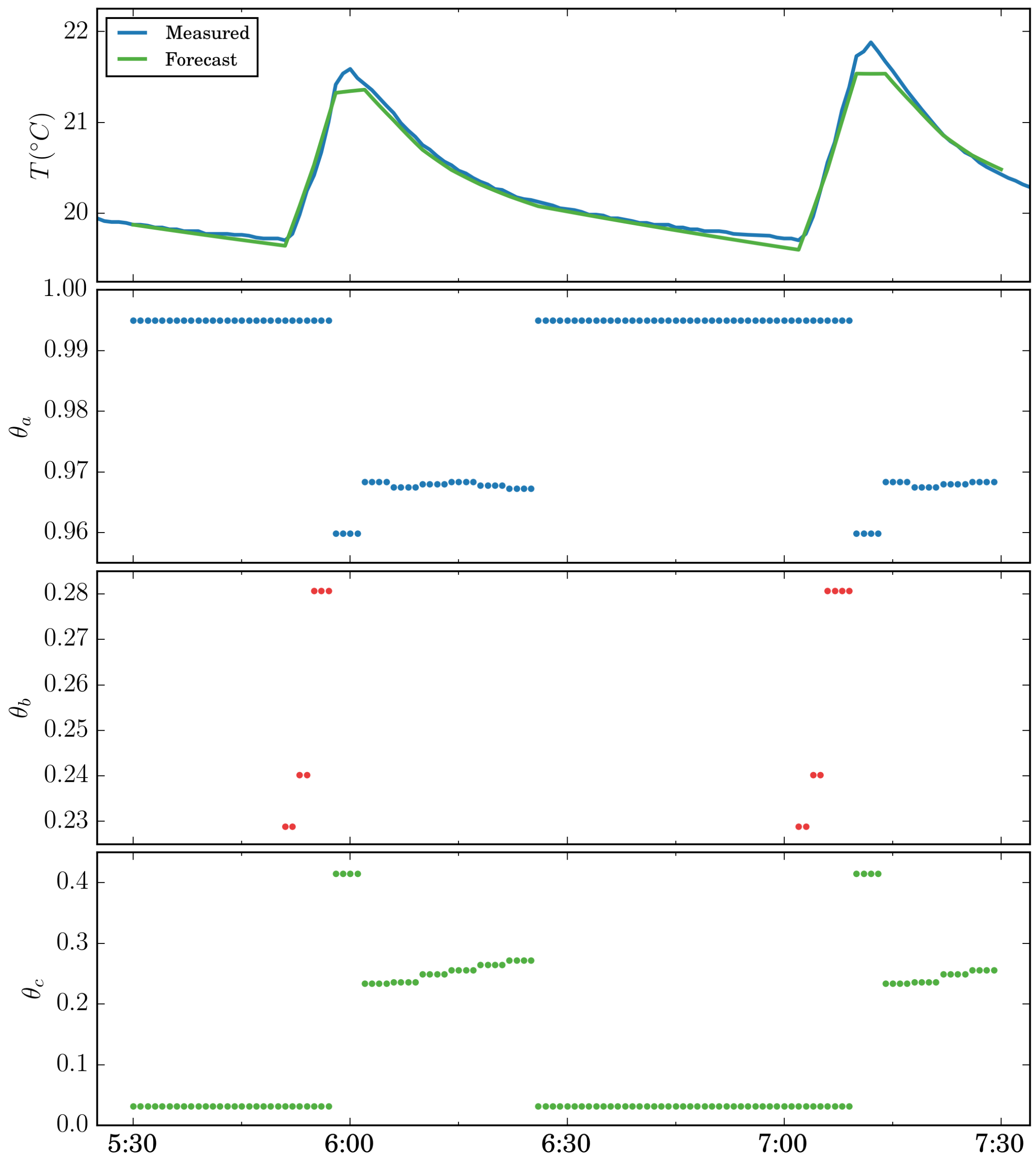

Figure 4.2: **Model Parameters.** Piecewise thermal model parameters with $N_a = 8$ and $N_b = 3$ used to produce a 2 hour forecast.

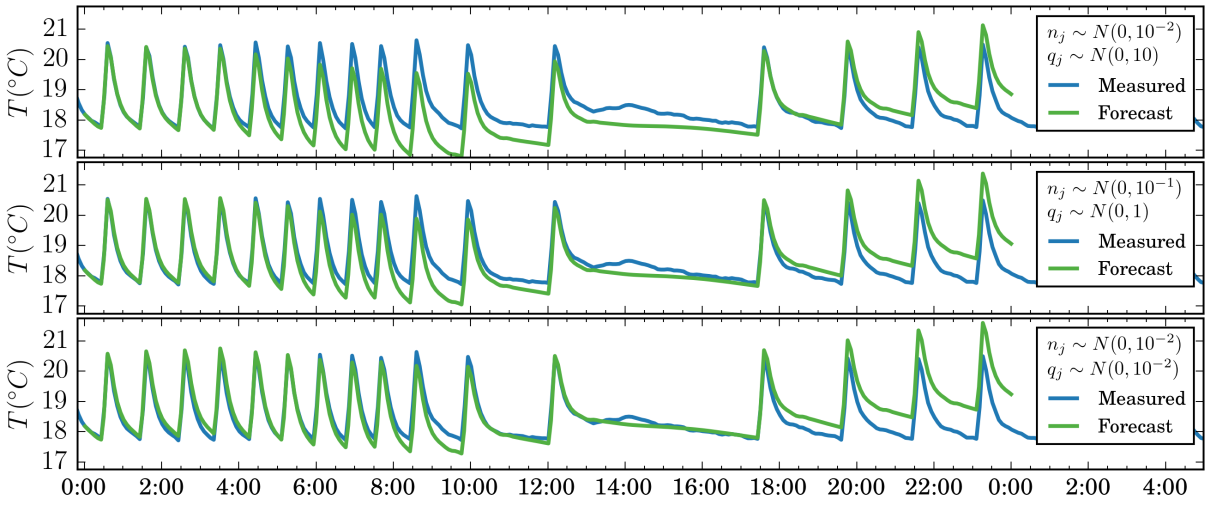

Figure 4.3: **Forecast.** Examples of piecewise thermal models used to produce a 24 hour forecast with various covariance values for the n_j and q_j noise terms. In each case, $N_a = 15$, $\delta_a = 5$, $N_b = 2$, $\delta_b = 4$, $n_{i,1} \sim N(0, 10^{-4})$, $n_{i,2} \sim N(0, 10^{-4})$, $q_i \sim N(0, 10^{-1})$.

Figure 4.3 illustrates the ability of the model to produce accurate multi-hour forecasts and the influence of the noise covariances on the parameter estimates. In each subplot, different covariance values are used to represent the n_j and q_j noise terms and the forecasts are produced using the model parameters as estimated at the start of the time horizon. In the top subplot, the model is very accurate for the first several hours before the forecasted temperature begins to drift downward. The root mean squared error (RMSE) over the first 3 hours is 0.039 °C and over the first 12 hours is 0.573 °C. In the bottom subplot, the error is less varied with an RMSE of 0.162 °C over the first 3 hours and 0.240 °C over the first 12 hours.

4.5 Conclusions

This chapter addressed the need for control-oriented thermal models of buildings. We presented a piecewise linear thermal model of a building that is suitable for model predictive control applications. To estimate the model parameters, we developed a Kalman filter based system identification method. Finally, we presented experimental results using real temperature data collected from an apartment with a forced-air heating and ventilation system. These results demonstrate the potential of the model and parameter estimation method to produce accurate forecasts of the air temperature within the apartment.

Part II

Control

Chapter 5

Alternative Control Trajectory Representation

This chapter presents the alternative control trajectory representation – a novel approach for representing the control of a non-convex discrete system as a convex program. The resulting convex program provides a solution that can be interpreted stochastically for implementation. This approach enables the approximate optimal control of non-convex agents using distributed convex optimization techniques.

5.1 Motivation & Background

A fundamental requirement of the electric power system is to maintain a continuous and instantaneous balance between generation and load. The variability of renewable energy resources, particularly wind and solar, poses a challenge for power system operators. Namely, as renewable penetration increases, it will be necessary for operators to procure more ancillary services, such as regulation and load following, to maintain balance between generation and load [60, 107]. In the long-term, grid-scale storage technologies (e.g. flywheels, batteries, etc.) are sure to play a major role in providing these ancillary services [45, 34]. In the near-term, there is a high potential for aggregated loads, in particular electric vehicles (EVs) and thermostatically controlled loads (TCLs), to providing such ancillary services [87, 18, 50, 51, 49, 13].

The advantages of responsive aggregated loads over large storage technologies include: 1) they are distributed throughout the power system thus providing spatially and temporally distributed actuation; 2) they employ simple and fast local actuation well-suited for real-time control; 3) they are robust to outages of individuals in the population; and 4) they, on the aggregate, can produce a quasi-continuous response despite the discrete nature of the individual controls [18, 64, 17].

Energy systems like EVs and TCLs often have binary or discrete states due to hardware limitations and efficiency characteristics. Consequently, non-convex techniques are generally required for optimal control. This poses a challenge for load aggregation applications since distributed optimization methods generally require linearity or convexity in the agents. In this chapter, we develop the alternative control trajectory representation – a novel approach for representing the control of a non-convex discrete system as a convex program. This representation enables the approximate optimization of energy systems using distributed convex algorithms, such as the alternating direction method of multipliers (ADMM), and provides a solution that can be interpreted stochastically for implementation.

Contributions

The alternative control trajectory representation enables the control of agents with non-convex constraints using convex optimization techniques. The solution to the convex program can be interpreted stochastically for implementation. Furthermore, the alternative control trajectory representation enables the approximate optimal control of a population of non-convex agents using distributed convex optimization techniques. Experimental results demonstrating the application of the alternative control trajectory approach to the distributed optimization of thermostatically controlled loads for electricity generation following ancillary services are presented in the next chapter.

Chapter Outline

This chapter is organized as follows. Section 5.2 briefly describes the optimization of non-convex systems and section 5.3 presents the alternative control trajectory representation. Section 5.4 overviews the incorporation of the ACT representation into a convex program and the stochastic interpretation of the solution. Section 5.5 describes the incorporation of the ACT representation into a distributed optimization algorithm, the statistical characteristics of the solution, and an iterative method for reducing variance by inducing sparsity. Finally, Section 5.6 provides an illustrative example of the proposed modeling and optimization approach.

5.2 Optimization of Non-Convex Systems

In this section, we consider the optimization of an arbitrary discrete-time system represented by the state-space model

$$\begin{aligned} x^k &= G(x^{k-1}, u^k) \\ y^k &= H(x^k, u^k) \end{aligned} \tag{5.1}$$

where G and H are known functions, x^k is the state of the system, u^k is the exogenous input, y^k is the output, and k denotes the integer-valued time step. For simplicity, this chapter will only consider the univariate case (i.e x^k, u^k, and y^k are univariate, $G : \mathbf{R}^2 \rightarrow \mathbf{R}$, and

$H : \mathbf{R}^2 \to \mathbf{R}$). Functions G and H may be any closed deterministic function (i.e. non-convex, piece-wise, semi-continuous, etc.) and x^k, u^k, and y^k may be continuous or discrete.

We would like to solve an optimization problem of the form

$$\begin{array}{ll} \underset{u}{\text{minimize}} & F_0(y) \\ \text{subject to} & F_i(x,u) \le b_i, \; i = 1, \ldots, M \\ & x^k = G(x^{k-1}, u^k), \; k = 1, \ldots, N \\ & y^k = H(x^k, u^k), \; k = 1, \ldots, N \\ & x^0 = x_0 \end{array} \tag{5.2}$$

where $F_0 : \mathbf{R}^N \to (-\infty, \infty]$ is a closed convex objective function, N is the number of time steps, and x_0 is the initial state. Functions $F_i : \mathbf{R}^2 \to \mathbf{R}$, $i = 1, \ldots, M$ represent the constraints of the system. Like G and H, F_i may be any closed deterministic function.

There are a number of non-convex optimization techniques, such as dynamic programming and genetic algorithms, suitable for solving (5.2) to identify a control trajectory u^* that optimizes the system. Convex optimization techniques, however, are unsuitable given the non-convex constraints and the discrete states, inputs, and outputs.

5.3 Alternative Control Trajectory Representation

In this section, we introduce the alternative control trajectory (ACT) representation, a novel approach for representing the control of non-convex systems in a manner suitable for convex programming. Put simply, we simulate the system under multiple alternative control inputs in order to generate a discrete set of output trajectories. These alternative control trajectories can be incorporated into a convex program as a linear constraint, thereby enforcing feasibility. By solving the convex program, we produce a solution that can be interpreted stochastically for implementation.

It should be noted that the alternative control trajectories do not represent the full decision space of the original optimization program (5.2) and that the stochastic solution has no optimality guarantee. Rather, the contribution of the ACT representation is to enable the optimization of a large population of non-convex agents using distributed convex optimization. Accordingly, by employing the ACT representation, we are accepting suboptimality in the individual objectives in order to achieve optimality in the global objective.

To produce the alternative control trajectory representation of a system, we first define N_a input trajectories for N_t time steps

$$\begin{array}{c} u_j = (u_j^1, u_j^2, \ldots, u_j^{N_t}) \\ \forall \; j = 1, \ldots, N_a \end{array} \tag{5.3}$$

with variable $u_j \in \mathbf{R}^{N_t}$ and $u_j^k \in \mathbf{S}_u$ for $k = 1, ..., N_t$, where $\mathbf{S}_u$ is the discrete or continuous constraint set of feasible inputs. Each of the input trajectories u_j must be distinct and should be selected to produce a distinguishable change in the system's output (i.e. performance extremes, efficiency optimum, etc.). Regardless of whether the input is discrete or continuous, the set of alternative input trajectories express only a small but key portion of the true decision space.

Next, for each input trajectory u_j, we simulate the system model (5.1) according to the update function G with $x^0 = x_i$ while imposing any additional constraints (represented by H_i in (5.2)). Given the simulation results, we generate N_a alternative state and output trajectories as defined by the x_j and y_j, respectively.

$$\begin{aligned} x_j &= (x_j^1, x_j^2, \dots, x_j^{N_t}) \\ y_j &= (y_j^1, y_j^2, \dots, y_j^{N_t}) \\ \forall\ j &= 1, \dots, N_a \end{aligned} \tag{5.4}$$

The input, state, and output trajectories can be expressed compactly as

$$\begin{aligned} \mathbf{U} &= (u_1, u_2, \dots, u_{N_a}) \\ \mathbf{X} &= (x_1, x_2, \dots, x_{N_a}) \\ \mathbf{Y} &= (y_1, y_2, \dots, y_{N_a}) \end{aligned} \tag{5.5}$$

with variables $\mathbf{U}$, $\mathbf{X}$, and $\mathbf{Y}$ representing the set of all u_j, x_j, and y_j sets for $j = 1, \dots, N_a$. Naturally, we can also view $\mathbf{U}$, $\mathbf{X}$, and $\mathbf{Y}$ as matrices $\in \mathbf{R}^{N_a \times N_t}$ such that the rows represent the alternative trajectories and the columns represent the time step k.

In the case that functions G and/or H are not injective/one-to-one and the distinctness of u_j does not guarantee the distinctness of x_j or y_j, it is necessary to reduce the number of trajectories in $\mathbf{U}$, $\mathbf{X}$, and $\mathbf{Y}$. We define the number of *distinct* alternative control trajectories as N_d such that $N_d \in \{1, \dots, N_a\}$.

5.4 Convex Optimization

In this section, we detail how the ACT representation described above can be introduced into a convex program. To begin, we introduce a variable $w \in \{0, 1\}^{N_d}$ such that

$$\begin{aligned} w_j &= \begin{cases} 1 & \text{if trajectory } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \\ \forall\ j &= 1, \dots, N_d \end{aligned} \tag{5.6}$$

Thus, if $j = 1$ is the selected trajectory (i.e. $w_1 = 1$)

$$\begin{aligned} \mathbf{U}^T w &= u_1 \\ \mathbf{X}^T w &= x_1 \\ \mathbf{Y}^T w &= y_1 \end{aligned}$$

The integer/binary program below demonstrates how $\mathbf{Y}$ and w can be introduced to solve for the optimal trajectory

$$\begin{array}{ll} \underset{w}{\text{minimize}} & F(\mathbf{Y}^T w) \\ \text{subject to} & \sum w_j = 1 \\ & w \in \{0,1\}^{N_d} \end{array} \tag{5.7}$$

where $F : \mathbf{R}^{N_t} \to (-\infty, \infty]$ is a closed convex objective function. The above program is an example of the generalized assignment problem (GAP). If feasible, (5.7) guarantees that only one component of minimizer w^* is non-zero. Therefore, $y^* = \mathbf{Y}^T w^*$ is the optimal output trajectory within the discrete set defined by $\mathbf{Y}$. However, the binary constraint makes the program non-convex and NP-complete. By relaxing the binary constraint such that $\hat{w} \in \mathbf{R}^{N_d}$, we can express the convex program as

$$\begin{array}{ll} \underset{\hat{w}}{\text{minimize}} & F(\mathbf{Y}^T \hat{w}) \\ \text{subject to} & \sum \hat{w}_j = 1 \\ & \hat{w} \geq 0 \\ & \hat{w} \in \mathbf{R}^{N_d} \end{array} \tag{5.8}$$

The program is now convex and the decision variable continuous. By minimizing the objective function with respect to $\hat{w}$, we allow the convex program to form weighted averages of the alternative output trajectories. Therefore, $\hat{y}^* = \mathbf{Y}^T \hat{w}^*$ is the optimal weighted average of the output trajectories within the discrete set defined by $\mathbf{Y}$. However, for many systems, the solution defined by $\hat{w}^*$ is not realizable (e.g. $\hat{u}^* = \mathbf{U}^T \hat{w}^*$ is not within the feasible space, $\hat{y}^{*,k} \neq H(\hat{x}^{*,k}, \hat{u}^{*,k})$, etc.). To produce a realizable solution, we can interpret $\hat{w}^*$ stochastically, as described in the next section.

Stochastic Solution

Due to the linear constraints, the optimal solution $\hat{w}_j^*$ is $\in [0,1]$ for $j = 1, \ldots, N_d$ and in practice, $\hat{w}_j^*$ can be interpreted as the probability of selecting control trajectory j. Thus, we can implement a single trajectory $\tilde{y} \in \mathbf{Y}$ based on the discrete probability distribution $\hat{w}^*$. Expressed alternatively, we can generate a discrete random variable $W \in \{1, \ldots, N_d\}$ such that $\hat{w}_j^* = \Pr(W = j)$ for $j = 1, \ldots, N_d$. The value of W represents the index of the stochastically selected control trajectory. Thus, we can define a variable $\tilde{w} \in \{0,1\}^{N_d}$, representing the stochastic solution of (5.8), as

$$\begin{gathered} \tilde{w}_j = \begin{cases} 1 & \text{if } W = j \\ 0 & \text{otherwise} \end{cases} \\ \forall\ j = 1, \ldots, N_d \end{gathered} \tag{5.9}$$

The selected output trajectory is therefore given by $\tilde{y} = \mathbf{Y}^T\tilde{w}$. By treating $\hat{w}^*$ as a discrete probability distribution, $\hat{y}^*$ becomes the probability-weighted average of possible output trajectories (as defined by $\mathbf{Y}$). Therefore, $\hat{y}^*$ is the expected value of $\tilde{y}$.

$$\mathrm{E}[\tilde{y}] = \hat{y}^* \tag{5.10}$$

To summarize, the optimal solution to (5.7) is physically realizable (i.e. only one component of w^* is non-zero) but not solvable using convex optimization. By contrast, (5.8) is convex but the optimal solution is not realizable (i.e. all components of $\hat{w}^*$ may be non-zero). Using (5.9), we can transform $\hat{w}^*$ into $\tilde{w}$, which is realizable (i.e. only one component of $\tilde{w}$ is non-zero). It should be noted that w^* and $\hat{w}^*$ are guaranteed to be optimal solutions to (5.7) and (5.8), respectively. However, $\tilde{w}$ may be an optimal or sub-optimal solution to both (5.7) and (5.8).

Throughout this chapter, we refer to the optimal output trajectory ($y = \mathbf{Y}^T w$) produced by (5.7) as the *discrete* solution y^* ($w^* \in \{0,1\}^{N_d}$), by (5.8) as the *continuous* solution $\hat{y}^*$ ($\hat{w}^* \in \mathbf{R}^{N_d}$), and by (5.8) and (5.9) as the *stochastic* solution $\tilde{y}$ ($\tilde{w} \in \{0,1\}^{N_d}$). It should be noted that y^* and $\hat{y}^*$ are deterministic whereas $\tilde{y}$ is, of course, stochastic.

5.5 Distributed Optimization

In this chapter, we have detailed the ACT representation for expressing the control of a non-convex discrete system as a convex program and have discussed how the solution can be interpreted stochastically for implementation. In this section, we briefly discuss the application of this approach to distributed convex optimization.

Consider the generic *sharing* problem of the form

$$\underset{y}{\text{minimize}} \quad \textstyle\sum f_i(y_i) + g(\sum y_i) \tag{5.11}$$

with variables $y_i \in \mathbf{S}_i^{N_y}$, the decision variable of agent i for $i = 1, \ldots, N_p$, where $\mathbf{S}_i$ represents the convex constraint set of agent i, N_p the number of agents in the population, N_y is the length of y_i, f_i is the convex objective function for agent i, and g is the shared convex objective function of the population. The function g takes as input the sum of the individual agent's decision variables, y_i. The sharing problem allows each agent in the population to minimize its individual/private cost $f_i(y_i)$ as well as the shared objective $g(\sum y_i)$. The problem is known to be solvable using iterative methods of distributed convex optimization, such as the alternating direction of multipliers algorithm (ADMM) [9].

The ACT representation can be incorporated into the objective functions of (5.11) as given by

$$\begin{array}{ll} \underset{\hat{w}}{\text{minimize}} & \sum f_i(\mathbf{Y}_i^T \hat{w}_i) + g(\sum \mathbf{Y}_i^T \hat{w}_i) \\ \text{subject to} & \sum \hat{w}_{i,j} = 1 \\ & \hat{w}_i \geq 0 \\ & \hat{w}_i \in \mathbf{R}^{N_{d,i}} \\ & \forall\ i = 1, \ldots, N_p \end{array} \tag{5.12}$$

with variables $\hat{w}_i \in \mathbf{R}^{N_{d,i}}$, the decision variable of agent i, $\mathbf{Y}_i \in \mathbf{R}^{N_{d,i} \times N_y}$, the set of alternative output trajectories for agent i, and $N_{d,i}$, the number of distinct trajectories in $\mathbf{Y}_i$ for $i = 1, \ldots, N_p$. Because the objective function and constraints of each agent are separable, the problem can be solved in a distributed manner. The optimal output $\hat{y}_i^* = \mathbf{Y}_i^T \hat{w}_i^*$ for $i = 1, \ldots, N_p$ is the *continuous* solution of each agent in the population. Thus, $\tilde{y}_i = \mathbf{Y}_i^T \tilde{w}_i$ is the final *stochastic* solution and can be implemented by each agent.

Aggregated Stochastic Solution

When trying to optimize the behavior of a population, we are interested in understanding the relationship between the aggregate of the continuous and stochastic solutions, as given by

$$\begin{aligned} \hat{S} &= \sum \hat{y}_i^* \\ \tilde{S} &= \sum \tilde{y}_i \\ e &= \tilde{S} - \hat{S} \end{aligned} \tag{5.13}$$

with variables $\hat{S} \in \mathbf{R}^{N_y}$, the sum of the continuous solutions, $\tilde{S} \in \mathbf{R}^{N_y}$, the sum of the stochastic solutions, and $e \in \mathbf{R}^{N_y}$, the error between $\hat{S}$ and $\tilde{S}$ (ideally, $e \in \{0\}^{N_y}$).

Because $\hat{y}_i^*$ is the expected value of $\tilde{y}_i$, $\hat{S}$ is the expect value of $\tilde{S}$

$$\begin{aligned} \mathrm{E}[\tilde{S}] &= \sum \mathrm{E}[\tilde{y}_i] \\ &= \sum \hat{y}_i^* \\ &= \hat{S} \end{aligned} \tag{5.14}$$

The error e is therefore related to the variance of $\tilde{S}$, given by

$$
\begin{aligned}
\mathrm{Var}(\tilde{S}) &= \mathrm{E}[(\tilde{S} - \mathrm{E}[\tilde{S}])^2] \\
&= \mathrm{E}[(\tilde{y}_1 - \hat{y}_1^* + \ldots + \tilde{y}_{N_p} - \hat{y}_{N_p}^*)^2] \\
&= \mathrm{E}[(\tilde{y}_1 + \ldots + \tilde{y}_{N_p})^2] \\
&\quad - (\hat{y}_1^* + \ldots + \hat{y}_{N_p}^*)^2 \\
&= \sum_{i=1}^{N_p} (\mathrm{E}[\tilde{y}_i^2] - (\hat{y}_i^*)^2) \\
&\quad + \sum_{i \neq j} (\mathrm{E}[\tilde{y}_i]\mathrm{E}[\tilde{y}_j] - (\hat{y}_i^* \hat{y}_j^*)) \\
&= \sum_{i=1}^{N_p} \mathrm{Var}(\tilde{y}_i) + \sum_{i \neq j} \mathrm{Cov}(\tilde{y}_i, \tilde{y}_j)
\end{aligned}
\tag{5.15}
$$

Because the random variables are uncorrelated ($\mathrm{Cov}(\tilde{y}_i, \tilde{y}_j) = 0, \forall (i \neq j)$), the variance of $\tilde{S}$ reduces to

$$
\begin{aligned}
\mathrm{Var}(\tilde{S}) &= \sum_{i=1}^{N_p} \mathrm{Var}(\tilde{y}_i) \\
&= \sum_{i=1}^{N_p} (\mathrm{E}[\tilde{y}_i^2] - (\hat{y}_i^*)^2)
\end{aligned}
\tag{5.16}
$$

Since $\hat{w}^*$ is a discrete probability distribution

$$
\begin{aligned}
\mathrm{Var}(\tilde{S}) &= \sum_{i=1}^{N_p} \sum_{j=1}^{N_{d,i}} \hat{w}_{i,j}^* (y_{i,j} - \hat{y}_i^*)^2 \\
&= \sum_{i=1}^{N_p} \left(\sum_{j=1}^{N_{d,i}} (\hat{w}_{i,j}^* y_{i,j}^2) - (\hat{y}_i^*)^2 \right) \\
&= \sum_{i=1}^{N_p} \sum_{j=1}^{N_{d,i}} (\hat{w}_{i,j}^* y_{i,j}^2) - \sum_{i=1}^{N_p} (\hat{y}_i^*)^2
\end{aligned}
\tag{5.17}
$$

where variable $y_{i,j}$ is the j-th alternative output trajectory for agent i.

In the remainder of this section, we discuss two particular characteristics that impact the error $e = \tilde{S} - \hat{S}$ and the variance of $\tilde{S}$: the homogeneity/heterogeneity of the agents in the population and the sparsity of the discrete probability distribution $\hat{w}_i^*$ (i.e. the number of non-zero terms) for $i = 1, \ldots, N_p$.

For a population of highly homogeneous agents with identical output trajectories and objective functions, solving (5.12) will cause each agent to converge to the same solution $\hat{w}_i^*$. Effectively, the output of each agent is defined by the same random variable $\tilde{y}_i$ with the same probability distribution $\hat{w}_i^*$ and expected value $\hat{y}_i^*$. This is a special case where

$$\begin{aligned} \mathrm{E}(\tilde{S}) &= N_p \hat{y}_i^* \\ \mathrm{Var}(\tilde{S}) &= N_p \sum_{j=1}^{N_{d,i}} \hat{w}_{i,j}^* (y_{i,j} - \hat{y}_i^*)^2 \end{aligned} \tag{5.18}$$

and the probability mass of $\tilde{S}$ becomes more and more concentrated about $\mathrm{E}(\tilde{S}) = \hat{S}$ as the number of agents N_p increases. If N_p is very large, the distribution has a narrow peak at $\hat{S}$ regardless of the sparsity of $\hat{w}_i^*$. Therefore, by the law of large numbers, $\tilde{S} \to \hat{S}$ and $e \to \{0\}^{N_y}$ as $N_d \to \infty$. As the heterogeneity of the population increases, this characteristic weakens as the probability mass of $\tilde{S}$ flattens. For a heterogeneous population, the output of each agent is no longer defined by the same random variable and (5.12) is less likely to converge to similar probability distributions.

The sparsity of $\hat{w}_i^*$ also impacts the variance of $\tilde{y}_i$. In the most sparse case, only one term in $\hat{w}_i^*$ is non-zero for every agent in the population. Therefore, $\tilde{y}_i$ is a constant random variable ($\mathrm{Var}(\tilde{y}_i) = \{0\}^{N_y}$) equal to its expected value ($\tilde{y}_i = \hat{y}_i^*$). Accordingly, $\mathrm{Var}(\tilde{S}) = \{0\}^{N_y}$ and $\tilde{S} \to \hat{S}$.

In the least sparse case, every agent is equally likely to implement any one of its control trajectories (i.e. $\hat{w}_{i,j}^* = 1/N_{d,i} \ \forall \ j = 1, \ldots, N_{d,i}$). Thus

$$\begin{aligned} \mathrm{E}(\tilde{S}) &= \hat{S} \\ \mathrm{Var}(\tilde{S}) &= \sum_{i=1}^{N_p} \sum_{j=1}^{N_{d,i}} \frac{(y_{i,j} - \hat{y}_i^*)^2}{N_{d,i}} \end{aligned} \tag{5.19}$$

and the aggregate behavior of the population becomes highly stochastic, especially as heterogeneity in $\mathbf{Y}_i$ increases.

Inducing Sparsity

The stochasticity of $\tilde{S}$ diminishes our ability to optimally control the behavior of the distributed population. Particularly, in order to optimize a highly heterogeneous population, it would be desirable to force the variance of $\tilde{S}$ towards zero. In this case, we would no longer rely on the law of large numbers to drive $\tilde{S}$ towards the expected value $\hat{S}$.

To decrease the variance, we focus on inducing sparsity in the continuous solution $\hat{w}^*$ of a single system. In this section, we begin by discussing the challenges of inducing sparsity and conclude with an iterative optimization technique. This iterative technique adds a linear cost function to (5.8) which drives the terms in $\hat{w}$ towards 0 and 1.

It is important to recognize that attempting to induce sparsity in the solution to (5.8) is prone to introducing non-convexity to the program. As mentioned previously, integer programming with a branch and bound algorithm is non-convex. The ℓ_1-norm, when added as a linear regularization penalty to an objective function, is known to incentivize sparsity in the solution [9, 19]. However, due to the linear constraints in (5.8), ℓ_1 regularization is ineffective (i.e. $\|\hat{w}\|_1 = 1$). Direct attempts to drive the terms in $\hat{w}$ towards 0 and 1 (i.e. min $F(\mathbf{Y}^T\hat{w}) + \sum \hat{w}_j(1-\hat{w}_j)$) or to minimize the variance of $\tilde{y}$ (i.e. min $F(\mathbf{Y}^T\hat{w}) + \sum \hat{w}_j(y_j - \mathbf{Y}^T\hat{w})^2$) are concave.

In the remainder of this section, we present an iterative technique for inducing sparsity in $\hat{w}^*$. Put simply, at each iteration n, we solve (5.8) with a linear weight $\beta^n \in \mathbf{R}^{N_d}$ applied to $\hat{w}^n$

$$\begin{array}{ll} \underset{\hat{w}^n}{\text{minimize}} & F(\mathbf{Y}^T\hat{w}^n) + \alpha^n(\hat{w}^n)^T\beta^n \\ \text{subject to} & \sum \hat{w}_j^n = 1 \\ & \hat{w}^n \geq 0 \\ & \hat{w}^n \in \mathbf{R}^{N_d} \end{array} \tag{5.20}$$

where α^n is a scaling parameter for the sparsity-inducing cost.

The linear weight is initialized at 0 ($\beta^0 \in \{0\}^{N_d}$) and after each iteration n, updated according to the previous solution $(\hat{w}^n)^*$

$$\begin{array}{c} \beta_j^{n+1} = ||y_j - \mathbf{Y}^T(\hat{w}_j^n)^*||_2^2 \\ \forall\ j = 1, \ldots, N_d \end{array} \tag{5.21}$$

Essentially, we are estimating the variance of $\tilde{y}$ (which is concave with respect to $\hat{w}$) as a linear cost. For each successive iteration, the terms in $\hat{w}^n$ are encouraged, though not required, to approach 0 or 1. To enable tie-breaking, we can add a small random perturbation to the weight update

$$\begin{array}{c} \beta_j^{n+1} = ||y_j - \mathbf{Y}^T(\hat{w}_j^n)^*||_2^2 + v \\ \forall\ j = 1, \ldots, N_d \end{array} \tag{5.22}$$

where $v \in \mathbf{R}$ is a Gaussian random variable with a small covariance (e.g. $v \sim N(0, 0.01)$). This will allow ties between different output trajectories to be broken randomly.

Lastly, we define a simple procedure for updating the scaling parameter α^n:

$$\begin{aligned} \alpha^0 &= 0 \\ \alpha^{n+1} &= \begin{cases} \alpha^n + 0.005 & \text{if } \mathrm{Max}(\mathrm{Var}(\tilde{y})^n) > \sigma_{max}^2 \\ 0.9\alpha^n & \text{if } \mathrm{Max}(\mathrm{Var}(\tilde{y})^n) < \sigma_{max}^2/2 \\ \alpha^n & \text{otherwise} \end{cases} \end{aligned} \tag{5.23}$$

where $\mathrm{Var}(\tilde{y})^n \in \mathbf{R}^{N_y}$ is the variance of $\tilde{y}$ at iteration n based on $(\hat{w}^n)^*$ and σ_{max}^2 defines the maximum desired variance.

This iterative technique for minimizing variance and inducing sparsity in $\hat{w}^*$ can be applied to the optimization of individual agents. While the objective of (5.20) is not constant, the change in β^n from one iteration to the next is relatively small. On the whole, the updating weight β^n introduces concavity into the problem. Specifically, the magnitude of each weight increases as the terms in $(\hat{w}^n)^*$ approach 0 or 1. With each successive iteration, $(\hat{w}^n)^*$ is forced further away from $(\hat{w}^0)^*$, the optimal solution to (5.8).

5.6 Illustrative Example

To illustrate the application of the ACT representation for the convex optimization of a non-convex discrete energy system, this section considers the control of a thermostatically controlled load (TCL). Specifically, we optimize the electricity demand of a simulated residential refrigerator using the techniques described in this chapter.

By modifying (5.24) to allow for setpoint changes, the TCL is modeled using the hybrid state discrete time model [69, 38, 17, 13, 14]

$$\begin{aligned} T^k &= \theta_1 T^{k-1} + (1-\theta_1)(T_\infty^k + \theta_2 m^k) + \theta_3 \\ m^k &= \begin{cases} 1 & \text{if } T^k < T_{set} - \frac{\delta}{2} + u^k \\ 0 & \text{if } T^k > T_{set} + \frac{\delta}{2} + u^k \\ m^k & \text{otherwise} \end{cases} \end{aligned} \tag{5.24}$$

where state variables $T^k \in \mathbf{R}$ and $m^k \in \{0, 1\}$ denote the temperature of the conditioned mass and the discrete state (on or off) of the mechanical system, respectively. Additionally, $k = 1, 2, \ldots, N_t$ denotes the integer-valued time step, $T_\infty^k \in \mathbf{R}$, the ambient temperature (°C), $T_{set} \in \mathbf{R}$, the temperature setpoint (°C), and $\delta \in \mathbf{R}$, the temperature deadband width (°C). The control input $u^k \in \mathbf{S}_u$ is a setpoint change at each time step where $\mathbf{S}_u$ defines the discrete set of feasible values.

The electricity demand of the TCL at each time step is defined by

$$y^k = \frac{|P|}{COP} m^k \tag{5.25}$$

where $y^k \in \mathbf{R}$ is the electric power demand (kW) and COP, the coefficient of performance. We now have the state and output equations necessary to model the system ((5.24) serves as G and (5.25) as H).

Figures 5.1, 5.2, and 5.3 present examples of $N_a = 3$ alternative trajectories for the TCL. In the examples, each alternative input u_j for $j = 1, 2, 3$ is $\in \{0, -1, 1\}^{20}$ (i.e $N_t = 20$). While the input trajectories are not plotted, they can be inferred from the changes in the setpoint and temperature bounds. For trajectory $j = 1$, $u_1^k = 0$ for $k = 1, \ldots, 20$. For trajectory $j = 2$, $u_2^k = 0$ for $k = 1, \ldots, 10$ and $u_2^k = -1$ for $k = 11, \ldots, 20$. For trajectory $j = 3$, $u_3^k = 0$ for $k = 1, \ldots, 10$ and $u_3^k = 1$ for $k = 11, \ldots, 20$.

The TCL has been simulated using (5.24) and (5.25) with a default setpoint T_{set} of 2.5°C, a deadband width δ of 2°C, an initial temperature T^0 of 3.3°C, and an initial mechanical state m^0 of 0. Figures 5.1, 5.2, and 5.3 present the T_j and y_j trajectories corresponding to each input u_j for $j = 1, 2, 3$. The mechanical state trajectories m_j can be inferred from the T_j and y_j trajectories. As illustrated by the figures, each distinct input u_j produces a distinct T_j, m_j, and y_j. Therefore, in this example, $N_d = N_a = 3$.

Next, we define some optimal power demand trajectory $p \in \mathbf{R}^{20}$ which we would like the TCL to match as closely as possible. As illustrated in Figure 5.4, we define $p^k = 0.3$ for

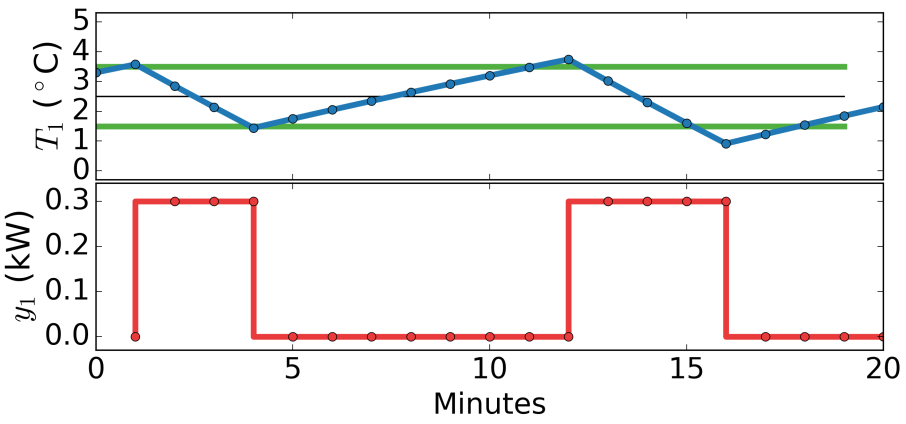

Figure 5.1: T_1 and y_1 trajectories given u_1

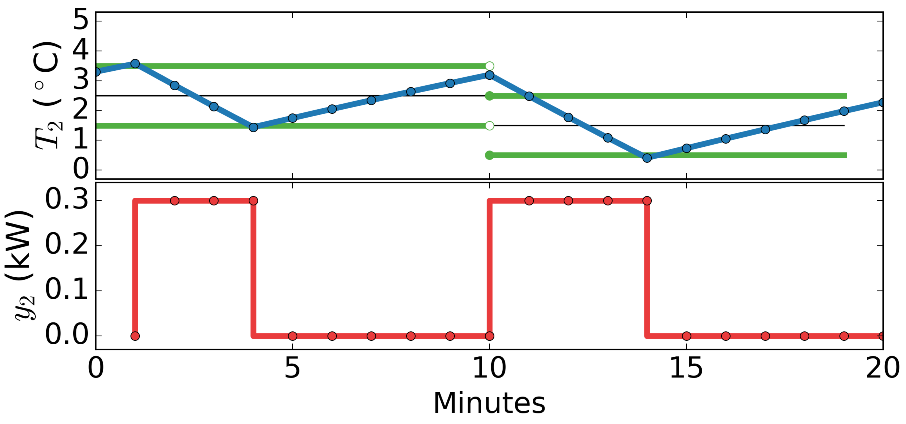

Figure 5.2: T_2 and y_2 trajectories given u_2

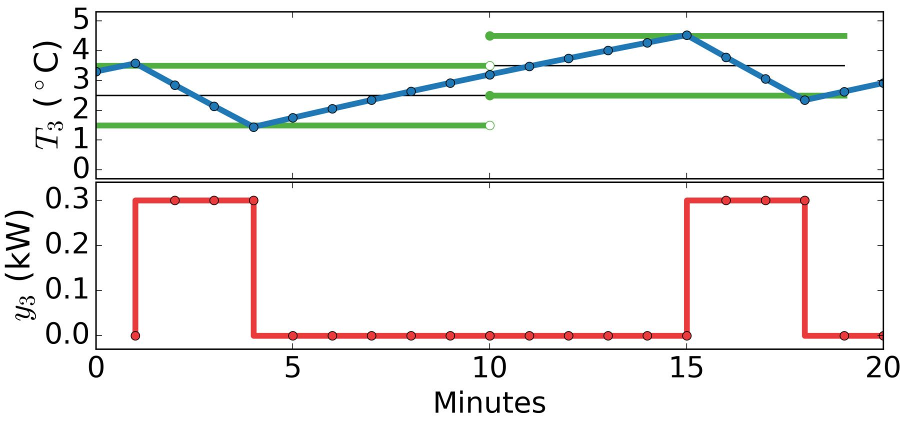

Figure 5.3: T_3 and y_3 trajectories given u_3

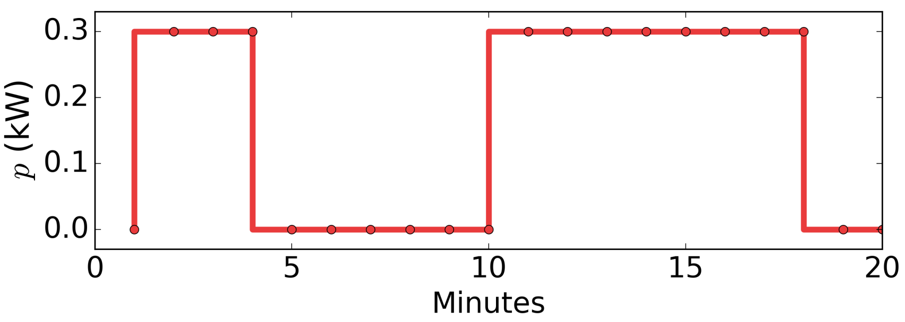

Figure 5.4: Target power demand p

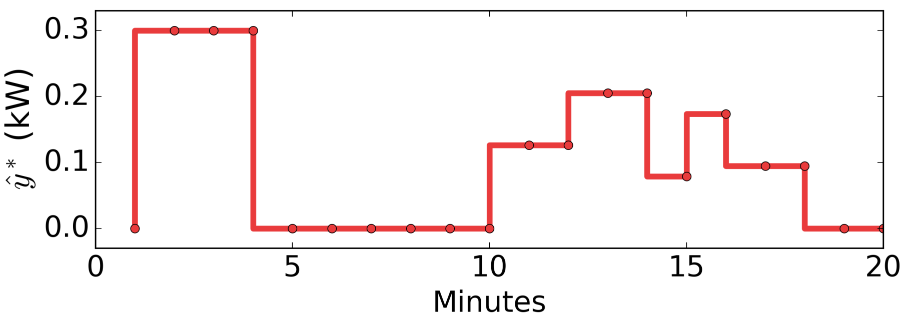

Figure 5.5: Continuous solution $\hat{y}^*$

$k = 2, \ldots, 4$ and for $k = 11, \ldots, 18$ and $p^k = 0$ otherwise. The convex optimization program is defined with a least squares objective function

$$\begin{array}{ll} \underset{\hat{w}}{\text{minimize}} & \|\mathbf{Y}^T \hat{w} - p\|_2^2 \\ \text{subject to} & \sum \hat{w}_j = 1 \\ & \hat{w} \geq 0 \\ & \hat{w} \in \mathbf{R}^{N_d} \end{array} \tag{5.26}$$

By solving (5.26) with $\mathbf{Y}$ and p as described above, we find that $\hat{w}^* = (0.263, 0.421, 0.316)$. The continuous solution $\hat{y}^*$, the optimal linear combination of the alternative output trajectories, is illustrated in Figure 5.5.

It should be noted that the squared error between p and y_1, y_2, and y_3 is 0.134, 0.134, and 0.15, respectively. Thus, the utilities of y_1 and y_2 are equal. However, if we apply (5.9), there are 3 possible outcomes for the discrete solution $\tilde{w}$,

$$\begin{aligned} \Pr(\tilde{w} = (1,0,0)) &= 26.3\% \\ \Pr(\tilde{w} = (0,1,0)) &= 42.1\% \\ \Pr(\tilde{w} = (0,0,1)) &= 31.6\% \end{aligned} \tag{5.27}$$

By applying the sparsity inducing penalty described in (5.20) and (5.22), we find that the $(\hat{w}^*)^n = (0, 1, 0)$ after 3 or 4 iterations. Despite the random perturbation added to the weights, we observe that, for this particular example, the program always converges to the same solution (i.e $(\hat{w}^*)^n \to (0, 1, 0)$ as $n \to \infty$). Thus, for this TCL, we would implement the control trajectory defined by u_2, T_2, m_2, and y_2.

5.7 Conclusions

In this chapter, we developed the alternative control trajectory (ACT) representation – a novel approach for representing the control of a non-convex discrete system as a convex program. The resulting convex program provides a solution that can be interpreted stochastically for implementation. This approach enables the approximate optimal control of non-convex agents using distributed convex optimization techniques. By inducing sparsity in the individual agents, we can increase the predictability (i.e. reduce the variance) of the aggregated output.

Chapter 6

Distributed Optimization of Thermostatically Controlled Loads

In this chapter, we apply the alternative control trajectory (ACT) representation presented in the previous chapter to the problem of aggregating a population of thermostatically controlled loads (TCLs). Specifically, we examine the potential of TCLs, such as refrigerators and electric water heaters, to provide generation following services in real-time energy markets (1 to 5 minutes). To perform distributed optimization across a large populations of TCLs, we apply a variation of the alternating direction method of multipliers (ADMM) algorithm. We numerically demonstrate the algorithm's potential for controlling a TCL population's total power demand within an error tolerance of 10 kW.

6.1 Motivation & Background

The variability of renewable energy resources, particularly wind and solar, poses a challenge for power system operators. Namely, as renewable penetration increases it will be necessary for operators to procure more ancillary services, such as regulation and load following, to maintain balance between generation and load [60, 107]. In the long-term, grid-scale storage technologies (e.g. flywheels, batteries, etc.) are sure to play a major role in providing these ancillary services [45, 34]. In the near-term, responsive thermostatically controlled loads (TCLs) have a high potential for providing such ancillary services [87, 18].

This chapter investigates the challenge of controlling a *heterogeneous* TCL population to perform an ancillary service, specifically 5-minute ahead generation following. For experimental purposes, we define generation following as the complement of load following whereby loads are employed to smooth the power generation from renewable energy sources.

The advantages of responsive TCLs over large storage technologies include: 1) they are well-established technologies; 2) they are distributed throughout the power system thus providing spatially and temporally distributed actuation; 3) they employ simple and fast local actuation well-suited for real-time control; 4) they are robust to outages of individuals in the population; and 5) they, on the aggregate, can produce a quasi-continuous response despite the discrete nature of the individual controls [18, 64, 17]. We refer the reader to [18] for a discussion of the advantages and disadvantages of TCLs compared to grid-scale storage technologies and to [64] for a look into the potential costs and revenues of demand response with TCLs.

Additionally, because TCLs are controlled according to a temperature setpoint, customers are generally indifferent to precisely when energy is consumed as long as the temperatures are maintained within a dead-band range. This natural flexibility makes TCLs a promising candidate for participating in power system services.

Contributions

Novel contributions of this work include:

- The application of an alternating direction method of multipliers (ADMM) sharing algorithm for the distributed convex optimization of TCLs. Each TCL agent optimizes a private objective function, while the central aggregator iteratively updates an incentive variable to drive the population towards a global objective, such as generation following. By distributing the computation using ADMM, each TCL is able to optimize its objective in parallel and the population can efficiently converge to a global solution.

- By applying the alternative control trajectory representation and alternating direction method of multipliers sharing algorithm, this chapter demonstrates the control of a population of systems with integer states using a convex algorithm. This is a fundamental gap that we bridge.

Literature Review

Early TCL Modeling and Cold Load Pickup

Research into the modeling and control of TCLs began with applications to peak shaving and cold load pickup in power systems. Cold load pickup is a phenomenon which occurs in a distribution network due to the restoration of power after an extended outage. Normally, the power demand of thermostatically controlled loads is desynchronized. However, following outages, TCLs will simultaneously demand full power, contributing to the cold load pickup peak. To address this problem, researchers focused on methods for modeling and reducing TCL demand during cold load pickup events as well as peak demand hours.

The earliest examples of such work include the Ihara and Scwheppe paper on space conditioning during cold load pickup [38] and the Chong and Debs paper on individual and aggregation load models [20], both of which used individual TCL models to describe load dynamics. In [69], Mortensen and Haggerty develop a discrete-time TCL model, which was later adapted by Ucak to model heterogeneous TCL populations [93]. In [75], Pahwa and Brice describe the modeling and parameter estimation of residential air conditioning loads as well as a basic aggregation method. Malhame and Chong's study [61] is among the first reports to use stochastic analysis to develop an aggregate model of a TCL population. The resulting coupled Fokker-Planck equations, derived in [61], define the aggregate behavior of a homogeneous population.

While efforts were made in these early works to model the aggregate demand of a TCL population and to propose control schemes for reducing demand during peak hours and cold load pickup events, the most meaningful contributions focused on the modeling and parameter estimation of individual TCLs.

Aggregate TCL Modeling and Centralized Control

Recent research efforts have focused on the modeling of TCL populations using aggregation methods. A key objective of this research is to develop and evaluate methods for characterizing the temperature density evolution of a TCL population. By incorporating centralized control strategies, aggregated TCL populations are able to provide ancillary power system services like load following and regulation rather than just load reduction. In [17], one of the first papers to develop a modeling and control strategy that allows TCLs to perform ancillary services, Callaway uses a linearized Fokker-Planck model to describe the aggregated behavior of a TCL population. Direct load control is achieved by broadcasting a single time-varying setpoint temperature offset signal to every agent. Numerical results demonstrate how small perturbations to the setpoint can enable TCLs to perform wind generation following. Later work builds upon concepts in [17] by considering sliding mode control [8], proportional-integral control [79], linear quadratic regulators [48], and switching rate broadcast actuation [91].

In [64] and [65], Mathieu, Koch, and Callaway propose a proportional controller which, at each time step, broadcasts a switching probability, η, to all the TCLs in the population. If $\eta < 0$, all TCLs that are on must switch off with a probability of η and if $\eta > 0$, TCLs that are off switch on with a probability of η. In [46], Koch et al. employ a linear time-invariant (LTI) representation of a TCL population. As in [8], a "state bin" modeling framework is used and the aggregate probability mass is allowed to move through these bins. A Markov Chain-based approach is used to predict the evolution of the heterogeneous TCL population.

Similar work can be found in [109], [111], and [110] where Zhang et al. use a state bin concept to represent the evolution of the TCLs and introduce clustering to better account for heterogeneity. In [109], a second-order aggregate model for a heterogeneous population of TCLs is developed. To address the high state-space dimensionality of this model, a complexity reduction method and reduced-order model is proposed in [111]. In [110], the second-order aggregate model is used to simulate a population of heating, ventilation, and air-conditioning (HVAC) systems and a novel method for incorporating minimum dwell time is proposed. Specifically, Zhang et al. define a state which represents the number of off TCLs that are "locked" and will not turn on in response to the central control signal. Thus, the individual TCLs are able to locally enforce dwell times and the aggregator is able to adjust the control signal to account for locked TCLs.

A significant body of research has grown out of the above literature in response to open challenges around aggregate model efficacy and efficiency, modeling and control framework limitations, and unaddressed system constraints. In [70], Moura et al. develop a diffusion-advection partial differential equation (PDE) model and a parameter identification scheme for an aggregated population of heterogeneous TCLs, alleviating the need for prior knowledge of TCL parameters. In [26], Ghaffari et al. develop a deterministic hybrid PDE-based model capable of representing a heterogeneous TCL population and apply a uniform dead-band shifting strategy for control. In [97], Vrettos and Anderson research the aggregation of TCLs to simultaneously provide frequency and voltage regulation services, recognizing that solving these problems separately can produce suboptimal solutions. Iacovella et al. introduce the use of tracer TCLs in [37]. These virtual tracer devices represent the state density distribution of a cluster of heterogeneous TCLs. The approach enables the use of reduced-order aggregate models with control achieved via a single broadcasted signal.

In [67] and [66], Mathieu et al. build upon previous work in [64, 65] to employ a state bin modeling framework with a "non-disruptive" approach in which the TCL's temperature is maintained within the existing dead-band. Hao et al. also consider a non-disruptive approach in [31] using a battery model of the TCL population and a priority stack strategy to determine which TCLs to control at a given time step.

Decentralized TCL Control for Frequency Services

Recognizing that system frequency is a universally available indicator of supply-demand imbalance, a number of researchers have developed fully decentralized techniques for performing frequency services with TCLs. In [84], Short et al. show the suitability of TCLs to perform frequency services using system frequency as a control signal and the potential for a population of TCLs to respond to a sudden loss of generation. This demand response capability reduces the dependence of grid operators on rapidly deployable backup generation.

In [107], Xu et al. develop a TCL model in which devices adjust their setpoints linearly according to the system frequency, allowing the population to act as a fast frequency controlled reserve. To address problems of long-term instability, Angeli and Kountouriotis develop a decentralized stochastic controller in [4] that is capable of maintaining desynchronization among the TCLs while regulating overall power consumption. In [89], Tindemans et al. present a stochastic controller whereby each TCL in the population independently targets a reference power profile. The result is a stable and fully decentralized system that requires only the locally available control signals of frequency and time.

A Distributed Approach

There are a number of advantages to the modeling and control approaches described above. Firstly, the aggregated models are based upon linear representations of TCL dynamics. This makes the aggregated models well suited for a variety of established control and optimization techniques. Moreover, these models are good at prediction and control over small time scales (i.e. seconds and milliseconds), making them ideal for producing fast short-term responses (e.g. frequency regulation) [65, 67].

A limitation of these aggregate models is low model fidelity and the inability to incorporate device specific dynamics. Note the literature is rich with techniques for multi-state thermal modeling of heating, ventilation, and air-conditioning (HVAC) systems in buildings including solar gain estimation and multi-zone state estimation [103, 29, 72, 80, 15]. Because aggregate models are not amenable to the incorporation of device specific, nonlinear, or non-parametric models, they are incapable of leveraging the work of these and other researchers. At larger time scales (i.e. minutes and hours), higher model fidelity becomes very important for the accurate forecasting of TCL power demand. By employing basic linear models, particularly when modeling the complex dynamics of HVAC systems in buildings, aggregated TCL modeling approaches are poorly suited for producing accurate long-term responses (e.g. load-shifting) [29, 72, 80]. Hao et al. [31], for example, derive a "generalized battery model" to predict aggregate TCL flexibility. Even with a simple single-state TCL model, summing the set of flexible trajectories involves an arduous Minkowski sum that they approximate through bounding sets. Recent work by Tindemans et. al. pursues a stochastic single TCL model that can be distributed [89]. However, this model is mathematically formulated as a partial differential equation that fundamentally relies on a single state to represent temperature. In this manuscript, we pursue a method extendible to the multi-state models that characterize data collected from real-world TCLs [29, 72, 80].

An additional limitation of linear models is that they permit the TCLs to short-cycle. Short-cycling is a behavior in which a TCL turns on and/or off for a short amount of time. This behavior is produced by linear controllers and optimization techniques when it is optimal for the temperature to oscillate around a point, such as the edge of the dead-band or the temperature setpoint. Over time, this short-cycling will reduce the efficiency and operational life of the hardware within a TCL. Efforts to prevent short-cycling, such as preferential binning, priority/preferential switching, and lockout estimation, are made in [67, 66, 110, 31]. However, the preferential techniques employed in [67, 66] cannot guaranteed the prevention of short cycling and the lockout estimation in [110, 31] requires centralized knowledge of the minimum dwell times of every agent in the population.

A key advantage of decentralized TCL control methods is the reduced or eliminated need for communication infrastructure. However, by relying on system frequency as the control signal, applications are limited to frequency regulation and real-time load shaping. To produce long-term responses (e.g. load-shifting), it is necessary for a grid entity to define the service objective, to forecast network states, and to coordinate or otherwise control the TCL population to meet the objective. Thus, the control paradigm shifts from decentralized to centralized or distributed control.

To control a TCL population to produce long-term responses in a manner that is agnostic of the individual TCL models (e.g. device specific, nonlinear, nonparametric) and that enables the incorporation of locally defined constraints (e.g. short-cycling), this manuscript presents a novel TCL modeling technique and distributed control approach. This work diverges from the above literature in the following respects:

- This chapter presents a distributed control scheme with a centralized aggregator via ADMM. Related distributed control schemes use consensus coordination [105], distributed model predictive control [54, 90], and iterative load profile aggregation [82].
- In this chapter, all TCL parameters, objectives, and constraints remain private. Each TCL is simulated locally and independently of the population. The only information that a TCL communicates with the central aggregator is its predicted power trajectory. Therefore, if necessary, TCL parameter identification can be performed locally [14].
- We do not employ an aggregate model of the TCL population. Thus, rather than modeling the entire population, the central aggregator is only responsible for updating an incentive variable that drives the population towards a desired behavior.
- There is no requirement that each TCL in the population employs the same model structure or local control scheme. The only requirement is that the TCL is able to produce predictions of its power demand under multiple alternative control scenarios. While we employ a hybrid state TCL model in this manuscript, this is not restrictive and the distributed optimization technique is compatible with a variety of different TCL modeling approaches.
- We do not use continuous setpoint control. In this chapter, all temperature setpoint offsets are integer valued and therefore easily implementable.
- Individual TCLs are not required to participate at every time step. Because the TCL population is not centrally modeled, the distributed scheme is robust to an arbitrarily large loss or acquisition of agents.
- Our proposed modeling and control approach is capable of honoring non-convex constraints, such as minimum dwell time - a critically important practical constraint that eliminates compressor short-cycling.
- Our proposed modeling and control approach is directly extendible to multi-state and nonlinear TCL models that characterize many TCLs in practice, as shown by the system identification studies in [29, 72, 80].

For the distributed optimization of a TCL population, we present a variant of the alternating direction method of multipliers (ADMM) algorithm known as sharing ADMM [9]. Due to its parallelizability and convergence characteristics, the sharing ADMM algorithm is generally applicable to the minimization of distributed agents. Furthermore, past research on the application of ADMM to the balancing of generators, fixed loads, deferrable loads, and storage devices has demonstrated the suitability of ADMM to efficiently solve large convex optimization problems in parallel [47]. In this chapter, we develop a formulation of the ADMM algorithm to enable a TCL population to perform 5-minute power generation following. Under our proposed control scheme, each TCL optimizes its behavior according to both a private objective function (which primarily enforces feasibility) and a shared objective function (which follows a generation signal). Optimization is achieved by iteratively updating a shared incentive variable, which is calculated and broadcast by a central aggregator, until the population converges to a feasible solution.

Chapter Outline

This chapter is organized as follows. Section 6.2 discusses the TCL model and section 6.3 overviews the sharing ADMM algorithm. Section 6.4 formulates sharing ADMM for distributed TCL control. Section 6.5 provides numerical examples of our proposed algorithms, and highlights its applicability to highly heterogeneous populations. Finally, Section 6.6 summarizes key results. Nomenclatures and notation used in this chapter are defined in the Appendix C.

6.2 TCL Model and Optimization

Hybrid State Model

Each TCL is modeled using the hybrid state discrete time model (5.24). We assume that θ_3 is normally distributed with variance $\Delta t \sigma^2$ (bulk units of °C^2). In this chapter, we assume a noise standard deviation σ of 0.01°C/$\sqrt{sec}$ or 0.6°C/$\sqrt{hr}$ [18].

The sign conventions in (5.24) assume that the TCL is providing a cooling load and that P (and thus θ_2) is negative. Therefore, we expand the m-update statement to account for both heating and cooling loads. Additionally, in this chapter, the optimal control of each TCL is based on setpoint manipulation. In other words, at each time step n, a TCL will either enforce T_{set} or move the setpoint by u^n. While we define u^n such that the setpoint may be adjusted at each time step, in practice, we employ a single adjustment over multiple consecutive time steps. The TCL model can now be expressed as

$$
\begin{aligned}
T^{n+1} &= \theta_1 T^n + (1-\theta_1)(T^n_\infty + \theta_2 m^n) + \theta_3 \\
m^{n+1} &= \begin{cases}
1 & \text{if } \theta_2 > 0 \text{ and} \\
 & \quad T^{n+1} < T_{set} - \frac{\delta}{2} + u^n \\
0 & \text{if } \theta_2 > 0 \text{ and} \\
 & \quad T^{n+1} > T_{set} + \frac{\delta}{2} + u^n \\
1 & \text{if } \theta_2 < 0 \text{ and} \\
 & \quad T^{n+1} > T_{set} + \frac{\delta}{2} + u^n \\
0 & \text{if } \theta_2 < 0 \text{ and} \\
 & \quad T^{n+1} < T_{set} - \frac{\delta}{2} + u^n \\
m^n & \text{otherwise}
\end{cases}
\end{aligned}
\tag{6.1}
$$

where $u^n \in \mathbf{R}$ is the setpoint change at time step n. While u^n may, by definition, take on any value in $\mathbf{R}$. In this chapter, we will only consider integer changes to the temperature setpoint (i.e. $u^n \in \mathbf{Z}$).

As noted in [17, 69], the discrete time model implicitly assumes that all changes in mechanical state occur on the time steps of the simulation. In this chapter, we will assume that this behavior reflects the programming of the systems being modeled. In other words, we will assume that the TCLs have a thermostat sampling frequency of $1/h$ Hz or once per minute.

Finally, in this chapter, we will emphasize *heterogeneous* TCLs populations and thus vary R, C, P, and COP for each agent in the population, as discussed in Section IV. Because R, C, and P define the thermal mass and rate of heat transfer, the parameters govern the system dynamics. The COP parameter does not impact the system dynamics but rather scales the magnitude of the electricity power demand.

6.3 Alternating Direction Method of Multipliers

In this section, we briefly cover the *alternating direction method of multipliers* (ADMM) algorithm for convex optimization. We refer the reader to [9, 27] for a more complete description of the algorithm. Next, we discuss a special case of block separable problems referred to as *sharing* ADMM [9]. We derive a formulation of the sharing ADMM algorithm suitable for the distributed optimization of TCLs and present primal and dual residual equations and stopping criteria not found in [9].

ADMM

The *alternating direction method of multipliers* is a common splitting method for solving problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array} \tag{6.2}$$

with variables $x \in \mathbf{R}^{N_x}$ and $z \in \mathbf{R}^{N_z}$, where $f : \mathbf{R}^{N_x} \to (-\infty, \infty]$ and $g : \mathbf{R}^{N_z} \to (-\infty, \infty]$ are closed convex functions, $A \in \mathbf{R}^{N_c \times N_x}$ and $B \in \mathbf{R}^{N_c \times N_z}$ are linear operators, and $c \in \mathbf{R}^{N_c}$ is a vector. ADMM is a variant of the augmented Lagrangian approach which uses partial updates of the dual variables at each iteration. The algorithm optimizes the coupled problem (6.2) by solving the uncoupled unscaled steps

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_x \; f(x) + \langle \lambda^k, Ax \rangle \\ &\quad + \frac{\rho}{2}\|Ax + Bz^k - c\|_2^2 && \text{(6.3a)} \\ z^{k+1} &= \operatorname*{argmin}_z \; g(z) + \langle \lambda^k, Bz \rangle \\ &\quad + \frac{\rho}{2}\|Ax^{k+1} + Bz - c\|_2^2 && \text{(6.3b)} \\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c) && \text{(6.3c)} \end{aligned}$$

where variable $\lambda \in \mathbf{R}^{N_c}$ is the dual variable, constant $\rho > 0$ is the augmented Lagrangian parameter, also referred to as the penalty parameter, and k is the integer valued iteration of the ADMM algorithm.

The necessary and sufficient optimality conditions for the ADMM problem (6.3) are given by the primal feasibility,

$$Ax^* + Bz^* - c = 0 \tag{6.4}$$

and dual feasibility,

$$0 = \nabla f(x^*) + A^T \lambda^* \tag{6.5}$$

$$0 = \nabla g(z^*) + B^T \lambda^* \tag{6.6}$$

assuming f and g are differentiable.

The convergence of (6.3) can be summarized by

- Objective Convergence: $f(x^k) + g(z^k) \to J^*$ as $k \to \infty$ where J^* denotes the optimal value of (6.2)
- Primal Residual Convergence: Residual $r^k \to 0$ as $k \to \infty$ where $r^k = Ax^k + Bz^k - c$
- Dual Variable Convergence: Variable $\lambda^k \to \lambda^*$ as $k \to \infty$

We refer the reader to [9, 27] for a discussion of the augmented Lagrangian, scaled form, primal and dual residuals, and convergence rates.

Sharing ADMM

In this chapter, we consider an ADMM-based method for solving the generic *sharing* problem using distributed optimization, as presented in [9]. In this section, we demonstrate how the *sharing* problem can be represented as a special case of (6.2) where f and A have a separable structure that we can exploit. The method is well suited for solving problems of the form

$$\text{minimize} \quad \textstyle\sum f_i(x_i) + g(\sum x_i) \tag{6.7}$$

with variables $x_i \in \mathbf{F}_i^{N_x}$, the decision variable of agent i for $i = 1, \ldots, N$, where $\mathbf{F}_i$ represents the convex constraint set of agent i, N the number of agents in the network, N_x is the length of x_i, f_i is the cost function for agent i, and g is the shared objective function of the network. The function g takes as input the sum of the individual agent's decision variables, x_i. The sharing problem allows each agent in the network to minimize its individual/private cost $f_i(x_i)$ as well as the shared objective $g(\sum x_i)$.

By introducing variable $z_i \in \mathbf{R}^{N_x}$, a term that copies the x_i decision variable of each agent, the sharing problem can be written in an ADMM-compatible form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum f_i(x_i) + g(\sum z_i) \\ \text{subject to} & x_i - z_i = 0,\ i = 1, \ldots, N \end{array} \tag{6.8}$$

with variables $x_i \in \mathbf{F}_i^{N_x}$, $z_i \in \mathbf{R}^{N_x}$, $\sum z_i \in \mathbf{G}^{N_x}$ for $i = 1, \ldots, N$ where $\mathbf{G}^{N_x}$ represents the convex constraint set of the shared objective. Therefore, the unscaled form of sharing ADMM is

$$\begin{aligned} x_i^{k+1} &= \underset{x_i}{\text{argmin}}\ f_i(x_i) && (6.9a)\\ &\quad + \langle \lambda_i^k, x_i \rangle + \frac{\rho}{2}\|x_i - z_i^k\|_2^2 \\ z^{k+1} &= \underset{z}{\text{argmin}}\ g(\textstyle\sum z_i) && (6.9b)\\ &\quad + \ \textstyle\sum \left(\langle \lambda_i^k, -z_i \rangle + \frac{\rho}{2}\|x_i^{k+1} - z_i\|_2^2\right) \\ \lambda_i^{k+1} &= \lambda_i^k + \rho(x_i^{k+1} - z_i^{k+1}) && (6.9c) \end{aligned}$$

with variable $z = (z_1, \ldots, z_N)$ and augmented Lagrangian parameter $\rho > 0$. Unlike (6.3), where there is a single globally defined dual variable λ, in (6.9), each agent has its own λ_i. Thus, the x_i-update and λ_i-update steps can be executed by each agent $i = 1, \ldots, N$ independently and in parallel. The z-update step is executed by a *collector* or *aggregator* with knowledge of each agent's decision variable x_i.

Sharing ADMM Residuals

Next, we define the sharing ADMM residuals. The necessary and sufficient optimality conditions for the sharing ADMM algorithm and derivation of the residuals are presented in Appendix C. The primal residual is defined as

$$r_i^{k+1} = x_i^{k+1} - z_i^{k+1} \tag{6.10}$$

and the dual residual as

$$s_i^{k+1} = -\rho(z_i^{k+1} - z_i^k) \tag{6.11}$$

Stopping Criteria

We define the stopping criteria as presented in [9] by

$$\|r^k\|_2 \le \epsilon^{primal} \text{ and } \|s^k\|_2 \le \epsilon^{dual} \tag{6.12}$$

where $r^k = (r_1^k, \dots, r_N^k)$, $s^k = (s_1^k, \dots, s_N^k)$, and $\epsilon^{primal} > 0$ and $\epsilon^{dual} > 0$ are feasibility tolerances for the primal and dual conditions (6.4) and (6.5). In this chapter, we set $\epsilon^{primal} = \epsilon^{dual} = 1$.

Averaged Sharing ADMM

As written, the sharing ADMM algorithm (6.9) requires the local calculation of a z_i^k, λ_i^k, and r_i^k term for each agent $i = 1, \dots, N$ in the network. Next, we will simplify the algorithm by introducing global variables $\bar{x}^k$, $\bar{z}^k$, and $\bar{\lambda}^k$ representing the arithmetic mean of all x_i^k, z_i^k, and λ_i^k, respectively. The unscaled form of the averaged sharing ADMM algorithm is given below. The derivation of the averaged sharing ADMM algorithm is presented in Appendix C.

$$\begin{aligned} x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}\ & f_i(x_i) + \langle \bar{\lambda}^k, x_i \rangle && (6.13a) \\ & + \frac{\rho}{2}\|x_i - x_i^k + \bar{x}^k - \bar{z}^k\|_2^2 \\ \bar{z}^{k+1} = \underset{\bar{z}}{\operatorname{argmin}}\ & g(N\bar{z}) + \langle \bar{\lambda}^k, -N\bar{z} \rangle && (6.13b) \\ & + \frac{N\rho}{2}\|\bar{x}^{k+1} - \bar{z}\|_2^2 \\ \bar{\lambda}^{k+1} = \ & \bar{\lambda}^k + \rho(\bar{x}^{k+1} - \bar{z}^{k+1}) && (6.13c) \end{aligned}$$

With this averaged sharing ADMM form, the individual agents no longer update their own λ_i variable. Instead, a single aggregator updates $\bar{\lambda}$, along with $\bar{x}$ and $\bar{z}$, and reports these global variables to every agent in the network.

Averaged Sharing ADMM Residuals

In order to apply the stopping criteria (6.12), we must redefine the primal and dual residuals for the averaged form. The derivation of the averaged residuals is presented in Appendix C. The averaged primal residual is defined as

$$r_i^{k+1} = \bar{x}^{k+1} - \bar{z}^{k+1} \tag{6.14}$$

and the averaged dual residual as

$$\begin{aligned} s_i^{k+1} = \rho(&(\bar{x}^{k+1} - \bar{x}^k) \\ &- (x_i^{k+1} - x_i^k) \\ &- (\bar{z}^{k+1} - \bar{z}^k)) \end{aligned} \tag{6.15}$$

The corresponding ℓ_2-norms of the stopping criteria are therefore

$$\begin{aligned} \|r^k\|_2 &= N\|\bar{x}^k - \bar{z}^k\|_2 \\ \|s^k\|_2 &= \sum \|s_i^k\|_2 \end{aligned} \tag{6.16}$$

6.4 Distributed TCL Optimization for Generation Following

In this section, we introduce the ACT representation into the ADMM algorithm. Note that we refer to the optimal power demand profile ($p = \mathbf{P}^T w$) produced by (5.7) as the *discrete* solution ($w^* \in \{0,1\}^{N_d}$), by (5.8) as the *continuous* solution ($\hat{w}^* \in \mathbf{R}^{N_d}$), and by (5.8) and (5.9) as the *probabilistic* solution ($\tilde{w} \in \{0,1\}^{N_d}$). Next, we describe the application of the sharing ADMM algorithm to the distributed optimization of TCLs with the objective of providing 5-minute ahead generation following ancillary services. Specifically, we define the optimization program for the individual TCLs and the aggregator. Then, we describe the final sharing ADMM algorithm for the TCL population. Results from multiple studies are described in the next section. Our formulation is based on the following assumptions:

1. Each TCL is capable of (i) manipulating its setpoint by a discrete/integer amount, (ii) accurately monitoring and forecasting its power demand, (iii) solving convex programs, and (iv) communicating with a central aggregator (representing a load-serving entity such as an electric utility).

2. The consumer is indifferent to the relative energy costs of the alternative control trajectories. In other words, either the consumer does not pay for energy used by the TCL or the compensation for participating in the demand response program is such that the change in energy cost is negligible. This does not imply that each alternative trajectory is of equal utility.

3. At each ADMM iteration and time step, a TCL's decision variable and selected power demand trajectory is shared with only the aggregator. The TCL's characteristics and decision making, including the $\mathbf{P}$ matrix, remain private to that TCL.

TCL Optimization

In this chapter, we consider four types of thermostatically controlled loads: refrigerators, electric water heaters, heat pumps, and electric baseboard heaters. Each TCL is simulated using (6.1) with published model parameter ranges, given in Table 6.1 and adopted from [64]. To generate a population, parameters are randomly drawn from a uniform distribution between the maximum and minimum values shown in the table. For heat pumps and

Parameter	Refrigerator	Water Heater	Heat Pump	Baseboard
Thermal resistance, R (°C/kW)	[80, 100]	[100, 140]	[1.5, 2.5]	[1.5, 2.5]
Thermal capacitance, C (kWh/°C)	[0.4, 0.8]	[0.2, 0.6]	[0.15, 0.25]	[0.15, 0.25]
Energy transfer rate, P (kW)	[-1, -0.2]	[4, 5]	[14, 25.2]	[0.5, 1.5]
Coefficient of performance, COP	2	1	3.5	1
Temperature set point, T_{set} (°C)	[1.7, 3.3]	[43, 54]	[15, 24]	[15, 24]
Dead-band width, δ (°C)	[1, 2]	[2, 4]	[0.25, 1]	[0.25, 1]
Ambient temperature, T_∞ (°C)	20	20	variable	variable
Number of zones	1	1	[5,10]	[1,2]
Number of trajectories, N_a	3	3	3	3
Feasible set point changes, S_u (°C)	{0, -2, 1}	{0, -5, 5}	{0, -2, 1}	{0, -2, 1}

Table 6.1: TCL parameter ranges adopted from [64]

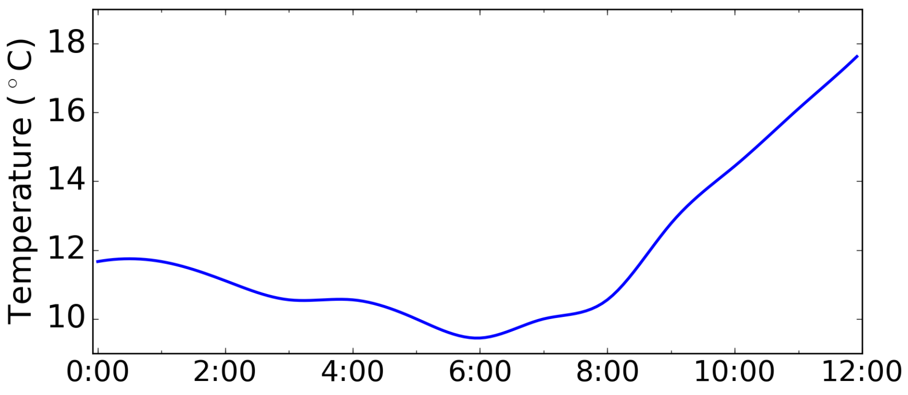

Figure 6.1: Ambient Temperature Data for Berkeley, CA, on the Morning of 3/19/2015

baseboard heaters, the C parameter is multiplied by the number of zones, an integer randomly drawn from the range given. Additionally, for the ambient temperature T_∞^n of the heat pumps and baseboard heaters, we utilize weather data for Berkeley, California from the morning of 3/19/2015, shown in Figure 6.1 [101]. The electric power demand of the TCL at each time step is given by (5.25).

TCL control takes the form of setpoint manipulation. Rather than considering the full set of feasible control inputs, we only consider a small subset of the feasible set. Specifically, we define $N_a = 3$ control inputs for each TCL in a population. The first control input applies no change to the temperature setpoint and corresponds to the default or normal operation of the TCL. The second input applies a setpoint change that will cause the system to turn on or stay on and is therefore expected to increase the average power demand of the TCL relative to normal operation. Conversely, the third input applies a setpoint change that will cause the system to turn off or stay off and is expected to decrease the average power demand of the TCL relative to normal operation.

To generate these control inputs, we define a discrete set of feasible/allowed setpoint changes, represented by S_u. Though we simulate the TCLs using a one minute time scale ($\Delta t = 1/60$ hours), we apply all setpoint changes over 5 consecutive time steps ($N_t = 5$). Thus, for a refrigerator with $S_u = \{0, -2, 1\}$,

$$\begin{aligned} u_1 &= (0, 0, 0, 0, 0) \\ u_2 &= (-2, -2, -2, -2, -2) \\ u_3 &= (1, 1, 1, 1, 1) \end{aligned}$$

In other words, the refrigerator has a maximum of $N_a = 3$ alternative control trajectories. As stated previously, each distinct input u_j is not guaranteed to produce a distinct output T_j, m_j, or p_j. Thus, for any given TCL, the number of *distinct* alternative control trajectories, N_d, is in the discrete set $\{1, \ldots, N_a\}$.

The zero input u_1 represents the default TCL input and is always first in the set of alternative control trajectories. If $N_d = 1$, we describe the TCL as fixed or inflexible. In other words, the TCL is at a point in its cycle such that setpoint manipulation does not impact the temperature trajectory. If $N_d = 2$ and the mean of p_2 is greater than the mean of p_1, then the TCL is only capable of increasing demand; if $N_d = 2$ and the mean of p_2 is less than or equal to the mean of p_1, then the TCL is only capable of decreasing demand. If $N_d = 3$, then the TCL is flexible and capable of increasing or decreasing demand. This classification is used to interpret results in Section V.

Thus, using the alternative control trajectory representation, we can simulate a TCL using $\mathbf{U}$ and (6.1) to output $\mathbf{T}$, $\mathbf{M}$, and $\mathbf{P}$ matrices such that $\mathbf{U}$, $\mathbf{T}$, $\mathbf{M}$, and $\mathbf{P} \in \mathbf{R}^{N_d \times N_t}$. Now, the individual TCL's optimization problem can be defined as a constrained least-squares fit.

$$\begin{aligned} \underset{\hat{w}}{\text{minimize}} \; & \alpha_x \|\mathbf{T}^T \hat{w} - T_{set}\|_2^2 \\ \text{subject to} \; & \textstyle\sum \hat{w}_j = 1 \\ & \hat{w} \geq 0 \end{aligned} \tag{6.17}$$

with variables $\mathbf{T} \in \mathbf{R}^{N_d \times N_t}$, representing the set of distinct temperature trajectories, $\hat{w} \in \mathbf{R}^{N_d}$, representing the optimal linear combination of trajectories and/or the discrete probability distribution of selecting control trajectory j for $j = 1, \ldots, N_d$, $T_{set} \in \mathbf{R}^{N_t}$ the TCL's temperature setpoint, N_t the number of time steps simulated, N_d the number of control trajectories, and α_x a weighting term for the TCL's objective. As previously described, the *continuous* solution for the power demand profile is determined by $x_i^* = \mathbf{P}^T \hat{w}_i^*$. Given $\hat{w}_i^*$

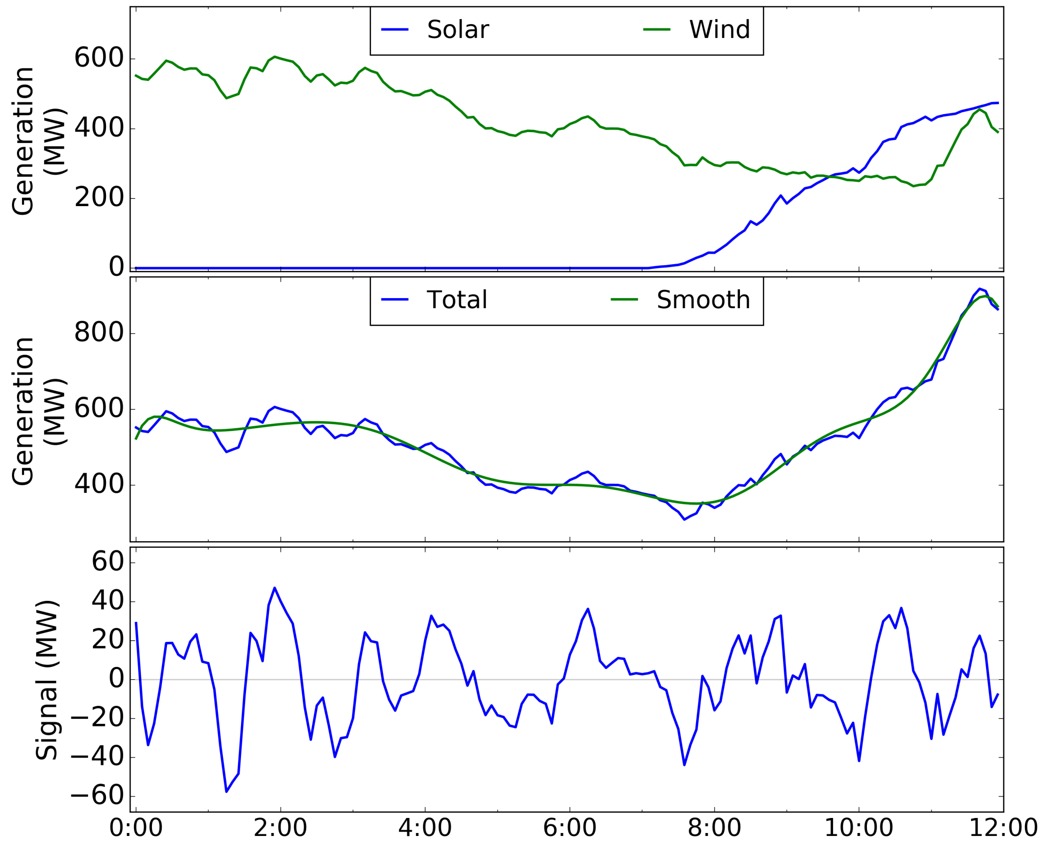

Figure 6.2: California ISO Wind and Solar Generation 5-Min Forecasts for 3/19/2015 (Top), Smooth Polynomial Fit of Total Generation (Center), and exemplary 5-minute Generation Following Signal (Bottom)

and (5.9), we denote the *probabilistic* solution as $\tilde{p}_i = \mathbf{P}^T \tilde{w}_i$. Because $\tilde{w} \in \{0,1\}^{N_d}$, $\tilde{p}$ is in the feasible set of power trajectories defined by $\mathbf{P}$. As previously stated, $\hat{w}_i^*$ and x_i^* are guaranteed to be optimal, but $\tilde{w}_i$ and $\tilde{p}_i$ may be sub-optimal.

It should be noted that the TCLs could be simulated and controlled with time steps of less than one minute without impacting the computational requirements of the distributed optimization algorithm. For example, we could simulate a TCL with a time scale of one second. To produce the alternative temperature and power trajectories required for the optimization, we would use the minute-wise averages of the simulated temperature and power demand of the TCL. In this way, the time scale used for optimization is uniform over the population while the time scale used for simulation and control is determined by the individual TCLs.

Aggregator Objective

In this chapter, the aggregator, representing a load-serving entity, will influence the behavior of the TCLs so as to perform 5-minute power generation following. To demonstrate this potential, we consider 5 minute ahead forecasts of wind and solar generation retrieved from the California Independent System Operator (ISO) [16]. Figure 6.2 presents the wind and solar power generation for the morning of 3/19/2015. The center plot shows a smooth polynomial fit of the total renewable generation. The error between the actual generation and the smooth fit will serve as our exemplary 5-minute generation following signal in this chapter, shown in the bottom plot.

Ideally, 5-minute generation following is a zero net energy service. Accordingly, the mean of the control signal is 1.229×10^{-7} MW. Considering that the signal is on the order of 10 MW and that TCLs are on the order of 1 kW loads, in this chapter, we will utilize the TCLs to respond to 1% of the signal shown in Figure 6.2. Additionally, we are simulating the TCL's using a one minute time scale but the signal is on a five minute time scale. Thus, we will treat the signal as a piecewise constant function. It is possible to interpolate between the current and previous control signal to produce a smooth or piecewise linear signal. Nonetheless, we are electing to use a piecewise constant interpretation.

To perform generation following, the aggregator's objective function can be defined as an unconstrained least-squares fit.

$$\text{minimize } \alpha_z \|\textstyle\sum x_i - d\|_2^2 \tag{6.18}$$

with variables $d \in \mathbf{R}^{N_t}$, the aggregator's desired power demand given the generation following signal $y \in \mathbf{R}^{N_t}$, and $x_i \in \mathbf{R}^{N_t}$, the power demand of TCL i for $i = 1, \ldots, N$, where N represents the number of TCLs in the network and $N_t = 5$ is the number of time steps in d and x_i. Lastly, α_z is a weighting term for the aggregator's objective.

We calculate the desired power demand d by adding the current generation following signal y to the power demand of the population in the previous time step (i.e. $d^n = \sum_i \tilde{p}_i^{n-1} + y^n$ for $n = 1, \ldots, N_t$). Since the value of the signal only changes once every 5 minutes, we optimize the aggregated power demand over a horizon of $N_t = 5$ time steps and thus,

$$d^n = \begin{cases} \sum_i \tilde{p}_i^{n-1} + y^n & \text{if } n = 1 \\ d^{n-1} & \text{otherwise} \end{cases} \tag{6.19}$$
$$\forall\ n = 1, \ldots, N_t$$

TCL Sharing ADMM

Given the TCL and aggregator optimization programs (6.17) and (6.18), we can now define the sharing ADMM algorithm for power generation following using a population of TCLs.

$$\begin{aligned}
\hat{w}_i^{k+1} &= \underset{\hat{w}_i}{\operatorname{argmin}}\ \alpha_{x,i}\|\mathbf{T}_i^T\hat{w}_i - T_{set,i}\|_2^2 && (6.20a)\\
&\quad + \ \langle\bar{\lambda}^k, \mathbf{P}_i^T\hat{w}_i\rangle + \ \frac{\rho}{2}\|\mathbf{P}_i^T\hat{w}_i - x_i^k + \bar{r}^k\|_2^2 \\
&\text{s. to} \ \ \textstyle\sum \hat{w}_j = 1, \quad \hat{w} \geq 0 \\
x_i^{k+1} &= \mathbf{P}_i^T\hat{w}_i^{k+1} && (6.20b)\\
\bar{z}^{k+1} &= \underset{\bar{z}}{\operatorname{argmin}}\ \alpha_z\|N\bar{z} - d\|_2^2 + \langle\bar{\lambda}^k, -N\bar{z}\rangle && (6.20c)\\
&\quad + \ \frac{N\rho}{2}\|\bar{x}^{k+1} - \bar{z}\|_2^2 \\
\bar{r}^{k+1} &= \bar{x}^{k+1} - \bar{z}^{k+1} && (6.20d)\\
\bar{\lambda}^{k+1} &= \bar{\lambda}^k + \rho(\bar{r}^{k+1}) && (6.20e)
\end{aligned}$$

In our implementation, the ADMM algorithm is run once every 5 minutes to determine the optimal power demand of the TCL population over the next 5 minutes at a 1 minute time scale. For simplicity, we report the power demand of the TCLs as a 5 minute average. For fixed TCLs (i.e. $N_d = 1$), the power demand profile is reported to the aggregator before the first ADMM iteration. The N and d parameters are adjusted accordingly and the ADMM algorithm run on the remaining population.

Generation Following Algorithm, Distributed Network Structure, and Communication

To achieve distributed control of a TCL population, we assume a certain amount of existing infrastructure for communication, computation, and control. Our assumptions are comparable to those made in [97, 66, 90] and include:

- Bi-directional communication between the individual TCLs and the aggregator via wired or wireless links.
- Sufficient local computation and hardware for solving convex programs and measuring TCL states.
- A local TCL model whose parameters are either known a priori or identified using a parameter estimation technique [80, 15, 14].

The execution of the generation following algorithm can be summarized by the follow 4 steps:

1. Aggregator Preparation: Every 5 minutes, the aggregator receives the signal y and produces the desired power profile d.

2. TCL Simulation: Each TCL i in the population simulates its dynamics to produces a set of alternative temperature trajectories $\mathbf{T}_i$ and power trajectories $\mathbf{P}_i$.

3. Optimization via ADMM: For each iteration k until the stopping criteria are met:

   a) Broadcast Signal: The aggregator reports the mean primal residual $\bar{r}^k$ (i.e. the difference between $\bar{x}^k$ and $\bar{z}^k$) and the mean dual incentive variable $\bar{\lambda}^k$ to each TCL i in the population.

   b) Local Optimization: Each TCL i optimizes (6.20a) and reports x_i^{k+1} (6.20b) to the aggregator.

   c) Aggregator Optimization: Given the mean TCL power profile $\bar{x}^{k+1}$, the aggregator optimizes (6.20c) and updates the mean primal residual $\bar{r}^{k+1}$ (6.20d) and the mean dual incentive variable $\bar{\lambda}^{k+1}$ (6.20e).

4. Interpretation: Each TCL i interprets the discrete probability distribution $\hat{w}_i^*$ to select a power trajectory $\tilde{p}_i$ from the set $\mathbf{P}_i$ and reports the probabilistic solution $\tilde{p}_i$ to the aggregator.

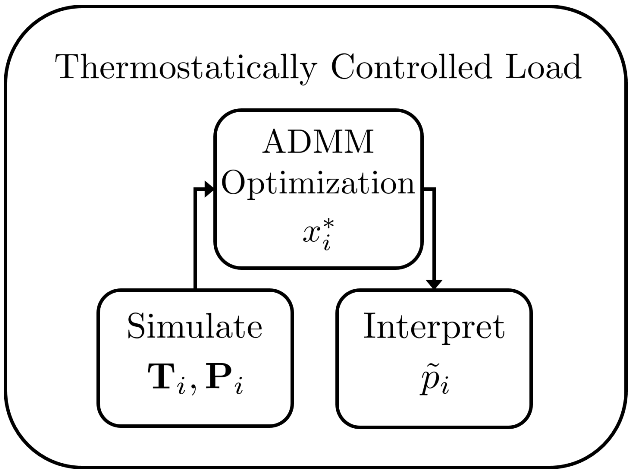
Thermostatically Controlled Load
ADMM Optimization x_i^*
Simulate $\mathbf{T}_i, \mathbf{P}_i$
Interpret $\tilde{p}_i$

Figure 6.3: TCL Model and Optimization Structure. Each TCL i in the population will simulate its dynamics to produce the alternative control trajectories, coordinate with an aggregator using the ADMM algorithm to produce a continuous solution, and finally interpret the discrete probability distribution to produce a probabilistic solution.

Figure 6.3 outlines the steps performed by each TCL i in the population. At each time step, the TCL simulates its dynamics to produce the alternative control trajectories as represented by $\mathbf{T}_i$ and $\mathbf{P}_i$, coordinates with an aggregator via ADMM to produce a

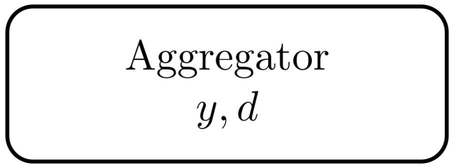

Figure 6.4: Aggregator Preparation Step. The aggregator receives the signal y and produces the desired power profile d.

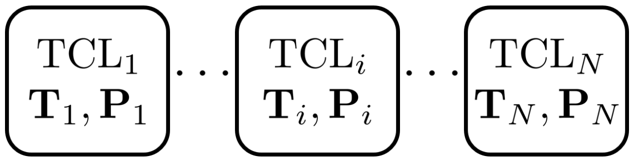
TCL_1
$\mathbf{T}_1, \mathbf{P}_1$
TCL_i
$\mathbf{T}_i, \mathbf{P}_i$
TCL_N
$\mathbf{T}_N, \mathbf{P}_N$

Figure 6.5: TCL Simulation Step. Each TCL i in the population simulates its dynamics to produce the alternative control trajectories.

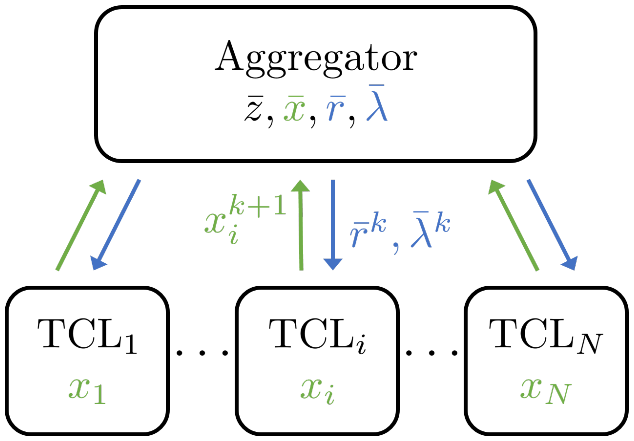

Figure 6.6: Optimization Step. For each iteration k of the ADMM algorithm, the aggregator reports the mean primal residual $\bar{r}^k$ and the mean dual incentive variable $\bar{\lambda}^k$ to each TCL i in the population. Each TCL i then reports its updated x_i^{k+1} to the aggregator.

continuous solution x_i^*, and finally interprets the discrete probability distribution $\hat{w}_i^*$ to produce a probabilistic solution $\tilde{p}_i \in \mathbf{P}_i$.

The 4 steps of the generation following algorithm, as well as the structure of the distributed system, are illustrated in Figures 6.4, 6.5, 6.6, and 6.7. In particular, the figures indicate for each step of the algorithm which variables are defined locally and which are communicated between the aggregator and the TCLs in the population.

In our algorithm, each TCL i reports the power demand profile x_i^{k+1} to the aggregator but not to the other TCLs in the network. Each TCL's $\mathbf{T}$, $\mathbf{P}$, and $\hat{w}^k$ remain private. In addition to the stopping criteria (6.12), we impose a limit on the absolute value of $\bar{\lambda}$ (i.e.

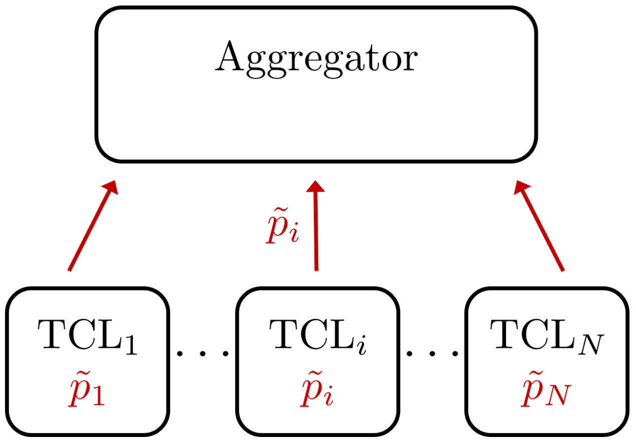

Figure 6.7: Interpretation Step. Each TCL i interprets the discrete probability distribution $\hat{w}_i^*$ to select a power trajectory $\tilde{p}_i$ from the set $\mathbf{P}_i$ and reports the probabilistic solution $\tilde{p}_i$ to the aggregator.

stop if $|\bar{\lambda}^n| \geq \lambda_+$ for $n = 1, \ldots, N_t$). This limit is empirically selected and serves as a means of detecting if the population of TCL's is able to match the signal within a certain tolerance. As defined by (6.18), any power demand is feasible, but in practice, we only want to perform generation following if the aggregate continuous solution $N\bar{x}^*$ is within a certain error tolerance, ϵ^{error}, of the control signal d (i.e. $\max(|N\bar{x}^* - d|) < \epsilon^{error}$). Therefore, if the ADMM algorithm does not converge to a solution within this tolerance, the population has failed to perform generation following and each TCL implements some default behavior. In this chapter, the default behavior is to return to the original temperature setpoint by implementing the control trajectory $u_1 = (0, 0, 0, 0, 0)$.

At optimality, the power demand profile x_i^* represents the TCL's *continuous* solution and is not directly implementable. While it is conceptually possible to cluster complementary TCLs or to incorporate energy storage so as to directly achieve the continuous solution, we assume no such coordination in this chapter. Instead, each TCL in the population will implement a single control trajectory given the discrete probability distribution $\hat{w}_i^*$. The TCLs' states are updated and the resulting power demand profile, referred to as the *probabilistic* solution $\tilde{p}_i$, is reported to the aggregator. The potential for error between the continuous and probabilistic solution is addressed in the next section.

ADMM Parameter Selection

The parameters of the ADMM algorithm are presented in Table 6.5. These parameters have been empirically selected to fit the characteristics of our application. Specifically, because the behavior of each TCL in the population is constrained by its alternative control trajectories, the goal of the generation following algorithm is primarily to shape the aggregate load in response to a signal. This is expressed in the ADMM parameters by selecting an aggregator coefficient α_z that is larger than the TCL coefficient α_x for each TCL in the population. Decreasing α_z or increasing α_x will cause less emphasis to be placed on the global objective. Therefore, the TCLs will choose to optimize their local objectives rather than optimizing the aggregate power demand (a further discussion of this behavior is presented in Section 5.7).

The primal and dual feasibility tolerances are positive values which define the stopping criteria of the ADMM algorithm. In our application, the mean primal residual $\bar{r}^k$ is the difference between $\bar{x}^k$, the mean power demand based on the continuous solutions reported by the TCL population, and $\bar{z}^k$, the mean power demand based on the solution to the aggregator's objective function. The primal feasibility tolerance ϵ^{primal} is a measure of the primal residual that we are willing to accept. Based on the relative weighting of the aggregator and TCL objectives and the error tolerance $\epsilon^{error} = 10\text{kW}$, ϵ^{primal} can effectively be any positive value less than $\sqrt{10}$. We have selected $\epsilon^{primal} = 1$ based on empirical observations that the value produces aggregate continuous solutions within the error tolerance ϵ^{error} within a modest number of ADMM iterations (i.e. <50).

The dual residual s_i^k of each TCL i is a measure of the change in the continuous solution x_i^k and in the primal residual $\bar{r}^k$ between ADMM iteration k and $k+1$. Thus, $\|s^k\|$ is a measure of the rate of change in the solutions of the aggregator and TCL population. A large dual feasibility tolerance ϵ^{dual} will cause the ADMM algorithm to stop once the primal feasibility criterion is met while a small tolerance will cause the algorithm to continue until the solutions of the aggregator and TCLs no longer change from one iteration to the next. We have empirically chosen a dual feasibility tolerance ϵ^{dual} of 1 such that the dual feasibility criterion is met a few iterations (i.e. <10) after the primal feasibility criterion.

The Lagrangian penalty has been tuned to be sufficiently large such that the ADMM algorithm converges relatively quickly but sufficiently small so as to avoid oscillatory behaviors in the ADMM updates as the algorithm begins to converge. Lastly, we have observed that when the desired power profile d is outside the feasible power demand range of the TCL aggregation, the absolute values of $\bar{\lambda}$ increase dramatically as the ADMM algorithm attempts to drive the TCL population toward an infeasible solution so as to reduce the aggregator's objective function. To detect this behavior and stop the ADMM algorithm, we impose a limit of $\lambda_+ = 50$ on the absolute value of $\bar{\lambda}$.

Reducing Variance

At optimality, the solution x_i^* represents the *continuous* solution of the relaxed form of the general assignment problem, as described in (5.8). While this relaxation is essential for distributed convex optimization, the continuous solution is not directly implementable. Instead, we employ the *probabilistic* solution $\tilde{p}_i$ and thereby introduce the potential for error between the solution returned by the ADMM algorithm and the actual power demand of the TCLs. For highly homogeneous populations of TCLs, we have observed that the aggregated continuous and probabilistic solutions are comparable (i.e. have similar errors with respect to the signal). The logical explanation is that due to the homogeneity, many TCLs converge to similar solutions. Thus, their probabilistic solutions are complementary

Generation following signal	y
Desired power demand profile	d
Temperature set point of TCL i	$T_{set,i}$
Temperature trajectories of TCL i	$\mathbf{T}_i$
Power trajectories of TCL i	$\mathbf{P}_i$

Table 6.2: Optimization Inputs

Final power demand profile of TCL i (probabilistic solution)	$\tilde{p}_i$

Table 6.3: Optimization Outputs

such that the aggregated power demand is close to the continuous solution returned by the ADMM algorithm. For highly heterogeneous populations, however, this is not the case.

One method for reducing the variance, which we will refer to as iterative optimization, is to repeatedly commit a fraction of the population to a solution and solve the optimization problem with the uncommitted population. Stated simply, we run ADMM on the entire population of TCLs. Upon convergence, we fix a certain number of the TCLs (10-20% of the total population) using the probabilistic solution. These TCLs are them removed from the population being optimized and the N and d parameters are adjusted accordingly. Next, we repeat the ADMM algorithm to find the continuous solution of the remaining population using the previous value of λ and adjusted values of $\bar{x}$ and $\bar{z}$ as a warm start. This process is repeated until all TCLs are fixed. For successive ADMM runs, we decrease the number of ADMM iterations as the problem becomes more constrained.

A second method, which we refer to as iterative variance minimization, is to gradually penalize the variance of individuals in the population, as described in Chapter 5 Section 5. Thus, the update step (6.20a) becomes

$$\begin{aligned} \hat{w}_i^{k+1} = \underset{\hat{w}_i}{\operatorname{argmin}}\ & \alpha_{x,i}\|\mathbf{T}_i^T\hat{w}_i - T_{set,i}\|_2^2 \\ &+\ \langle\bar{\lambda}^k, \mathbf{P}_i^T\hat{w}_i\rangle + \frac{\rho}{2}\|\mathbf{P}_i^T\hat{w}_i - x_i^k + \bar{r}^k\|_2^2 \\ &+\ \alpha_{\sigma,i}^k \sum\|p_{i,j} - x_i^k\|_2^2 \\ \text{s. to}\ \ & \sum\hat{w}_j = 1, \quad \hat{w} \geq 0 \end{aligned} \tag{6.21}$$

where $p_{i,j}$ is the j-th alternative power trajectory of TCL i and $\alpha_{\sigma,i}^k$ is the scaling parameter for the sparsity-inducing cost of TCL i. After each ADMM iteration, the value of $\alpha_{\sigma,i}^k$ for each TCL in the population is updated according to (5.23).

Probability distribution of TCL i at iteration k	$\hat{w}_i^k$
Power demand profile of TCL i at iteration k (continuous solution)	x_i^k
Mean TCL power demand profile at iteration k (continuous solution)	$\bar{x}^k$
Mean aggregator power demand profile at iteration k (continuous solution)	$\bar{z}^k$
Mean primal residual	$\bar{r}^k$
Mean dual variable	$\bar{\lambda}^k$

Table 6.4: ADMM Variables

Lagrangian Penalty	ρ	10
Aggregator Coefficient	α_z	20
TCL Coefficient (Refrigerator)	α_x	0
TCL Coefficient (Water Heater)	α_x	0
TCL Coefficient (Heat Pump)	α_x	1
TCL Coefficient (Baseboard Heater)	α_x	1
Primal Feasibility Tolerance	ϵ^{primal}	1
Dual Feasibility Tolerance	ϵ^{dual}	1
Error Tolerance	ϵ^{reg}	10 kW
$\bar{\lambda}$ Limit	λ_+	50

Table 6.5: ADMM Parameters

6.5 Experimental Results

In this section, we present results for 6 experimental studies. In each experiment, we model a population of TCLs to follow 1% of the signal described in Figure 6.2. This 5-minute generation following is achieved by running the sharing ADMM algorithm every 5 minutes between midnight and noon for the morning of 3/19/2015. In the first experiment, we consider a large, highly homogeneous population of refrigerators. Second, a small, heterogeneous population of refrigerators. Third, a highly heterogeneous population of refrigerators, water heaters, heat pumps, and baseboard heaters. Fourth, a highly heterogeneous population of refrigerators, water heaters, heat pumps, and baseboard heaters using the divide and conquer approach described above.

For each study, we employ the ADMM parameters in Table 6.5. For refrigerators and water heaters, $\alpha_x = 0$ indicating that the consumer is indifferent to the selection of a control trajectory. Thus, the TCL's objective function (6.17) is constant and weakly convex. At each iteration, the TCL enforces feasibility and adjusts its power demand according to the incentive signal λ. For heat pumps and baseboard heaters, $\alpha_x = 1$ indicating that the consumer would prefer to keep the temperature near the setpoint. The weight α_x is not large enough to prevent the selection of any alternative control trajectory, but rather numerically incentives the utilization of more cooperative/responsive refrigerators and water heaters before heat pumps and baseboard heaters. Lastly, S_u defines a set of 3 feasible change in setpoint values. Thus, each TCL has a maximum of $N_a = 3$ alternative control trajectories.

For each of the experimental studies, we present the aggregated power demand and response of the population for the respective experiment. The aggregated continuous and probabilistic power *demand* are presented as the mean of the total power demand over each $N_t = 5$ minute interval.

$$x_\Sigma^k = \frac{1}{N_t} \sum_{n=1}^{N_t} \sum_{i=1}^{N} (x_i^n)^* \tag{6.22}$$

$$p_\Sigma^k = \frac{1}{N_t} \sum_{n=1}^{N_t} \sum_{i=1}^{N} \tilde{p}_i^n \tag{6.23}$$

where variables $x_\Sigma^k, p_\Sigma^k \in \mathbf{R}$, N is the number of TCLs in the population, and k denotes the integer valued time step of each ADMM run (i.e. each $N_t = 5$ minute interval between midnight and noon).

The continuous and probabilistic *responses* of the population denote the change in power demand, and are respectively given by

$$x_\Delta^k = x_\Sigma^k - p_\Sigma^{k-1} \tag{6.24}$$

$$p_\Delta^k = p_\Sigma^k - p_\Sigma^{k-1} \tag{6.25}$$

Because x_i^* is not directly realizable, x_Δ^k is calculated relative to the previous probabilistic demand p_Σ^{k-1}.

For each time step k, we also present the minimum and maximum power demand that the population of TCLs *could* have achieved given the set of feasible power trajectories $\mathbf{P}_i$ for each TCL. For each TCL i, we denote the trajectories with the minimum and maximum mean power demand as $p_i^{\min} \in \mathbf{P}_i$ and $p_i^{\max} \in \mathbf{P}_i$, respectively. Therefore, the minimum and maximum mean power demand of the population is

$$p^k_{\min\Sigma} = \frac{1}{N_t}\sum_{n=1}^{N_t}\sum_{i=1}^{N}(p_i^n)^{\min} \tag{6.26}$$

$$p^k_{\max\Sigma} = \frac{1}{N_t}\sum_{n=1}^{N_t}\sum_{i=1}^{N}(p_i^n)^{\max} \tag{6.27}$$

Thus, the maximum up or down response of the population is given by

$$p^k_{\min\Delta} = p^k_{\min\Sigma} - p^{k-1}_{\Sigma} \tag{6.28}$$

$$p^k_{\max\Delta} = p^k_{\max\Sigma} - p^{k-1}_{\Sigma} \tag{6.29}$$

where variable $p^k_{\min\Delta}$ corresponds to demand decrease and $p^k_{\max\Delta}$ to demand increase (from the perspective of the load). In the case that $p^k_{\min\Delta} > 0$ or $p^k_{\max\Delta} < 0$, the population is incapable of decreasing or increasing its power demand, respectively.

Highly Homogeneous Population

To begin, we present the results using a highly homogeneous population of refrigerators. Specifically, we have modeled and controlled a population of $N = 20{,}000$ refrigerators with identical parameters (the mean of the parameter ranges in Table 6.1). We have limited the number of ADMM iterations to 10.

Figure 6.8 presents the results from the homogeneous experiment. The top plot shows how well the continuous responses x^k_Δ and the probabilistic responses p^k_Δ compare to the signal y^k for each 5 minute interval between midnight and noon. To reiterate, the continuous response is the difference between the aggregated solution to the ADMM algorithm and the power demand in the previous time step. The probabilistic response is the difference between the aggregated probabilistically selected TCL trajectories and the power demand in the previous time step. The RMSEs of the continuous and probabilistic responses are 0.11 kW and 14.25 kW, respectively. The ADMM algorithm only failed to converge to a continuous solution within the error tolerance of 10 kW during two intervals at 10:00 and 10:05 AM, resulting in a generation following success rate of 98.6% over the time period studied.

The second plot in Figure 6.8 shows the probabilistic p^k_Σ, the minimum $p^k_{\min\Sigma}$, and the maximum $p^k_{\max\Sigma}$ power demand of the population at each time interval. The third plot shows the corresponding minimum $p^k_{\min\Delta}$ and maximum $p^k_{\max\Delta}$ potential (i.e. the difference between the minimum or maximum power demand and the demand in the previous time step). While it is possible for the aggregator to discern these minimum and maximum values

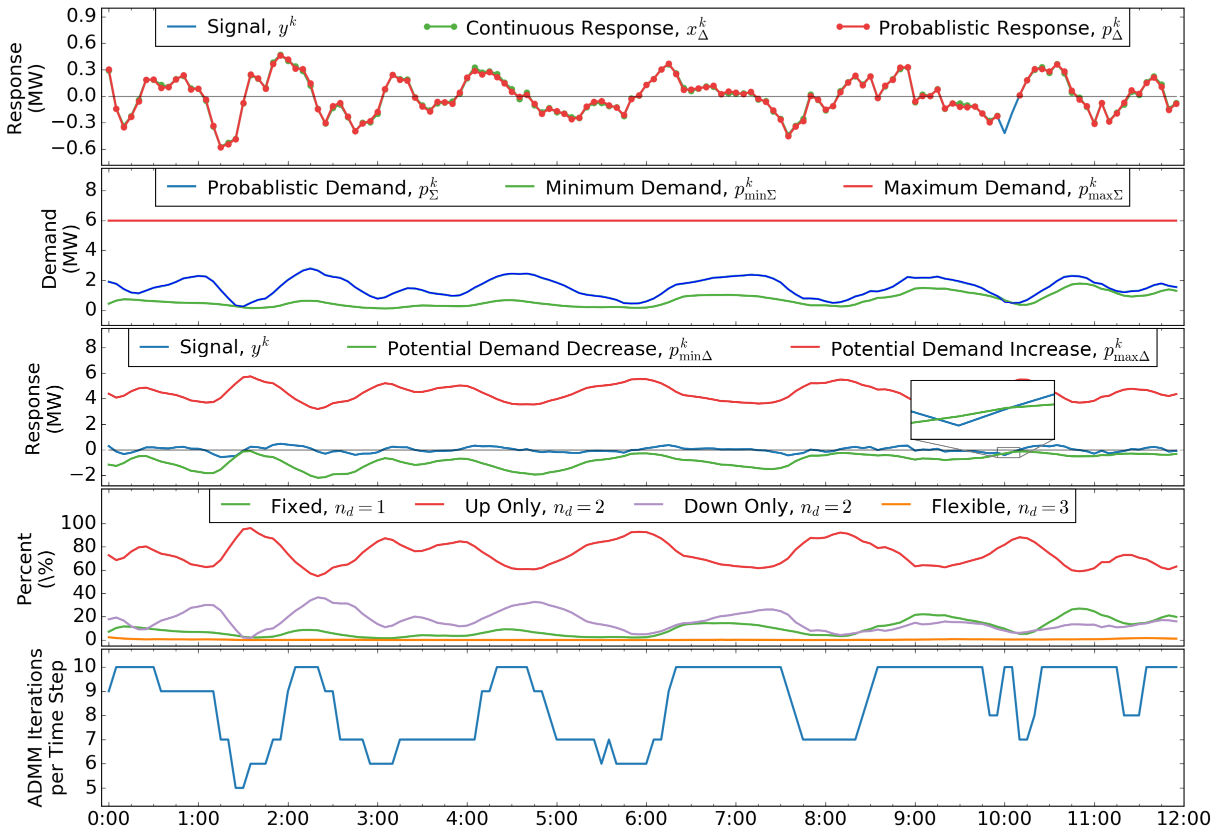

Figure 6.8: Highly Homogeneous Population

by manipulating $\bar{\lambda}$ to drive the TCLs to their extremes, we have assumed no such behavior in our implementation. Thus, the aggregator can only determine if the signal and the feasible up or down responses are within the specified error tolerance after the ADMM algorithm converges. The only exception is if $\bar{\lambda}$ violates the λ_+ limit, indicating that the ADMM algorithm is attempting to drive the population toward an infeasible solution so as to reduce the aggregator's objective function (though the TCLs will guarantee that the solution at each iteration is feasible).

The fourth plot shows the percentage of the population that is either fixed, flexible, or capable of only up or down responses. From midnight to 6:00, we observe that the TCLs move between up only and down only conditions, with the percent of fixed and flexible TCLs remaining small. After 6:00, the TCLs in the up only population begin to move to the down only or fixed populations. In the fifth plot, which shows the number of ADMM iterations executed before stopping, we see that the ADMM algorithm has more difficulty finding a solution in these later time intervals and begins hitting the iterations limit of 10. This trend represents a decline in the capability of the population to perform generation following.

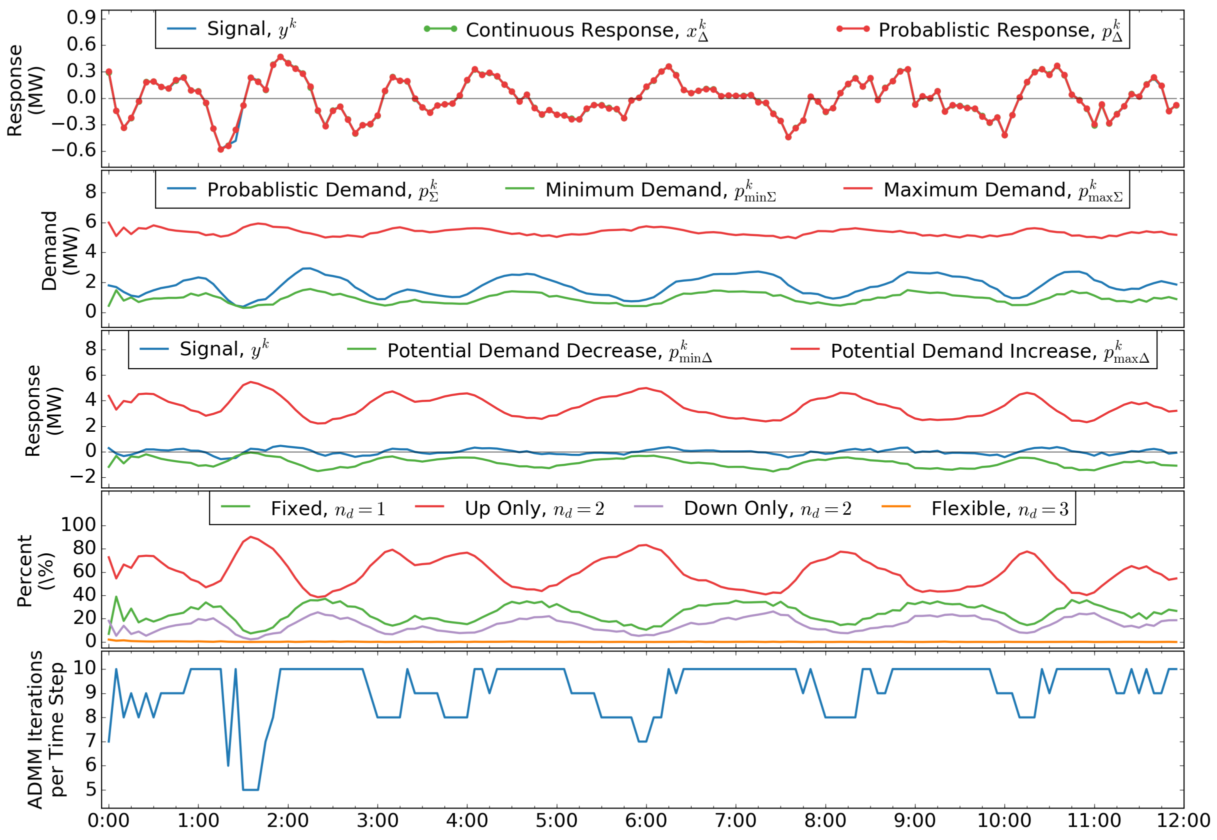

Figure 6.9: Homogeneous Population with 5 Minute Dwell Time

Homogeneous Population with Dwell Time

In this study, we demonstrate the suitability of the control framework to honor minimum dwell time constraints. We consider a homogeneous population of $N = 20{,}000$ refrigerators with identical parameters (the mean of the parameter ranges in Table 6.1). The TCLs are controlled such that a minimum dwell time of 5 minutes is enforced (i.e. if a TCL turns on or off, it must remain in the new state for at least 5 minutes). Again, we have limited the number of ADMM iterations to 10.

The minimum dwell time constraint is applied at the Simulate TCLs step of the generation following algorithm. Specifically, if a TCL simulation produces a mechanical state trajectory m_j such that the minimum dwell time of 5 minutes would be violated if the trajectory was implemented, the trajectory is discarded by excluding the corresponding u_j, T_j, m_j, and p_j from $\mathbf{U}$, $\mathbf{T}$, $\mathbf{M}$, and $\mathbf{P}$.

The results, presented in Figure 6.9, show a generation following success rate of 100.0% over the time period studied. The RMSEs of the continuous and probabilistic responses are 8.13 kW and 11.80 kW, respectively. Note that this study employs the same number of

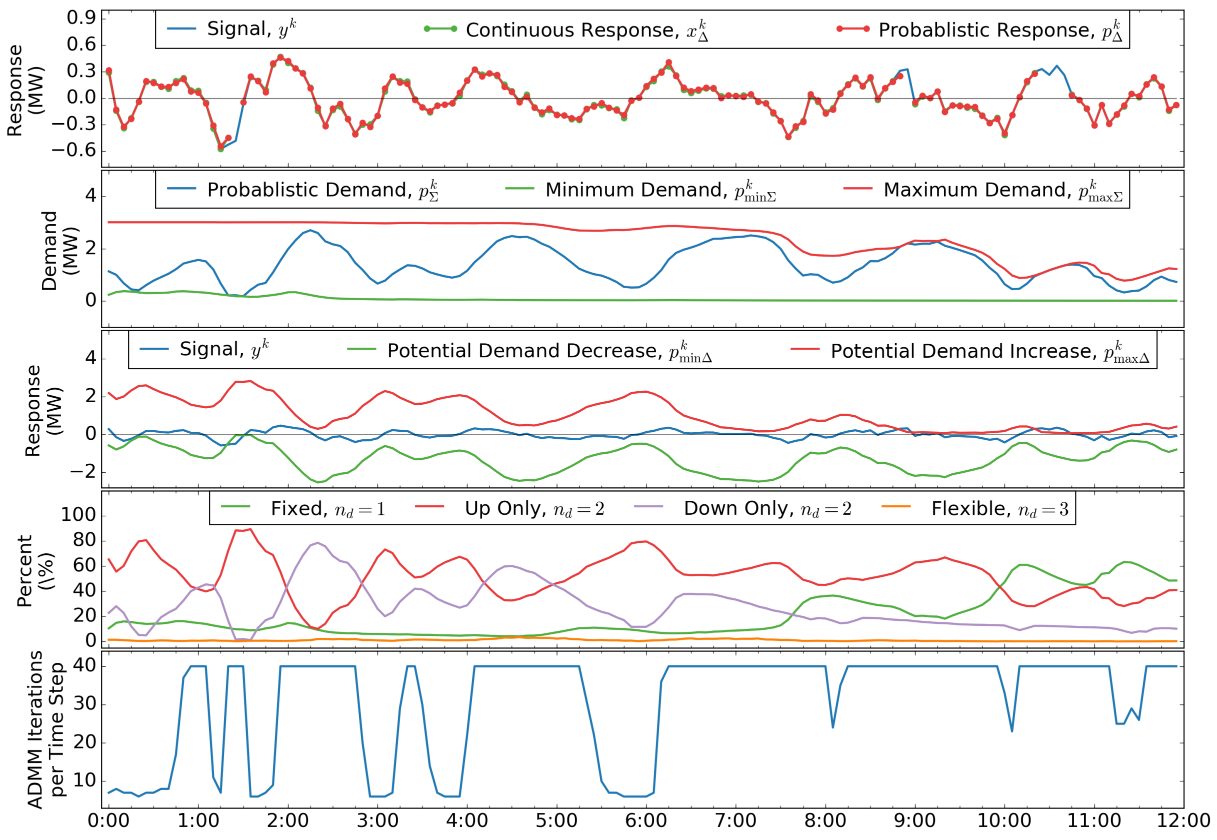

Figure 6.10: Heterogeneous Population

TCLs with the same parameter values as those in the previous homogeneous study. However, due to the enforcement of the dwell time constraint, we observe a greater percentage of the population in the fixed and down only conditions. In the previous homogeneous study, the means of the fixed, up only, and down only populations over the 12 hours were 9.35%, 74.09%, and 16.23%, respectively. With the enforcement of the dwell time, the mean percentages are 24.87%, 60.22%, and 14.74%, respectively. Due in part to the increase in the fixed population, more ADMM iterations are required to find a solution.

Heterogeneous Population

To begin introducing heterogeneity, we have modeled the control of $N = 10{,}000$ refrigerators with parameters randomly drawn from the uniform distributions in Table 6.1. We have also raised the ADMM iterations limit to 40. The results from this study are presented in Figure 6.13 and show a success rate of 95.8% over the time period studied. The RMSEs of the continuous and probabilistic responses are 8.81 kW and 17.84 kW, respectively.

In this study, we have significantly decreased the population size and thus the potential

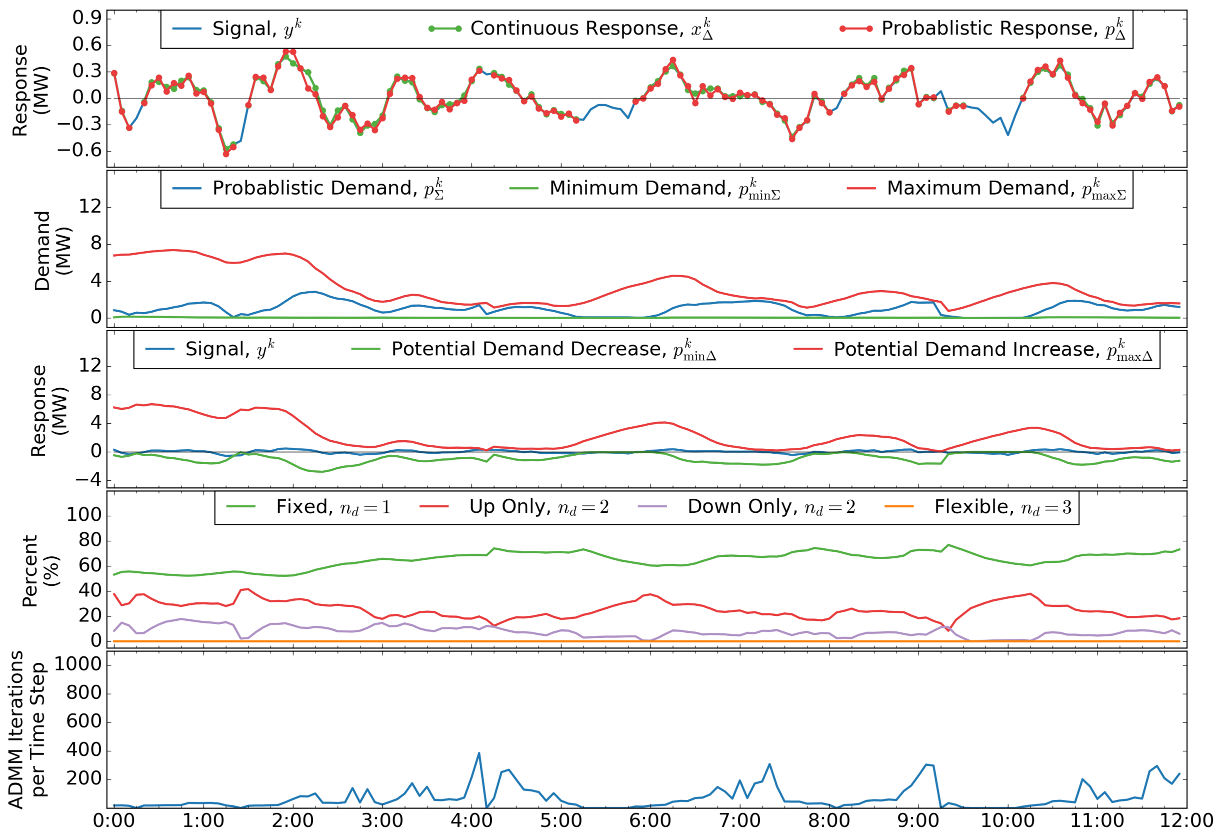

Figure 6.11: Highly Heterogeneous Population

for increasing demand. The second and third plots indicate that as we approach noon, we experience a decline in the maximum feasible power demand $p^k_{\max\Sigma}$ and the demand increase potential $p^k_{\max\Delta}$. The fourth plot shows the percentage of the population that is either fixed, flexible, or capable only up or down responses and presents some insight into the loss of demand increase potential. Between midnight and 7:00, we observe that the TCLs generally oscillate between up only and down only, with the percent of fixed and flexible TCLs remaining small. After 7:00, the TCLs in the down only population begin to become fixed. Finally, the TCLs begin switching between up only and fixed, making it more difficult to perform generation following and driving up the number of ADMM iterations.

Highly Heterogeneous Population

In this study, we consider a highly heterogeneous population of refrigerators, water heaters, heat pumps, and baseboard heaters with parameters randomly drawn from the uniform distributions in Table 6.1. We model 3,000 refrigerators, 2,000 water heaters, 1,800 heat pumps, and 1,800 baseboard heaters for a total of N = 8,600 TCLs. We set the ADMM iterations limit to 1000 to increase the likelihood that the algorithm converges to a solution rather than terminating due to the iterations limit.

The results, presented in Figure 6.11, show a generation following success rate of 86.8% over the time period studied. The RMSEs of the continuous and probabilistic responses are 4.83 kW and 37.16 kW, respectively. This increase in the error of the probabilistic response can be attributed to the increased heterogeneity of the TCL population.

The fourth plot in Figure 6.11 shows that at each time interval, the percentage of fixed TCLs remained over 50%. Nonetheless, the potential for increasing the demand fluctuated between 1 MW and 6 MW. Overall, the population suffered from an insufficient potential for decreasing demand. This could be addressed by better conditioning the TCLs so that more remain in a flexible or down only condition or by extending the forecasting horizon beyond the next 5 minutes, allowing the aggregator and TCLs to better prepare for future signals.

Heterogeneous Population with Iterative Optimization

To address the error between the probabilistic response p_{Δ}^{k} and the signal y^{k}, we have re-simulated the highly heterogeneous population of $N = 8{,}600$ TCLs using the iterative optimization method for reducing variance. In other words, we have run the ADMM algorithm 20 times at each 5-minute interval. After each run, we fixed 5% of the total population so that after the final run, all 8,600 TCLs are fixed. Additionally, between each run, the N and d parameters are adjusted according to the results of the newly fixed TCLs. Lastly, as a warm start, the previous value of λ and adjusted values of $\bar{x}$ and $\bar{z}$ are employed to initialize the next ADMM run. If the error tolerance is violated at the end of an ADMM run, the algorithm is terminated. For the first ADMM run, the iteration limit is set to 200. For successive ADMM runs, the limit is 100.

To improve the performance of the algorithm, we have sorted the TCLs such that those with the highest power demand are fixed first and those with the lowest are fixed last. In other words, the order of consideration is heat pump, electric water heater, electric baseboard heater, and refrigerator.

The test results are presented in Figure 6.12. Note that while we are using a population with the same number of TCLs as the previous study, the model parameters were randomly drawn from the ranges in Table 6.1 for each study. Therefore, the populations exhibit different dynamics. While the iterative optimization method has resulted in a significant increase in the total number of ADMM iterations at each time interval, the RMSEs of the continuous and probabilistic responses have been reduced to 3.51 kW and 4.12 kW, respectively. This demonstrates that the TCLs can be controlled such that the probabilistic response p_{Δ}^{k} is within the error tolerance of 10 kW. The success rate for the time period simulated is 85.4%. Once again, the population struggles to match the required demand decrease.

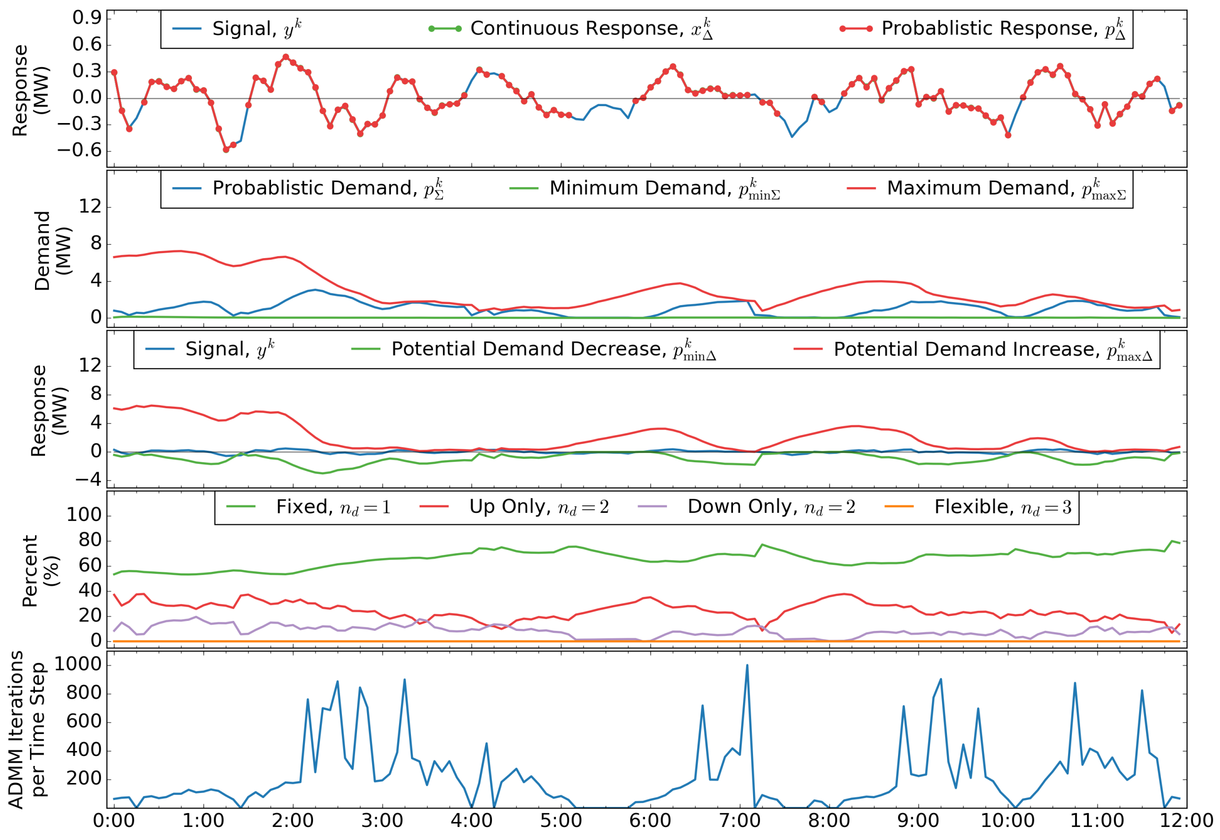

Figure 6.12: Highly Heterogeneous Population with Iterative Optimization

Heterogeneous Population with Iterative Variance Minimization

Finally, we have re-simulated the highly heterogeneous population of $N = 8{,}600$ TCLs using the iterative variance minimization approach. Specifically, we replace the update step (6.20a) with (6.21), which includes a linear cost associated with the variance of the solution. After each iteration, we update the scaling parameter $\alpha^k_{\sigma,i}$ for each TCL according to (5.23). In this study, we set σ^2_{max}, the maximum desired variance, to 25 kW^2. Once again, we note that the model parameters were randomly drawn from the ranges in Table 6.1 for each study and therefore exhibit different dynamics than the previous studies.

The test results are presented in Figure 6.13. As was the case with the iterative optimization method, we observe a significant increase in the number of ADMM iterations at each time interval. However, with the iterative variance minimization method, we are able to reduce the variance of the probabilistic solution such that the RMSEs of the continuous and probabilistic responses are 4.16 kW and 5.08 kW, respectively. This demonstrates that the TCLs can be controlled such that the probabilistic response p^k_Δ is within the error tolerance of 10 kW. The success rate for the time period simulated is 79.86%. Further research is

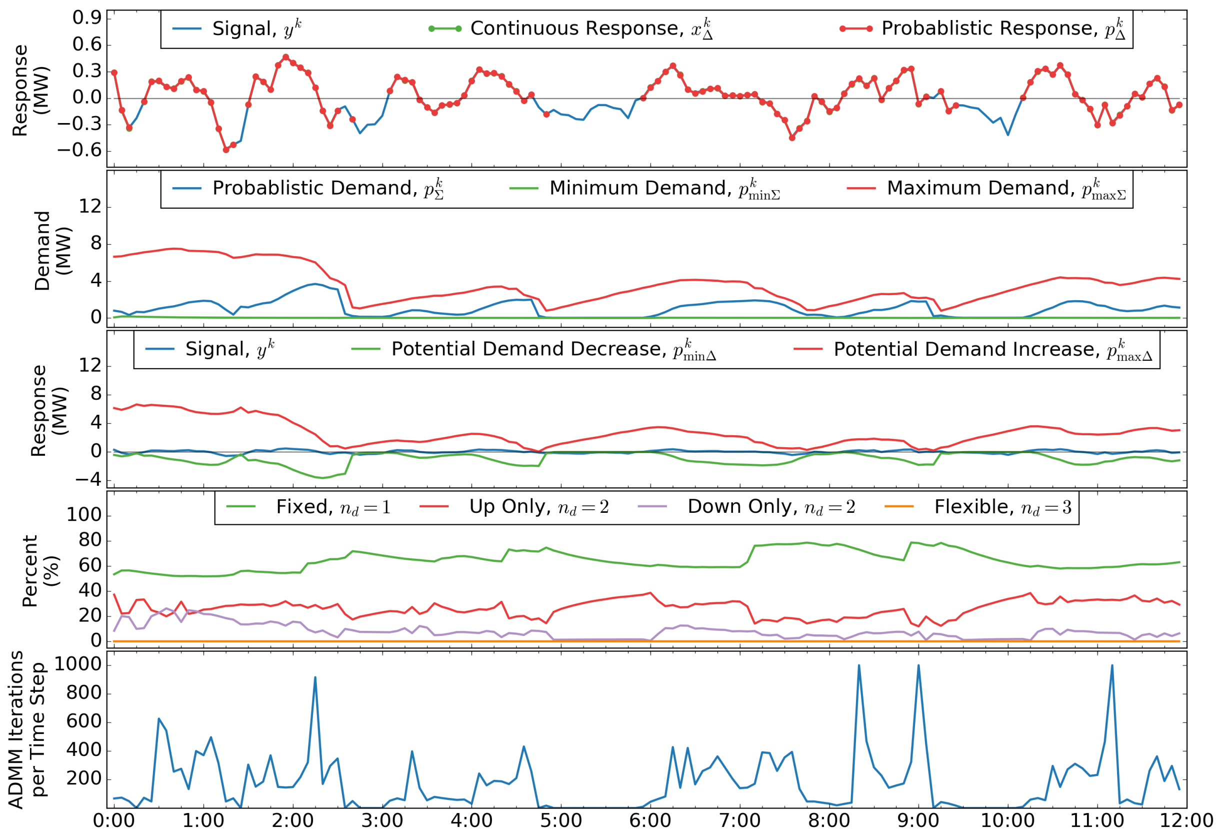

Figure 6.13: Highly Heterogeneous Population with Iterative Variance Minimization

required to assess the relative advantages and disadvantages of the iterative optimization and iterative variance minimization methods.

Increasing Population Size

To test the impact of population size on the number of ADMM iterations, we have designed an experiment in which TCL populations of varying size are employed to respond to the load following signal. Each population is comprised of homogeneous refrigerators with identical parameters (the mean of the parameter ranges in Table 6.1).

To account for the variation in population size, the percentage of the signal followed by the aggregator is scaled such that the per TCL signal remains constant across the different populations. The same is done with the error tolerance of the aggregator. Specifically, the percentage of the signal is defined as 10^{-4}% per TCL and the error tolerance is 10^{-4} kW or 0.1 W per TCL. Therefore, 100 TCLs are employed to follow 0.01% of the signal with an error tolerance of 0.01 kW and 1,000,000 TCLs are employed to follow 100% of the signal with an error tolerance of 100 kW.

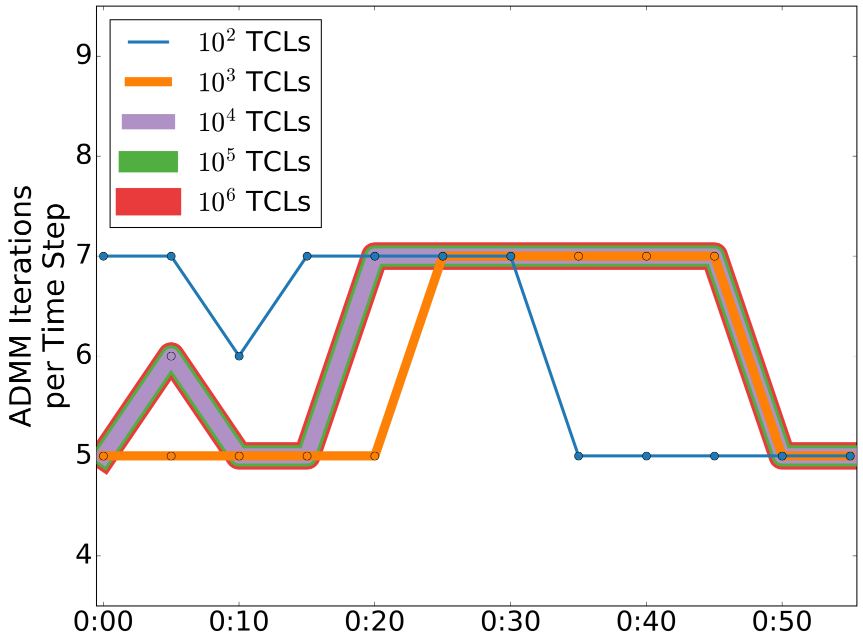

Figure 6.14: ADMM iterations at each time step with TCL populations of varying size

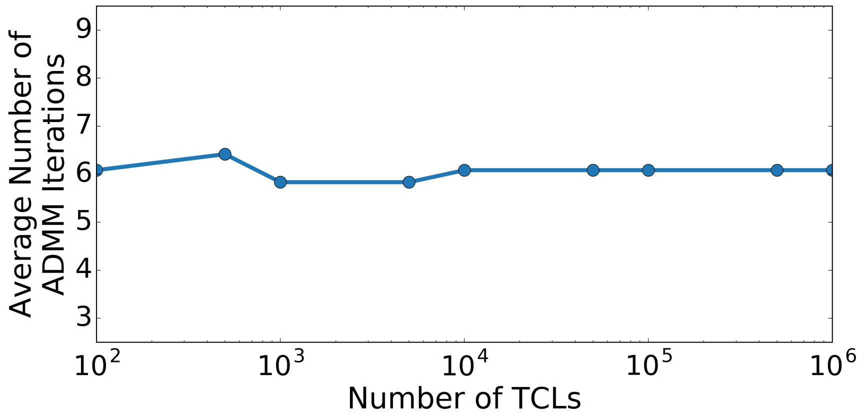

Figure 6.15: Mean number of ADMM iterations for each TCL population over first hour of generation following

In this experiment, we generate populations of 100, 500, 1,000, 5,000, 10,000, 50,000, 100,000, 500,000, and 1,000,000 TCLs and employ each population to follow the first hour (i.e. first 12 time steps) of the signal. Additionally, the ADMM algorithm is stopped once the aggregate power demand of the population is within the error tolerance. This can be viewed as a relaxation of the stopping criteria in (6.12).

The results of this experiment are presented in Figures 6.14 and 6.15. Figure 6.14 shows

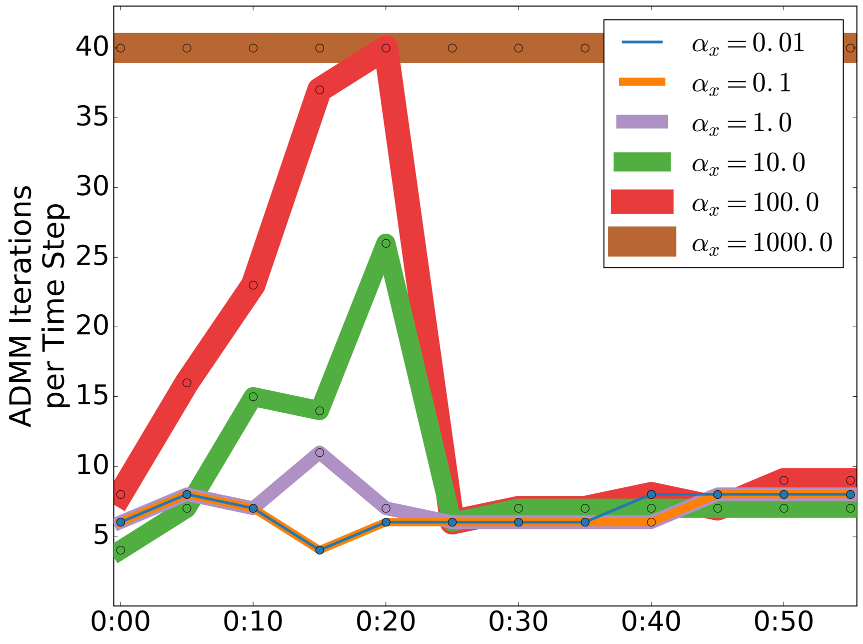

Figure 6.16: ADMM iterations at each time step with varying values of α_x

the number ADMM iterations at each time step for the 100, 1,000, 10,000, 100,000, and 1,000,000 TCL populations. Note that for the 10,000, 100,000, and 1,000,000 TCL populations, the iteration numbers at each time step are equal.

Figure 6.15 shows the mean number of iterations in the the first hour of generation following for each of the nine TCL populations. The results suggest that the number of ADMM iterations is independent of the population size. Therefore, increasing the number of the TCLs in the population does not directly increase the number of ADMM iterations required to perform generation following.

Increasing α_x

The weighting term α_x represents the willingness of a TCL to permit temperature drift away from the setpoint. To test the impact of α_x on the number of ADMM iterations, we have designed an experiment in which TCL populations with varying α_x respond to 1% of the load following signal. Each population is comprised of 10,000 homogeneous refrigerators with identical parameters (the mean of the parameter ranges in Table 6.1) and is employed to follow the first hour (i.e. first 12 time steps) of the signal. We limit the number of ADMM iterations to 40.

The results of this experiment are presented in Figures 6.16, 6.17, and 6.18. Figure 6.16 shows the number ADMM iterations at each time step for the different values of α_x and Figure 6.17 shows the mean number of iterations in the first hour of generation following.

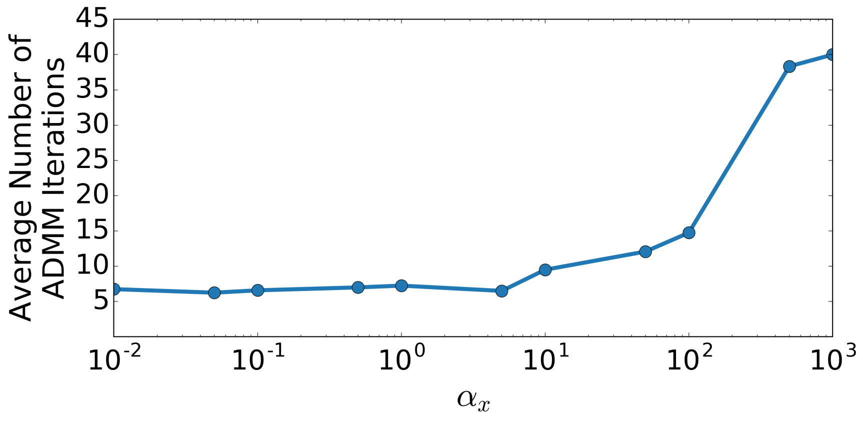

Figure 6.17: Mean number of ADMM iterations for each value of α_x over first hour of generation following

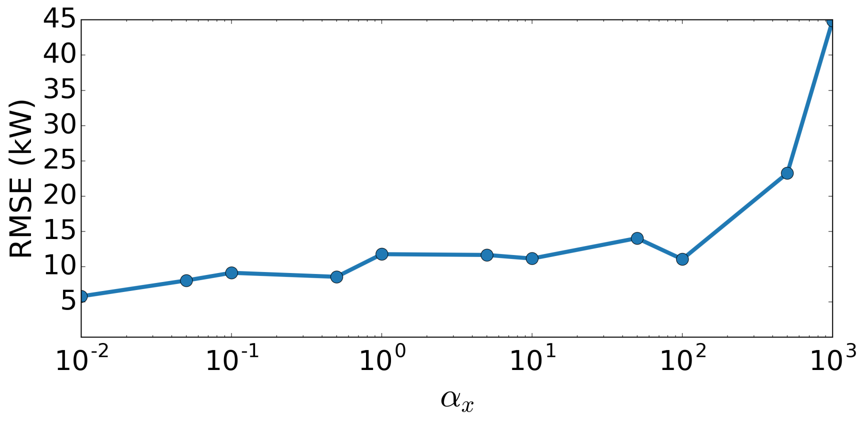

Figure 6.18: RMSE of continuous response for each value of α_x over first hour of generation following

As shown, when α_x is small, fewer ADMM iterations are required to find a solution. This can be attributed to the weighting of the aggregator objective function relative to the objective function of the individual TCLs in the population. In other words, the TCLs seek to minimize the global objective of generation following and allow their temperatures to drift from the setpoint. As α_x increases (and thus the relative weighting of the aggregator objective decreases), the TCLs become less cooperative and more iterations are required to find a solution.

Eventually, α_x increases to a point where the optimal solution of the distributed ADMM algorithm is to minimize the objectives of the individual TCLs rather than the aggregator objective. In other words, the refrigerators in the population choose to minimize the deviation of their internal temperatures from the setpoint rather than participating in the generation following aggregation. As a result, the average number of ADMM iterations increases to the limit of 40, as shown in Figures 6.16 and 6.17, and we observe an increase in the RMSE of the continuous response, as shown in Figure 6.18.

6.6 Conclusions

In this chapter, we presented a formulation of the sharing ADMM algorithm suitable for the distributed optimization of TCLs. The formulation is highly parallelizable and requires the broadcasting of only λ^k and $(\bar{x}^k - \bar{z}^k)$. Given the objective function of every agent is convex, the algorithm is guaranteed to converge to an optimal solution.

Additionally, we have applied the sharing ADMM algorithm with TCL alternative control trajectory representation to the problem of 5-minute ahead renewable energy generation following. Findings of this paper include:

- Using actual wind and solar generation forecasts, ambient temperature records, and published TCL parameters, we have demonstrated how populations of TCLs can be optimized to perform power system services.
- By applying the alternative control trajectory representation to TCLs, we have shown how a population of systems with integer states can be controlled using a convex algorithm.
- By distributing the computation using the sharing ADMM algorithm, we have demonstrated that the generation following algorithm can be scaled to large populations of TCLs without increasing the number of ADMM iterations.
- For highly heterogeneous TCL populations, we have shown that a divide and conquer approach can be employed to minimize the error between the probabilistic solution and the signal.

There are a number of advantages to the distributed TCL control method presented in this manuscript. Firstly, each TCL models its dynamics locally and there is no requirement that TCLs all employ the same model structure or control scheme. Individual TCLs can incorporate higher fidelity or device specific models and still participate in the distributed optimization. Secondly, TCLs can prevent short-cycling. For example, a TCL could exclude any alternative trajectories that violate a minimum dwell time. Thirdly, due to the bi-directional communication, the aggregator can have perfect knowledge of the population's future power demand. There is no need to estimate the power demand if the TCLs are capable of committing to the solution of the optimization algorithm. Quantifying and qualifying the advantages of these characteristics will be the focus of future research.

A challenge not addressed in this chapter is that, because we are not centrally modelling the TCL population, the aggregator does not know the current state or generation following potential of the population. Methods for better understanding and maintaining the generation following potential, which is related to average temperature of each TCL, will be the subject of future work.

Using our sharing ADMM algorithm, we have demonstrated the potential for TCLs to help maintain a continuous and instantaneous balance between generation and load by participating in real-time ancillary service markets. The deployment of such responsive load will be essential for maintaining the stability of power systems with high renewable energy penetration.

Chapter 7

Conclusion

This dissertation contributes to the ongoing development of monitoring and control systems for building energy use. In Chapter 2, we presented an ensemble learning method for the short-term forecasting of total electricity demand in buildings. The method combines the predictions from multiple minimally-customized forecasting models to produce a single prediction. To improve the accuracy of this prediction, we learn, from past data, how the multiple forecasts should be combined. Rather than assuming that demand behaviors are time invariant, the proposed method responds to changes in electricity demand patterns by continuously updating the forecaster's parameters. Experimental results demonstrate enhanced forecasting accuracy across a diversity of buildings.

In Chapter 3, we presented a recursive parameter estimation technique for identifying a thermostatically controlled load (TCL) model that is non-linear in the parameters. To improve the performance of distributed TCL control methods, it is necessary to employ recursive or online parameter estimation algorithms to fit and continuously update each TCL's model. Using the Kalman filter and unscented Kalman filter, we presented four filter methods (single, joint, dual, and triple) for recursively estimating the parameters of a discrete-time TCL model. Additionally, we presented experimental results using real temperature data from a 500W and a 100W residential refrigerator. For each of the four filter methods, the algorithm successfully converges to comparable parameter estimates and adapts to changing TCL characteristics.

In Chapter 4, we developed a piecewise linear thermal model of a residential building that is capable of capturing the predominant dynamics and disturbance patterns. To estimate the model parameters, we presented a Kalman filter based system identification method. Finally, we presented experimental results using real temperature data collected from an apartment with a forced-air heating and ventilation system. These results demonstrate the potential of the model and parameter estimation method to produce accurate forecasts of the air temperature within the apartment.

In Chapter 5, we presented the alternative control trajectory (ACT) representation, a novel method for the approximate optimization of non-convex discrete systems. Energy systems like EVs and TCLs often have binary or discrete states due to hardware limitations and efficiency characteristics. Consequently, non-convex techniques are generally required for optimal control. The ACT representation enables the control of a non-convex energy system to be represented as a convex program. The solution to this program can be interpreted stochastically for implementation. The significant contribution of this approach is that it allows for the approximate optimal control of a population of non-convex agents using distributed convex optimization techniques.

Finally, in Chapter 6, we examined the potential of TCLs, such as refrigerators and electric water heaters, to provide generation following services in real-time energy markets (1 to 5 minutes). Using the ACT representation, we developed a formulation of the sharing ADMM algorithm suitable for the distributed optimization of TCLs. The formulation is highly parallelizable and given the objective function of every agent is convex, the algorithm is guaranteed to converge to an optimal solution. Experimental results demonstrated the potential for TCLs to help maintain a continuous and instantaneous balance between generation and load by participating in real-time ancillary service markets. The deployment of such responsive load will be essential for maintaining the stability of power systems with high renewable energy penetration.

Appendix A

Gated Ensemble Learning Method for Demand-Side Electricity Load Forecasting

This appendix describes research on the development of a gated ensemble learning method for producing short-term building electricity demand forecasts [12].

A.1 Motivation & Background

To improve the accuracy of electricity demand forecasts and aid in the management of power systems, recent attention has been placed on short-term building-level electricity demand forecasting using a wide range of models [92, 41]. The ability to accurately and adaptively forecast demand-side loads will play a critical role in maintaining grid stability and enabling renewables integration. Additionally, many novel optimal control schemes, under research umbrellas such as demand response and microgrid management, require short-term building electricity demand forecasts to aid in decision making [40].

The supply-side and load-side time series forecasting of electricity demand has been a topic of research for many decades. The literature is filled with a variety of well-cited modelling approaches, each differing in algorithmic complexity, estimation procedure, and computational cost. Of particular note are the variants of Artificial Neural Networks (ANN) [3, 92, 41, 7, 33, 43, 68], Support Vector Regression (SVR) [62, 76, 44, 35] and Autoregressive Integrated Moving Average (ARIMA) models [3, 76, 44, 108, 21, 88, 71]. Lesser but nonetheless noteworthy attention has been given to approaches such as Multiple Linear Regression [3, 62, 81], Fuzzy Logic [3, 52], Decision Trees [92], and k-Nearest Neighbors (k-NN).

These studies provide a broad catalog of use-cases and demonstrate the performance of certain forecasting algorithms when applied to specific building types. In particular, [3, 92, 68, 88] provide a survey of electricity forecasting methods and a high-level comparison of techniques. [33] provides a detailed description of ANNs and their application to load forecasting, including data pre-processing and ANN architectures. [41] details the development of a seasonal ANN approach and the advantage over a Seasonal ARIMA (SARIMA) model when applied to 6 building datasets. [71] focuses on the introduction of motion sensor data to improve the accuracy of an ARIMA model. In [43, 62, 108, 71, 52], the authors perform an in-depth analysis of the power demand patterns of a particular building in order to customize a forecasting model.

In papers with experimental results, the authors have generally applied their electricity demand forecasting technique to only a small number of datasets. Consequently, the literature is rich with forecasting algorithms customized for individual buildings. This leads us to the following question: Is it possible to design a single minimally-customized forecasting algorithm that is widely applicable across a diversity of building types, enabling scalability? We pursue this question by proposing a novel ensemble learning method for electricity demand forecasting.

Specifically, due to unique building characteristics, occupancy patterns, and individual energy use behaviors, we argue that no single model structure is capable of accurately forecasting electricity demand across all commercial and residential buildings.

For example, some forecasting models may produce accurate predictions under certain observable or unobservable conditions, such as a seasonal trend, a morning routine, or an extended absence. Other models may be ideal for buildings with energy use behaviors that are stable over long periods of time. For buildings with frequent changes in occupancy patterns, models that are trained over a moving horizon may yield the highest accuracy. In short, this work will develop an ensemble learning method that trains and validates multiple forecasting models before applying a gating method to select a single model to perform electricity demand forecasting.

In this way, the ensemble method is able to learn from real-time data and to produce short-term electricity demand forecasts that are automatically tailored to a particular building and instance in time. In addition to forecast accuracy, this appendix will place an emphasis on method adaptability and ease of use. While we have implemented certain forecasting models, the method is intended to allow the models to be interchangeable.

To demonstrate the use of our ensemble method to produce short-term forecasts, this appendix includes 3 experimental studies: Single Model Studies, Multiple Model Study, and Residential Study. For each of these studies, we will make the following assumptions with respect to the availability of building electricity demand data:

A1. We have access to hourly historical building electricity demand at the meter.

A2. We have access to hourly historical weather data near the building location.

A3. We do not have access to submetered electricity demand data or building operations data, such as occupancy measurements or mechanical system schedules.

The limited access to input data with which to produce forecasts is representative of the challenge faced by grid operators. Accordingly, this appendix will demonstrate the potential of our ensemble method to non-invasively forecast total electricity demand using data-driven methods. Additionally, unlike in [43, 62, 108, 71, 52], where the authors perform an in-depth analysis of the power demand patterns in order to customize a model to a particular building, this appendix will focus on developing a forecasting approach that is generally applicable to all buildings without customization.

This appendix is organized into five sections: Regression Models, Single Model Studies, Ensemble Method, Multiple Model Study, and Residential Study. Section II. Regression Models briefly presents background theory for 5 regression models that will be employed in this appendix. In Section III. Single Model Studies, we apply the forecasting models to 8 commercial/university building electricity demand datasets using batch and moving horizon training approaches. Section IV. Ensemble Method presents our method for training and validating multiple models and for selecting the optimal model using a gating method. Section V. Multiple Model Study applies our ensemble learning method to 8 commercial/university building electricity demand datasets and quantifies and qualifies the advantage over a single model approach. Finally, in Section VI. Residential Study, we apply our ensemble learning method to 24 residential building electricity demand datasets and summarize the results. Key conclusions and future research directions are summarized in Section VII.

A.2 Regression Models

In this appendix, we will consider one parametric regression model, Ordinary (Linear) Least Squares with ℓ_2 Regularization (Ridge), and four nonparametric models, Support Vector Regression with Radial Basis Function (SVR), Decision Tree Regression (DTree), k-Nearest Neighbors with uniform weights and binary tree data structure (k-NN), and Multilayer Perceptron (MLP), a popular type of feedforward Artificial Neural Network (ANN). In this section, we will briefly describe the structure of each regression model.

Ordinary Least Squares with ℓ_2 Regularization

Ordinary Least Squares with ℓ_2 Regularization (Ridge) fits a linear model with coefficients $w \in \mathbf{R}^n$ to minimize the residual sum of squared errors between the observed and predicted responses while imposing a penalty on the size of coefficients according to their ℓ_2-norm. The linear model of a system with univariate output is given by

$$\begin{aligned} \hat{y} &= w_0 x_0 + w_1 x_1 + \ldots + w_n x_n \\ &= \sum_k w_k x_k = w^T x \end{aligned} \tag{A.1}$$

with variables $x \in \mathbf{R}^n$, the model input, $\hat{y} \in \mathbf{R}$, the predicted response, n, the number of inputs or features in x, and $k = 1, \ldots, n$.

The linear model is trained on a set of inputs and observed responses by optimizing the function

$$\underset{w}{\text{minimize}} \sum_i \|w^T x_i - y_i\|_2^2 + \lambda \|w\|_2^2 \tag{A.2}$$

with variables $x_i \in \mathbf{R}^n$, the model input for the i-th data point, $y_i \in \mathbf{R}$, the i-th observed response, $w \in \mathbf{R}^n$, the weighting coefficients, and $i = 1, \ldots, N$, where N is the number of data samples and n is the number of features in x_i. Lastly, λ is a weighting term for the regularization penalty.

For a system with a multivariate output $\hat{y} \in \mathbf{R}^m$, we will treat the outputs as uncorrelated and define a set of coefficients $w_j \in \mathbf{R}^n$ for each predicted response $\hat{y}_j \in \mathbf{R}$ for $j = 1, \ldots, m$. Thus, the multivariate linear model is given by

$$\hat{y}_j = w_j^T x, \ \forall j = 1, \ldots, m \tag{A.3}$$

The weights of the multivariate model are determined by optimizing the function:

$$\underset{w}{\text{minimize}} \sum_i \sum_j \|w_j^T x_i - y_{i,j}\|_2^2 + \sum_j \lambda \|w_j\|_2^2 \tag{A.4}$$

with variables $x_i \in \mathbf{R}^n$, the model input, $y_i \in \mathbf{R}^m$, the observed multivariate response, $w_j \in \mathbf{R}^n$, the weighting coefficients of the j-th response, $i = 1, \ldots, N$, and $j = 1, \ldots, m$, where N is the number of data samples, n is the number of features in x_i, and m is the number of observations in y_i.

Support Vector Regression

In non-linear Support Vector Regression (SVR), the model input x is transformed into a higher dimensional feature space using a mapping function ϕ. Then, a linear model is constructed in this feature space, as given by

$$\hat{y} = f(x) = w^T \phi(x) + b \tag{A.5}$$

where variable $b \in \mathbf{R}$ is a bias term, $w \in \mathbf{R}^n$ the linear coefficients, and ϕ a non-linear mapping function. Using an ϵ-intensive loss function, the support vector regression model can be trained by optimizing

$$\begin{aligned} \underset{w,b,\zeta,\zeta*}{\text{minimize}} \ & \frac{1}{2}\|w\|_2^2 + C \sum_i (\zeta_i + \zeta_i^*) \\ \text{subject to} \ & y_i - w^T \phi(x_i) - b \leq \epsilon + \zeta_i \\ & w^T \phi(x_i) + b - y_i \leq \epsilon + \zeta_i^* \\ & \zeta_i, \zeta_i^* \geq 0 \end{aligned} \tag{A.6}$$

where constant $\epsilon \in \mathbf{R}$ denotes the radius of the ϵ-intensive region (i.e. region in which the value of the loss function is 0) and constant $C > 0$ denotes the trade-off between the empirical risk (i.e. deviations beyond ϵ) and the regularization term (i.e. the flatness of the model). The positive slack variables $\zeta_i \in \mathbf{R}$ and $\zeta_i^* \in \mathbf{R}$ denote the magnitude of the deviation from the ϵ-intensive region.

By applying Lagrangian multipliers and Karush-Kuhn-Tucker conditions, the primal problem can be reformulated into a dual form that is easier to solve.

$$\begin{aligned}\underset{\alpha,\alpha*}{\text{minimize}} \; & \frac{1}{2}\sum_i\sum_j(\alpha_i-\alpha_i^*)^T K(x_i,x_j)(\alpha_i-\alpha_j^*) \\ & +\epsilon\sum_i(\alpha_i+\alpha_i^*)-\sum_i y_i(\alpha_i-\alpha_i^*) \\ \text{subject to} \; & \sum_i(\alpha_i-\alpha_i^*)=0 \\ & 0\le\alpha_i,\alpha_i^*\le C\end{aligned} \tag{A.7}$$

with variables $\alpha_i, \alpha_i^* \in [0, C]$, the Lagrangian multipliers, and K, the kernel function given by the inner product of the mapping functions, $K(x_i, x_j) = \phi(x_i)\phi(x_j)$. In this appendix, we will employ the (Gaussian) Radial Basis Function kernel, given by

$$K(x_i,x)=\exp\left(-\frac{\|x_i-x\|_2^2}{2\sigma^2}\right) \tag{A.8}$$

where variable $\sigma \in \mathbf{R}$ is a free parameter.

Using the Lagrangian multipliers, the support vector regression model with univariate output can be rewritten as

$$\hat{y}=f(x)=\sum_i(\alpha_i-\alpha_i^*)K(x_i,x)+b \tag{A.9}$$

For a system with a multivariate output $\hat{y} \in \mathbf{R}^m$, we will treat the outputs as uncorrelated and define m support vector regression models (i.e. one for each output). Using m models to independently predict each of the m outputs is simple to implement, but may be less accurate than a single model capable of simultaneously predicting all m outputs. We refer the reader to [85, 55] for a more detailed description of the support vector regression algorithm.

Decision Tree Regression

In Decision Tree Regression (DTree), the feature space is recursively sub-divided or partitioned into smaller regions or leaves (i.e. terminal node). Once this splitting is complete, a simple regression model is fit to the data samples that have been grouped into each leaf. The objective is to form a tree data structure with which a new observation can be assigned to a leaf using nested boolean logic (i.e. moving right or left down each branch according to a threshold value). Once the correct leaf has been identified, the observation can be mapped to a continuous target value using a simple model.

Expressed mathematically, we can represent the data at node a by B. For each potential division $\theta = (k, t_a)$ composed of feature k and threshold value $t_a \in \mathbf{R}$, we may partition the data into two subsets or branches, $B_L(\theta)$ and $B_R(\theta)$, given by

$$\begin{aligned} B_L(\theta) &= (x, y) \text{ if } x_k \leq t_a \\ B_R(\theta) &= (x, y) \text{ if } x_k > t_a \end{aligned} \tag{A.10}$$

The accuracy of a decision tree regression model can be represented by the impurity of each branch. In this appendix, we will define the impurity function H as the mean squared error between each response and the mean response in a branch. Thus, for node a, H is given by

$$\begin{aligned} \bar{y}_a &= \frac{1}{N_a} \sum_{i=1}^{N_a} y_{a,i} \\ H(X_a) &= \frac{1}{N_a} \sum_{i=1}^{N_a} (y_{a,i} - \bar{y}_a)^2 \end{aligned} \tag{A.11}$$

with variable $N_a \in [N_{min}, N]$, the number of data points in branch a, N_{min}, the minimum number of data points in a branch, X_a, the set of all data points $x \in \mathbf{R}^n$ in branch a, and y_a, the set of all observed responses $y \in \mathbf{R}$ in branch a.

The decision tree is trained or grown by recursively selecting the parameters that minimize the impurity of the tree and of each branch. In other words, we begin with a node a containing all data points. Then, we partition the data according to the optimization function

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{N_L}{N_a} H(B_L(\theta)) + \frac{N_R}{N_a} H(B_R(\theta)) \tag{A.12}$$

with variable $N_L, N_R \in [N_{min}, N_a]$, the number of data points in the left and right branches, respectively. The optimization function is recursively applied to each new branch until the maximum tree depth is reached, one of the resulting branches would contain less than N_{min} data points, or there is no split that will decrease the impurity of the branch by more than some threshold $\delta \in \mathbf{R}$.

For each leaf a, we will define the regression model as the mean of the contained univariate observations.

$$\hat{y} = \frac{1}{N_a} \sum_{i=1}^{N_a} y_{a,i} \tag{A.13}$$

For a system with multivariate output $\hat{y} \in \mathbf{R}^m$, each leaf in the tree will store observations of length m rather than 1. Therefore, the impurity function H can be redefined as the mean of the mean squared error between each j-th response and the mean j-th response in a branch for $j = 1, \ldots, m$.

$$\begin{aligned} \bar{y}_{a,j} &= \frac{1}{N_a} \sum_{i=1}^{N_a} y_{a,i,j} \ \forall j = 1, \ldots, m \\ H(X_a) &= \frac{1}{mN_a} \sum_{j=1}^{m} \sum_{i=1}^{N_a} (y_{a,i,j} - \bar{y}_{a,j})^2 \end{aligned} \tag{A.14}$$

And the multivariate regression model can be defined as

$$\hat{y}_j = \frac{1}{N_a}\sum_{i=1}^{N_a} y_{a,i,j} \ \forall j = 1, \ldots, m \tag{A.15}$$

We refer the reader to [23, 92] for a more detailed description of the decision tree regression algorithm.

k-Nearest Neighbors Regression

In k-Nearest Neighbors Regression (k-NN), an input $x \in \mathbf{R}^n$ is mapped to a continuous output value according to the weighted mean of the k nearest data points or neighbors, as defined by the Euclidean distance. In this appendix, we will use uniform weights. In other words, each point in a neighborhood a contributes uniformly and thus the predicted univariate response $\hat{y} \in \mathbf{R}$ is the mean of the k-nearest neighbors.

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_{a,i} \tag{A.16}$$

with variable y_a, the set of k observed responses $y \in \mathbf{R}$ in neighborhood a. For a system with multivariate output $\hat{y} \in \mathbf{R}^m$, the model is defined as the mean of each observation j over the k-nearest neighbors.

$$\hat{y}_j = \frac{1}{k}\sum_{i=1}^{k} y_{a,i,j} \ \forall j = 1, \ldots, m \tag{A.17}$$

Given a new input x, it is possible to determine the neighborhood by computing the Euclidean distance (i.e. ℓ_2-norm of the difference) between the new input x and every data point in the training data set x_i for $i = 1, \ldots, N$ and then ordering the distances to identify the nearest neighbors. However, this brute-force search is computationally inefficient for large datasets.

To improve the efficiency of the neighborhood identification, the training data points are partitioned into a tree data structure. A commonly used approach for organizing points in a multi-dimensional space is the ball tree data structure, a binary tree in which every node defines a D-dimensional hypersphere or ball. At each node, data points are assigned to the left or right balls according to their distance from the ball's center. At each terminal node or leaf, the data points are enumerated inside the ball.

We refer the reader to [73] for a description of ball tree construction algorithms.

Multilayer Perceptron Artificial Neural Network

Artificial Neural Networks (ANN) are a class of statistical learning algorithms inspired by the neurophysiology of the human brain. These numerical models are composed of intercon-

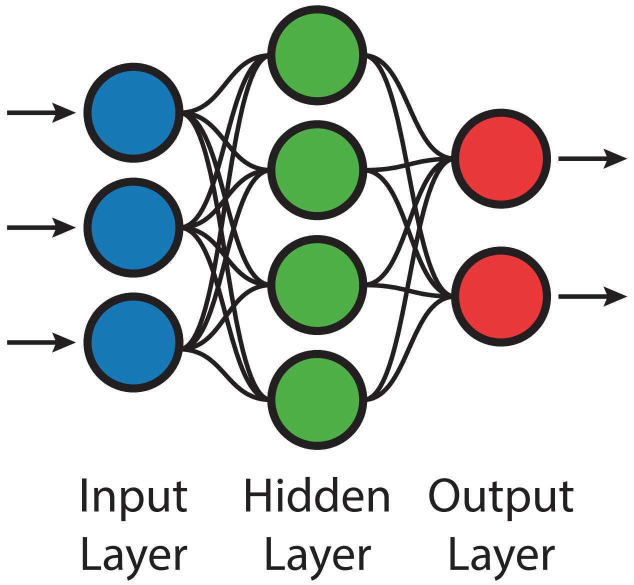

Figure A.1: Artificial Neural Network

nected "neurons" which use stimulation thresholds to predict how a system will respond to inputs.

The most popular feedforward (i.e no feedback) neural network is the multilayer perceptron (MLP). Figure A.1 illustrates an example of a 3 layer perceptron network consisting of a 3 neuron input layer, 4 neuron hidden layer, and 2 neuron output layer. The structure of a neuron in the hidden layer is presented in Figure A.2 where variables $x_1, x_2, x_3 \in \mathbf{R}$ are inputs to the neuron and $w_1, w_2, w_3 \in \mathbf{R}$ are synaptic weights. Variable $y \in \{0, 1\}$ is the output and $z_1, z_2 \in \mathbf{R}$ are the synaptic weights of the next neurons in the network. The weighted sum of the inputs is the excitation level of the neuron

$$v = \sum_{i=1}^{n} w_i x_i - h \tag{A.18}$$

where variable $v \in \mathbf{R}$ is the excitation, $h \in \mathbf{R}$ is a threshold, and n is the number of inputs to the neuron. Next, we want to define the output or activation function f of the neuron such that if $v \geq 0$ then $y = 1$ otherwise $y = 0$. The simplest activation function is the hard limiter,

$$y = f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \tag{A.19}$$

However, the hard limiter function cannot be practically implemented because it is not differentiable. Thus, artificial neural network algorithms employ differentiable activation

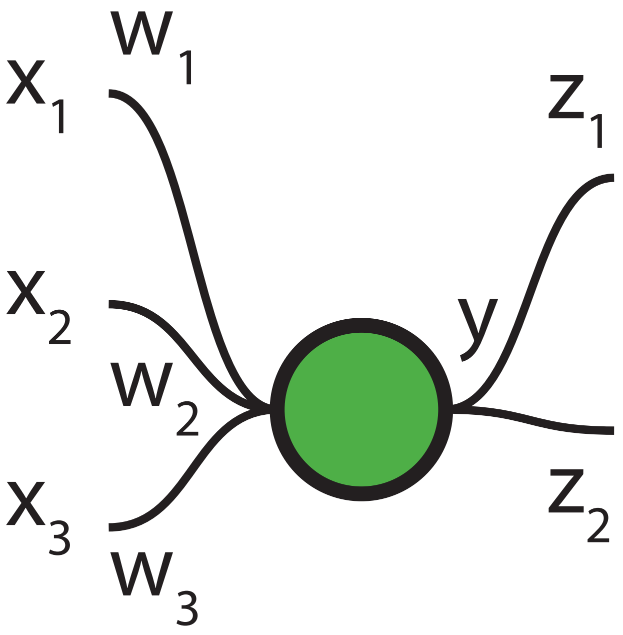

Figure A.2: Structure of a Neuron

functions that have horizontal asymptotes at both 0 and 1 (i.e. $\lim_{v\to\infty} f(v) = 1$ and $\lim_{v\to-\infty} f(v) = 0$). An example is the sigmoid function,

$$y = f(v) = \frac{1}{1+e^{-v}} \tag{A.20}$$

For the neurons in the output layer, it is common to use a linear activation function,

$$\hat{y} = f(v) = v \tag{A.21}$$

The structure of the multilayer perceptron artificial neural network makes it capable of both univariate and multivariate predictions. Networks are trained using a backpropogation algorithm which adjusts the weights and thresholds of each neuron to minimize the error between the observations and the network outputs. In this appendix, we will employ a 3 layer feedforward ANN with a 30 neuron sigmoid hidden layer and 6 neuron linear output layer. The size of the linear input layer will vary. The ANN will be trained using gradient descent backpropagation for 200 epochs.

We refer the reader to [22] for a further discussion of artificial neural networks and backpropogation training algorithms.

Software Packages

This work employs the PyBrain Artificial Neural Network library [83] and the Sci-Kit Learn Ordinary Least Squares, Support Vector Regression, Decision Tree, and k-Nearest Neighbors libraries [77]. All plots are generated with Matplotlib [36].

A.3 Single Model Studies

To motivate the advantage of our ensemble approach, we will begin by considering single model approaches to electricity demand forecasting. In this section, we will apply the regression models above to 8 building datasets containing 2 years of metered hourly electricity demand (kW). This time-series data has been provided by the facilities team at the University of California, Berkeley and will be used as the observation data for each forecasting model. Submetered electricity demand data and building operations data, such as occupancy measurements and mechanical system schedules, are not available.

The 8 buildings are located on the University of California, Berkeley campus and have been selected to represent a heterogeneous population. The buildings A, B, and D are occupied by the physics, civil engineering, and environmental engineering departments, respectively, and are primarily comprised of faculty offices and research laboratories. Building C is a university library and building E houses faculty and departmental offices for multiple humanities departments. Buildings F and G are university administrative buildings and building H is comprised mainly of lecture halls and classrooms. Table A.1 presents the square footage as well as basic statistics regarding the electricity demand of each building.

	Building							
	A	B	C	D	E	F	G	H
Size (10^3 sq.ft.)	97	140	67	142	306	111	153	140
Mean (kW)	333	33	96	109	113	114	23	73
SD (kW)	44	5	26	8	36	33	2	30
Min (kW)	190	20	48	61	60	69	5	29
Max (kW)	602	69	221	150	271	236	73	195

Table A.1: Building Electricity Demand Statistics

The models will be used to generate short-term multivariate electricity demand forecasts, specifically 6 consecutive hourly electricity demand predictions (i.e. $\hat{y} \in \mathbf{R}^6$). The accuracy of each forecast $\hat{y}$ will be measured by the root mean squared error (RMSE). To allow for comparison between buildings, the performance of forecasting models will be measured by the mean absolute percent error (MAPE).

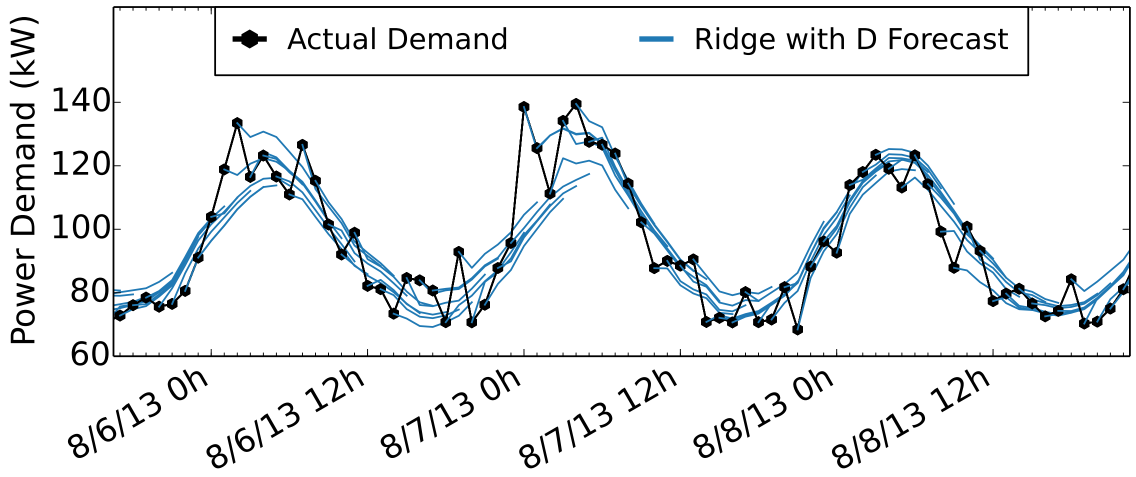

Figure A.3: Building E Ridge Forecast Sample

$$\text{Forecast } i \text{ RMSE} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(y_{i,j} - \hat{y}_{i,j})^2} \tag{A.22}$$

$$\text{Model MAPE} = \frac{100\%}{mN}\sum_{i=1}^{N}\sum_{j=1}^{m}\left|\frac{y_{i,j} - \hat{y}_{i,j}}{y_{i,j}}\right| \tag{A.23}$$

with variables $y_i \in \mathbf{R}^m$, the i-th observation, and $\hat{y}_i \in \mathbf{R}^m$, the i-th prediction, where m represents the number of outputs in the prediction ($m = 6$) and N, the number of predictions.

The regression models will employ 4 different input types: electricity demand (D), time (T), electricity demand and time (DT), and electricity demand, time, and exogenous weather data (DTE). The electricity demand input type (D) consists of the 24 hourly records that precede the desired forecast ($x \in \mathbf{R}^{24}$). The time input type (T) is the current weekday and hour represented as a sparse binary vector ($x \in \{0, 1\}^{31}$). The demand and time input type (DT) combines the demand and time inputs ($x \in \mathbf{R}^{55}$). The demand, time, and exogenous weather data input type (DTE) is the demand and time input with current outdoor air temperature °C) and relative humidity (%RH) data retrieved from a local weather station ($x \in \mathbf{R}^{57}$)[86]. The output of each forecasting model is a prediction of the hourly electricity demand over the following 6 hours ($y \in \mathbf{R}^6$).

Throughout this appendix, we will use the term “regression model” to refer to the model structure and algorithm used to perform regression, “input type” to refer to the subset of features used by each model, and “forecasting model” to refer to the pairing of regression model and input type.

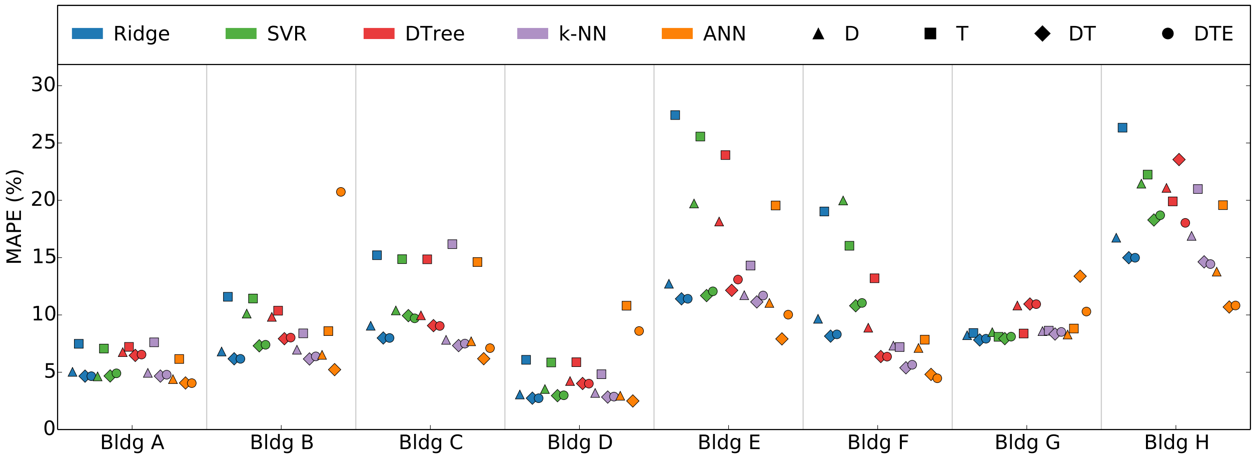

Figure A.4: Batch Study Results

Batch Study

For each of the 8 buildings, we train the forecasting models with demand data from that building. Electricity demand data from one building is not used to fit the models of another building. We consider 5 regression models (Ridge, SVR, DTree, k-NN, and ANN) and 4 input types (D, T, DT, and DTE) for a total of 20 forecasting models per building. The forecasting models for each building are trained in a batch manner (i.e. trained once on a large dataset) using 18 months of hourly input data from January 1st, 2012, to July 1st, 2013 (i.e. 13,128 training data points). For each model, the training dataset depends on the input type (D, T, DT, and DTE) but may include hourly electricity demand records for the respective building (D), time records represented as a sparse binary vector (T), and hourly outdoor air temperature and relative humidity records (E).

For validation, the trained models are used to generate a 6 hour electricity demand forecast ($\hat{y} \in \mathbf{R}^6$) for each building for every hour from July 1st, 2013, to January 1st, 2014 (i.e. 4,416 testing data points). The results of the batch regression study are presented in Figure A.4. In the figure, each data point represents the MAPE of one forecasting model (i.e. 8 buildings with 20 models each for a total of 160 models).

Examples of the electricity demand forecasts produced by the Building E Ridge regression model with demand (D) input are presented in Figure A.3. Note that each blue line represents a multivariate forecast $\hat{y}$ and that the figure plots $\hat{y}$ starting at but excluding the most recent power demand record.

By comparing the results in Figure A.4 for each building, we can immediately distinguish forecasters that consistently perform poorly (e.g. T input) from forecasters that perform well (e.g. ANN, Ridge, and k-NN with DT input). We also observe that certain forecasting models perform inconsistently across the different buildings. For example, SVR with D input performs well in Building A but relatively poorly in Buildings E and F. Furthermore, there is dispersion among the results, particularly in Buildings E, F, and H. This dispersion represents a challenge for building level applications. To produce the best results using a batch approach, an engineer must perform model selection for every deployment. Just because a certain regression model and input type has performed well for one building does not guarantee it will do the same for another building.

As shown, an ANN model outperforms the Ridge, SVR, DTree, and k-NN models in 7 out of 8 building. However, there is no input type that performs best in each building. In Building B, we observe that the addition of the exogenous weather input decreases the performance of the ANN. A possible explanation is that the ANN found a link between the weather input and the electricity demand in the training data. However, this link may not have continued to appear in the test data, resulting in a loss in performance.

Additionally, it could be argued that the ANN model does not outperform the much simpler Ridge and k-NN models to an extent that warrants the additional complexity, particularly in Buildings D and G. If further tuned to a particular building and input type and trained over a higher number of epochs, it is possible that the ANNs' performances could be further improved. However, this increase in forecaster accuracy would come at a high computational cost. Instead, this work will focus on developing a computationally efficient ensemble method for producing electricity demand forecasts by learning from data streams and adapting to individual building energy use patterns.

Moving Horizon Study

In the batch regression study presented above, each forecasting model was trained once on 18 months of data and then used to generate predictions up to 6 months after the last training data point. For buildings with very consistent electricity demand patterns, using such a large training set will help to prevent overfitting and to produce the best possible results. For buildings with inconsistent or changing electricity demand patterns, the absence of the most recent data from the training set may limit the performance of the forecasting model.

In this section, we will consider training the models over a moving horizon. Specifically, at each hour between July 1st, 2013, and January 1st, 2014, we will train the forecasting models for each building on the 3 months of data that precede that time step (i.e. 2,016 training data points). Then, we will produce a 6 hour forecast ($\hat{y} \in \mathbf{R}^6$) and record the errors. In this way, the start and end times of the training set will move relative to the current time step and we can hope to better capture recent electricity demand patterns. It should be noted that because we have significantly reduced the training set size, we have introduced the potential for overfitting the models, a point that will be addressed by our ensemble method. We will consider 4 regression models (Ridge, SVR, DTree, and k-NN) and 4 input types (D, T, DT, and DTE) for a total of 16 forecasting models per building. The computational cost of training an ANN makes the model unsuitable for a moving horizon approach.

The results of the moving horizon regression study are presented in Figure A.5. In the figure, each data point represents the MAPE of one forecasting model (i.e. 8 buildings with 16 models each for a total of 128 models). The horizontal dotted line marks, for each

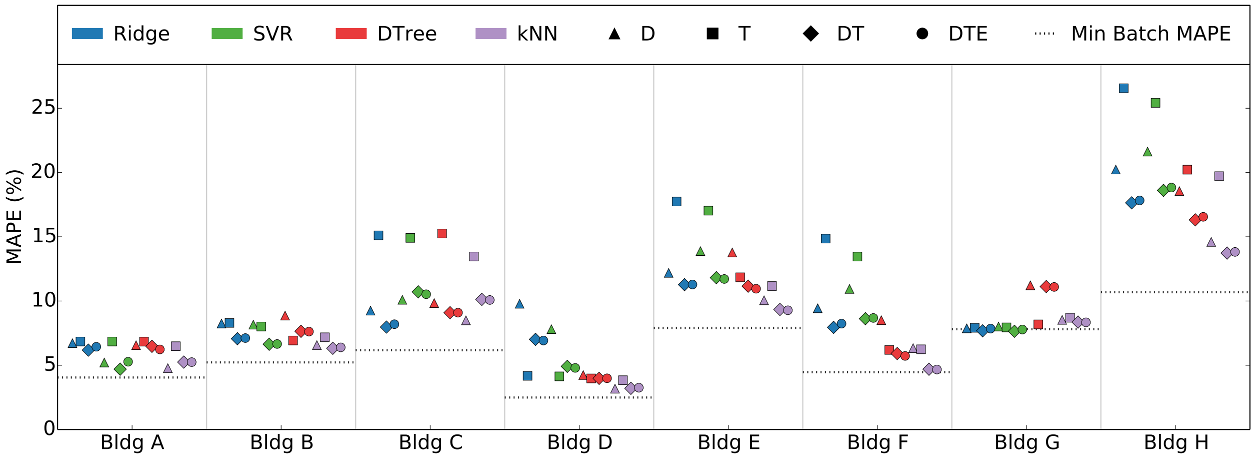

Figure A.5: Moving Horizon Study Results

building, the highest performing forecasting model (lowest MAPE) from the batch regression study. In other words, for Buildings B, C, D, E, and H, the dotted line indicates the MAPE of the ANN model with DT input. For Buildings A and F, the line marks the MAPE of the ANN model with DTE input. For Building G, Ridge with DT produced the lowest MAPE in the batch study.

While none of the models trained on a moving horizon show a significant improvement in accuracy over the best batch model, the results for each individual building are generally less dispersed in the moving horizon cases than in the batch cases.

Because we have generated 16 forecasts for every hour between July 1st, 2013, and January 1st, 2014, we are able to compare the performance of the forecasting models at each time step and across all 8 buildings. This analysis will aid in determining which models to include in the ensemble method. Figure A.6 shows the fraction of time steps that a specific regression model produced the most accurate (lowest RMSE) electricity demand prediction (regardless of input type). Here, we see that k-Nearest Neighbors models produce the best forecast with the highest frequency of any of the regression models considered. Because we plan to incorporate several forecasters in the ensemble method, we are also interested in identifying models that consistently rank among the top. Figure A.7 shows the fraction of time steps that a specific regression model produced a forecast that was among the best four predictions. Here, we see that Ridge models most consistently produced such a prediction. Repeating this analysis for input types (not displayed), we find that every input scores between 20% and 30% for both the top and top four predictions, suggesting no clear relative advantage.

Recognizing that a specific forecasting model may have the highest accuracy in one time step and the lowest accuracy in the next, we are also interested in which regression models and inputs show the poorest performance (highest RMSE). For the model that generates the worst forecast at each time step (not displayed), Decision Tree Regression ranks poorest

(44%) followed by Ridge (31%). For the worst four predictions, Decision Tree Regression ranks poorest again (42%) followed again by Ridge (26%). Figures A.8 and A.9 show the fraction of time steps that a specific input type produced the least accurate electricity demand prediction. Not surprisingly, the demand only input (D) ranks worst in both cases. For the buildings studied, exogenous weather inputs do not appear to significantly improve the forecast accuracy. Buildings located in less temperate climates could be expected to show greater correlation between temperature and electricity demand.

Single Model Study Conclusions

Based on the results from the batch and moving horizon studies, we assert that no single forecasting model (regression model and input type) will produce the best results across every building. To produce good results using a single model approach, an engineer must perform model and feature selection for each deployment, increasing the cost of energy management applications that require building-level electricity demand forecasts.

However, the results also show that there is a subset of forecasting models that perform well for each building. Furthermore, at each time step, one forecasting model produces a prediction that is more accurate than any other prediction. If we train a set of regression models and determine which model in the set is most likely to produce the best prediction at a given time step, we can formulate an ensemble method that is able to perform model and feature selection by learning from past electricity demand. This is the core motivation of the gated ensemble learning method presented in the next section.

A.4 Ensemble Method

Background

Given the many unpredictable behaviors of occupants and the unique physical and mechanical characteristics of every building, a single model approach to electricity demand forecasting may perform very well in one case and very poorly in another. Without being able to observe the causes of electricity demand changes (through extensive sub-metering and/or occupancy sensing), it is difficult to justify why a model does or does not perform well. Furthermore, the incorporation of exogenous signals like regional weather conditions may improve a model's accuracy but such benefits cannot be guaranteed. Only through observation and experimentation can the best regression model and input type be identified for a particular building.

The assertion that the best forecasting model can be identified through data driven experimentation underlies the ensemble method presented in this appendix. To build upon existing literature and to improve the portability of electricity demand forecasters, we have developed a method that tests multiple models before selecting one that is best suited for a particular building and instance in time.

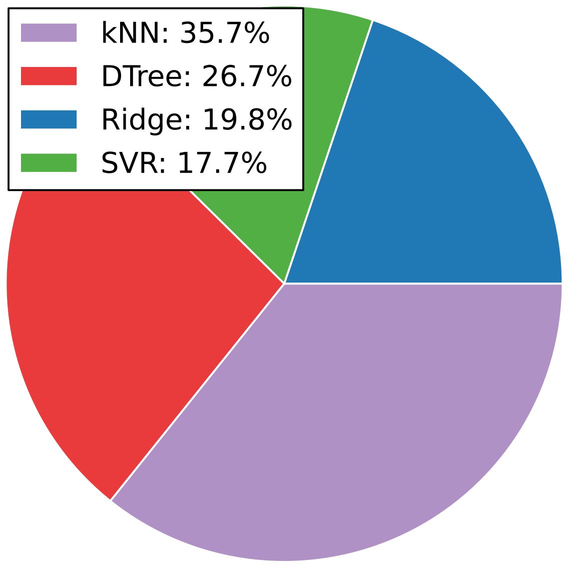

Figure A.6: Best Prediction

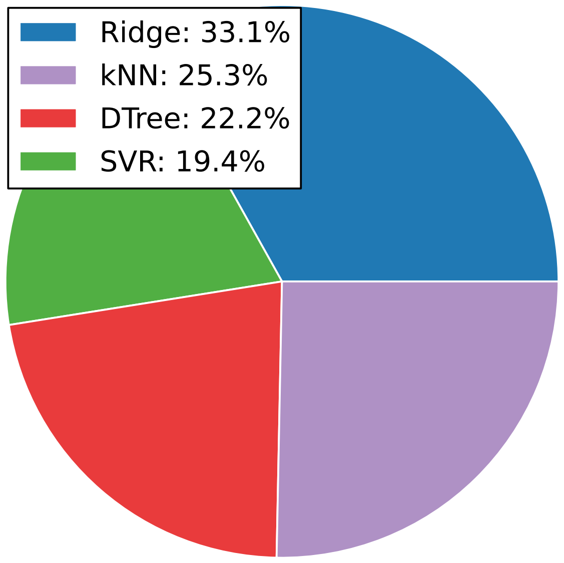

Figure A.7: Best Four Predictions

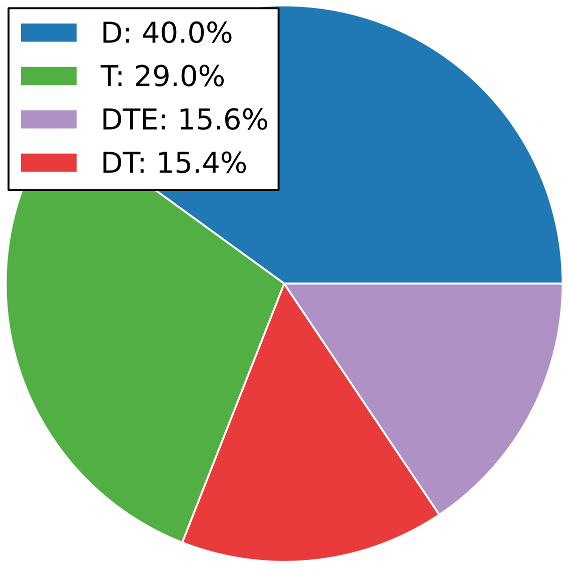

Figure A.8: Worst Prediction

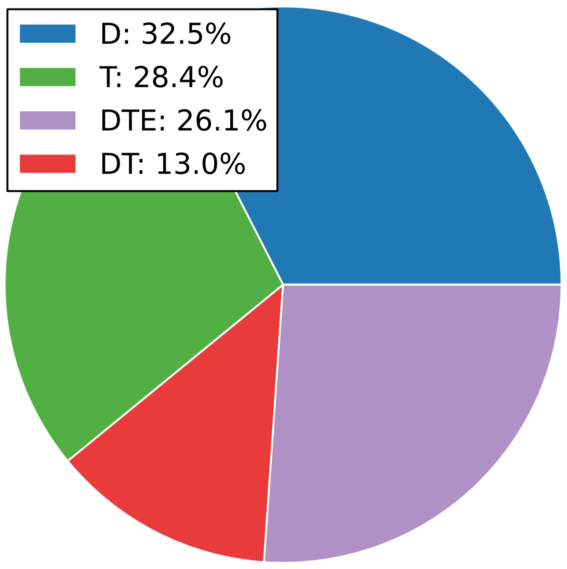

Figure A.9: Worst Four Predictions

Our multiple model regression method falls under a category of ensemble learning methods commonly referred to as a "bucket of models" [2, 24]. It is important to recognize that unlike other ensemble methods (which average, stack, or otherwise combine the outputs from multiple models), the bucket of models approach selects a single model from the ensemble set. Consequently, a bucket of models approach can perform no better than the best model in the set. Therefore, to produce accurate forecasts, it is important that the individual models perform well, though perhaps only in certain conditions. Additionally, our ensemble method must be able to identify and avoid models that are likely to perform poorly for a particular building. This capability will alleviate the need for prior knowledge of a building's energy use and, in practice, allow for model and feature selection to be performed in real time.

Method

Our gated ensemble learning method can be divided into 4 steps: Training, Validation, Gating, and Testing. In the training step, each model is trained on a subset of historic data, with the most recent electricity demand data points reserved for validation, as shown in Figure A.10. The size of the training subset may vary by application and by training approach. For models trained in a batch manner, a large data set (e.g. >12 months) is required. Additionally, the training step is either performed only once or periodically (e.g. every 6 months) rather than at every time step.

For models trained in a moving horizon manner, a smaller dataset is required (e.g. 2 to 12 months) and the training step is performed periodically (e.g. daily) or at every time step. Again, the length of the dataset can be customized to the application. For example, because the use patterns of university buildings change according to an academic calendar, a training set size of 2 or 3 months may be ideal. For an office building with very consistent energy use patterns, a training set size of 6 months or more may produce the best results. Stated more explicitly, small training sets are more capable of capturing recent energy use patterns but carry the risk of allowing the regression model to overfit the data. By contrast, a large dataset will capture consistent energy use patterns but may miss recent or short-term changes, thereby appearing to underfit the data. In this appendix, we will favor smaller training sets (3 months) for moving horizon models and larger trainings sets (18 months) for batch models.

In the validation step, the most recent historic data is used to generate predictions with each forecasting model, as shown in Figure A.10. These forecasts are compared with the electricity demand data (that was not used for model training) to determine each model's performance. Again, the size of the data set used for validation may vary by application, but in our implementation, the length of the validation set is equal to twice the desired forecast length (i.e. for a 6 hour forecast, the previous 12 hourly electricity demand data points are reserved for validation).

The measure of a model's performance or the criteria for identifying the "best" model will depend on the application. In some cases, we may want to define "best" as the forecast that will produce the highest pay-off or incur the least risk. In other cases, we may want

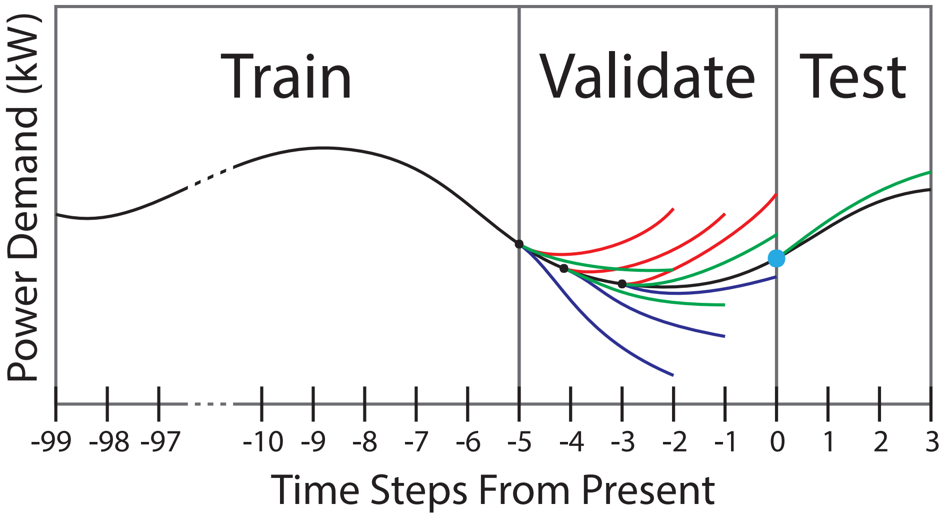

Figure A.10: Graphical Representation of Gated Ensemble Regression Method with 3 Models and 3 Hour Forecast Horizon

the forecast with the smallest positive or negative error. In this appendix, we will focus on producing the most accurate forecast as defined by the lowest root mean squared error (RMSE).

In the gating step, a method is applied to select a single model from the bucket of models according to its relative performance during the validation step. In other words, the gating method is responsible for choosing which model in the bucket of models will be used in the test step to generate the electricity demand prediction. The objective of the gating method is to select the best model based on present and/or past information from the validation step. In this appendix, we will implement and compare 2 alternative gating methods. The first method is cross-validation selection (CV). Put simply, the CV gate will select the forecasting model that performed best (produced the lowest RMSE) during the validation step. The CV gate can be considered a greedy approach because it has no memory of how its past decisions impacted the accuracy of the test prediction. Instead, the model selection is based entirely of the current performances in the validation step.

The second gating method uses a single recursively trained linear regression model (SR) to predict the performance of each forecaster in the bucket of models. Given n forecasters, the SR model input $x \in \mathbf{R}^{n+1}$ is the forecasting model's performance during the validation step (i.e. $x_{n+1} =$ Past RMSE of forecaster j) and a sparse binary vector representing each forecaster (i.e. for forecaster j, $x_j = 1$ and $x_i = 0$ for $i = 1, \ldots, j-1, j+1, \ldots, n$). The SR model output $\hat{y} \in \mathbf{R}$ is the forecaster's predicted performance during the test step (i.e. $\hat{y} =$ Predicted RMSE of forecaster j). In other words, the linear model includes a parameter for each of the regression models and a parameter corresponding to the validation RMSE. Thus, given a forecasting model and its performance during the validation step, we will train the linear model to predict the performance during the test step. In this way, the SR gate

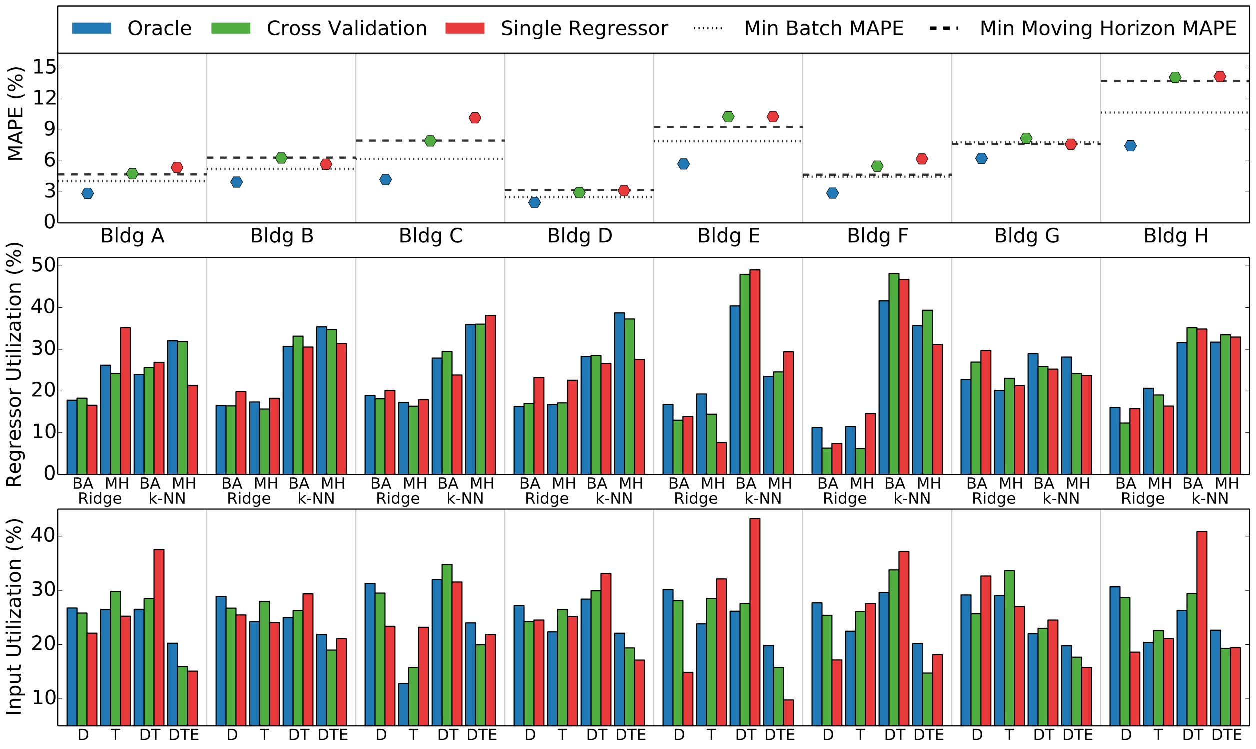

Figure A.11: Gated Ensemble Study Results

learns how well a model's validation performance does or does not indicate the performance during the test step. This capability should help to avoid models that perform poorly and favor models that perform well. The model with the best predicted performance (i.e. lowest $\hat{y}$) is selected by the SR gate for use in the test step.

Finally, in the testing step, the model selected in the gating step is used to generate an electricity demand forecast. Our multiple model method can be summarized by the following steps:

1. Train each model on a subset of the historical data.
2. Validate each model using historic data that immediately precedes the current time step.
3. Apply a gating method to select a model according to its performance during validation.
4. Use the selected model to generate a prediction.

A.5 Multiple Model Study

To test our approach, we have implemented the gated ensemble learning method using two regression models [Ordinary Least Squares with ℓ_2 Regularization (Ridge) and k-Nearest Neighbors (k-NN)] and four input types [electricity demand (D), time (T), electricity demand and time (DT), and electricity demand, time, and exogenous weather data (DTE)]. Despite the poor performance of the demand only and time only inputs in the batch and moving horizon studies, we have included them to observe if the gating methods choose to avoid the inputs. Therefore, the bucket of models will include the best (k-NN with DT) and the worst (Ridge with T) forecasting models from the single model studies. We have elected to exclude the ANN models from the ensemble due to their computational complexity.

The regression models are trained in both batch (BA) and moving horizon (MH) manners, for a total of 16 forecasting models per building. Next, we generated a 6 hour electricity demand forecast ($\hat{y} \in \mathbf{R}^6$) for every hour between July 1st, 2013, and January 1st, 2014 using the Training, Validation, Gating, and Testing steps described above. The batch models are trained once on 18 months of data and the moving horizon models are trained at every time step on 3 months of data.

The results, employing both of the gating methods described, are presented in Figure A.11. For testing purposes, we have also implemented an Oracle gate, which simply selects the best prediction for each time step regardless of the performances of the models in the validation step. The results from the Oracle gate represent the theoretical optimal of the ensemble approach.

In Figure A.11, the top subplot presents the overall performance of each gate as measured by the MAPE of the selected predictions. The dotted lines represent the best performance (lowest MAPE) of any single model from the batch study and the dashed lines, from the moving horizon study. The results show that our ensemble method is able to perform comparably to the best forecaster from the batch and moving horizon studies without any prior knowledge of the energy end-use or the relative performance of the contained forecasting models. The Oracle gate is able to outperform the ANN models from the batch study, however, the cross-validation gate (CV) and single regression (SR) gate are not. Among the gating methods studied, there is no clear winner. The SR gate shows a small advantage in buildings B and G, but the worst performance in building C. In each of the buildings, the CV gate performs comparably to the best model from the moving horizon study. The performance of the Oracle gate, particularly in buildings C, E, and H, suggests there is still potential for a gating method (not presented here) to further improve our ensemble approach.

The second subplot shows the percent utilization of each regression model by the Oracle, CV, and SR gating methods, regardless of input type. In other words, the plot shows the percentage of time steps that a particular regression model and training manner was selected by the respective gating method. Because the Oracle gate represents the optimal decision given the forecasting models in the set, we would like to see comparable utilization percentages between the Oracle and the other gates. Based on the subplot, this is generally the case with respect to regression model selection.

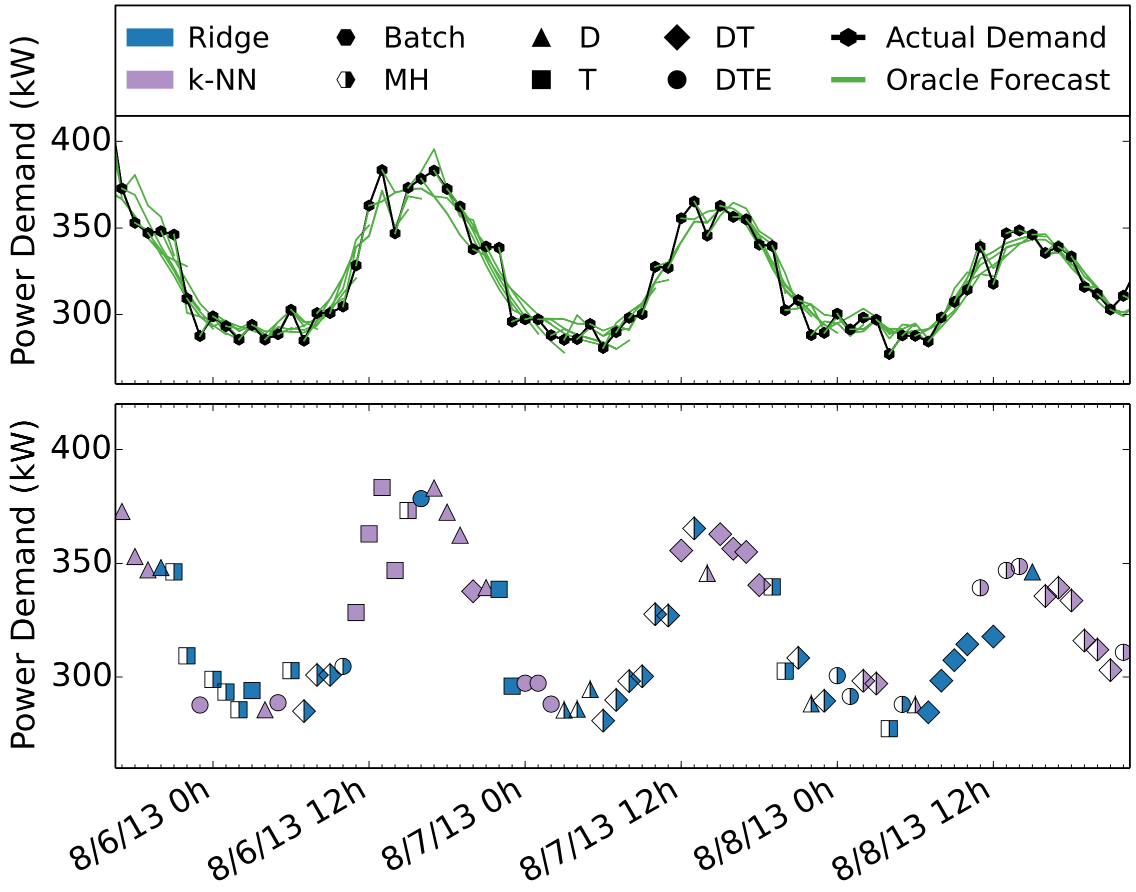

Figure A.12: Building A Oracle Forecast Sample

The bottom subplot shows the percent utilization of each input type by the Oracle, CV, and SR gating methods. Here, we observe larger differences between the behaviors of the gates. For Buildings A, E, and F, the SR gate favors the electricity demand and time (DT) inputs much more than the Oracle gate. For Buildings E and D, the SR gate underutilizes the electricity demand only (D) input compared to the CV and Oracle gates. For Building C, the CV and Oracle gates avoid the time only (T) input but the SR gate does not. It should be noted that neither of these subplots quantifies the impact of the percent utilizations on the MAPE of the gates. Nonetheless, the percent utilization metric provides useful insight into the decision making of each gate.

In general, we see very similar utilization rates between the Oracle and CV gates. This does not mean that both gates select the same forecaster at the each time step, but does suggest that the CV gate is capable of identifying forecasters that work well for a certain building. In Buildings E and F, we see that all 3 gates favor the k-NN models over the Ridge models. Additionally, in each of the buildings, the inclusion of exogenous weather data does not appear to significantly improve the accuracy of the predictions, as suggested by

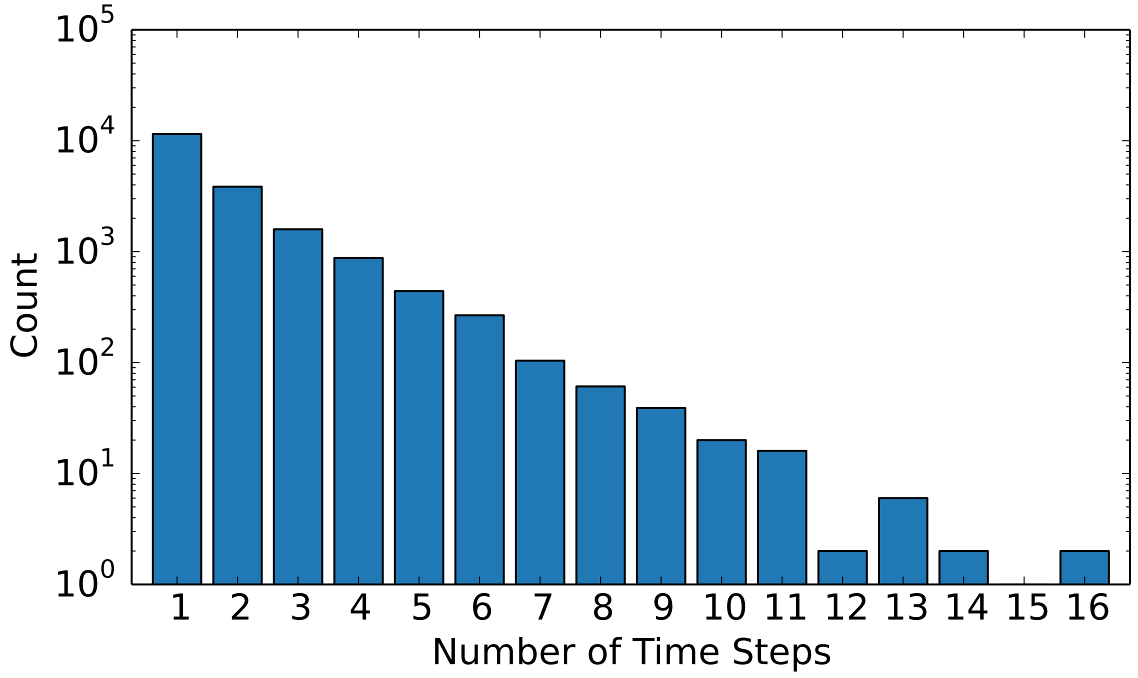

Figure A.13: Optimal Prediction Generation by a Forecaster over Consecutive Time Steps When Applied to Commercial Dataset

the Oracle's DTE utilization rate. This could mean that electricity demand is not strongly correlated with weather or that correlation only appears under certain conditions (e.g. an unusually cold or warm day).

Figure A.12 presents a sample of the predictions for Building A. The top subplot shows the actual electricity demand data from 8/6/13 to 8/9/13 and a series of multivariate forecasts selected by the Oracle gate (i.e. the most accurate forecast at each time step). The bottom subplot details which forecasting model generated the selected prediction. The marker color denotes the regression model and the marker shape, the input type. A filled marker indicates that the model was trained in a batch manner and a half-filled marker, a moving horizon manner.

Since the Oracle gate chooses the most accurate forecast at each time step, Figure A.12 supports the notion that a certain forecaster may generate the best prediction several time steps in a row. Figure A.13 shows, for all 8 buildings, the number of times a forecaster produced the best prediction over multiple consecutive time steps and the lengths of such sequences. For the validation step to properly inform the gating method, we would like to observe high frequencies of large repeated model sequences. Using fewer models would, of course, increase the probability of such sequences at the loss of potential performance.

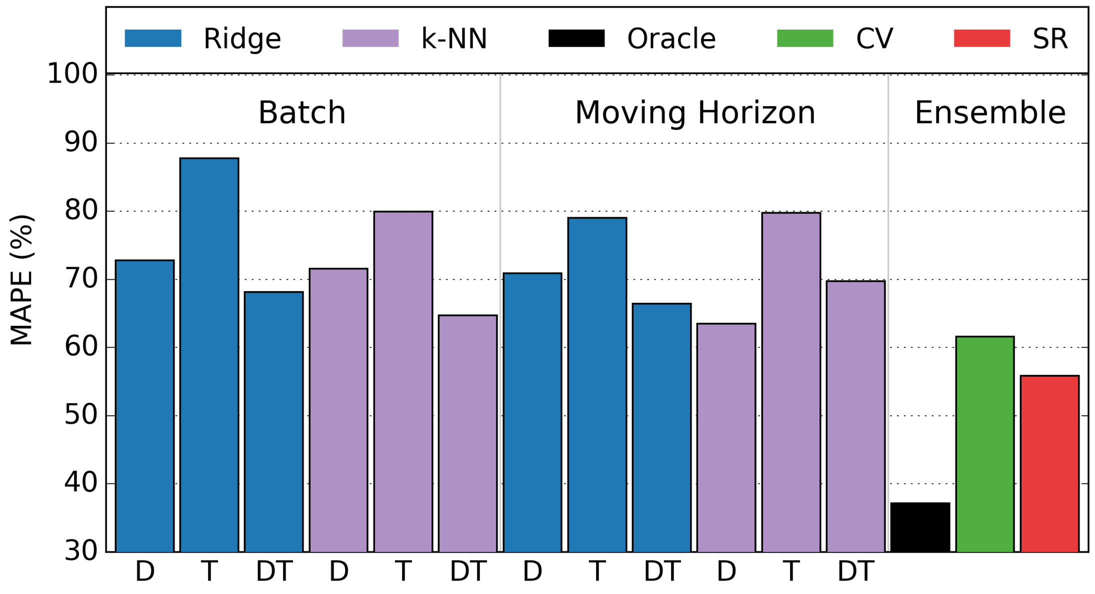

Figure A.14: Residential Study Results

A.6 Residential Study

To further test our ensemble approach, we have repeated the single model batch study, single model moving horizon study, and multiple model ensemble study using 24 residential electricity demand datasets. This time-series demand data has been downloaded by the individual customers from the electric utility's website and provided to our research group. Each dataset is from a home in northern California but has been anonymized such that we do not know its exact location. Accordingly, for this study, we have excluded the forecasting models that use weather data as inputs.

We utilize two regression models [Ordinary Least Squares with ℓ_2 Regularization (Ridge) and k-Nearest Neighbors (k-NN)] and three input types [electricity demand (D), time (T), and demand and time (DT)]. The regression models are trained in both batch (BA) and moving horizon (MH) manners, for a total of 12 forecasting models. In each of the studies, we generate a 6 hour electricity demand forecast ($\hat{y} \in \mathbf{R}^6$) for every hour between October 1st, 2014, and January 1st, 2015. The batch models are trained once on 9 months of data and the moving horizon models are trained at every time step on 3 months of data.

The results from the single model batch, single model moving horizon, and multiple model ensemble studies are presented in Figure A.14. In the figure, each bar represents the mean absolute percent error (MAPE) of the respective predictions across all 24 residential building datasets. Moving from left to right, the first 6 bars represent the results from the single model batch study and the next 6 bars, the single model moving horizon study. Here, we observe that the MAPEs are much larger for the residential predictions than for the

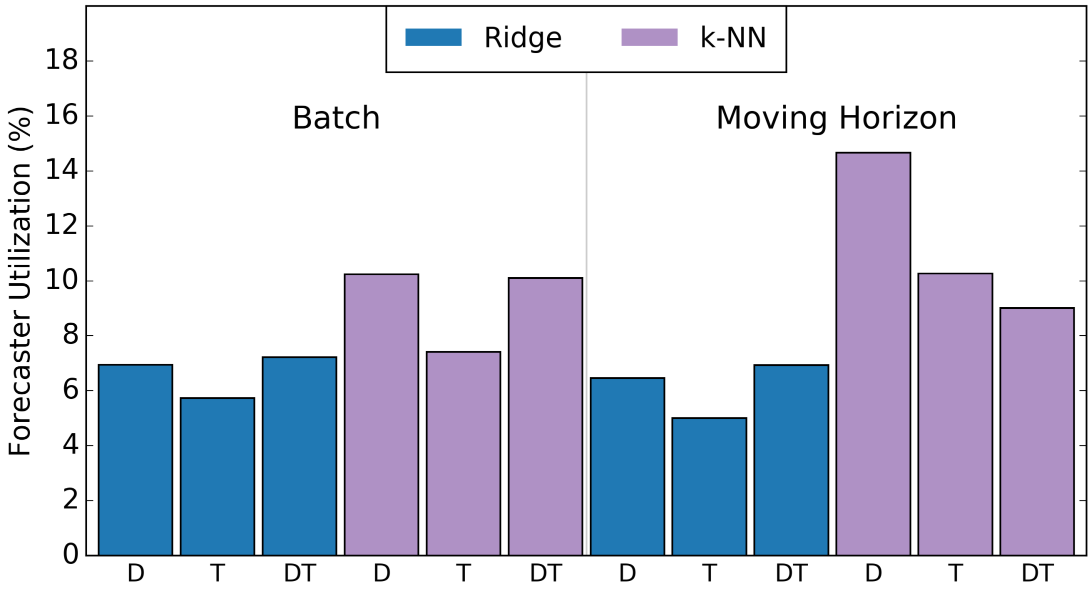

Figure A.15: Oracle Forecaster Utilization

commercial/university predictions. This can be explained by the fact that the residential electricity demands are much smaller and are composed of fewer loads than the commercial/university demands. Therefore, any change to the residential demand produces a larger percent change and any prediction error will correspond to a larger percent error. In the single model studies, the highest performing forecaster is the k-Nearest Neighbors regression model with demand input (D) trained in a moving horizon manner (MAPE: 63.5%).

In Figure A.14, the Oracle, CV, and SR bars show the results produced by our ensemble learning method. As previously stated, the results from the Oracle gate (MAPE: 37.1%) represent the optimal potential given the forecasters in the bucket of models. The CV gate represents the results (MAPE: 61.5%) using cross-validation to select which forecaster to utilize. Here, we observe that the CV gate performs comparably to the best forecaster in the single model studies. The SR gate represents the results (MAPE: 55.8%) using a single recursively trained linear regression model to select a forecaster by predicting the performance of each model. These results suggest that the SR gate outperforms the CV gate as well as any of the single model approaches. In other words, the SR gate is able to learn from the past performance of each model and to make better decisions about which model to select at each time step.

The Oracle gate's percent utilization of each forecaster in the bucket of models is shown in Figure A.15. These results show that, across the 24 residential electricity demand datasets, the Oracle gate shows a slight preference for the k-Nearest Neighbors models. This does not indicate that the k-NN models perform significantly better (lower RMSE) than the Ridge models, only that the k-NN models produce the best prediction with a higher frequency.

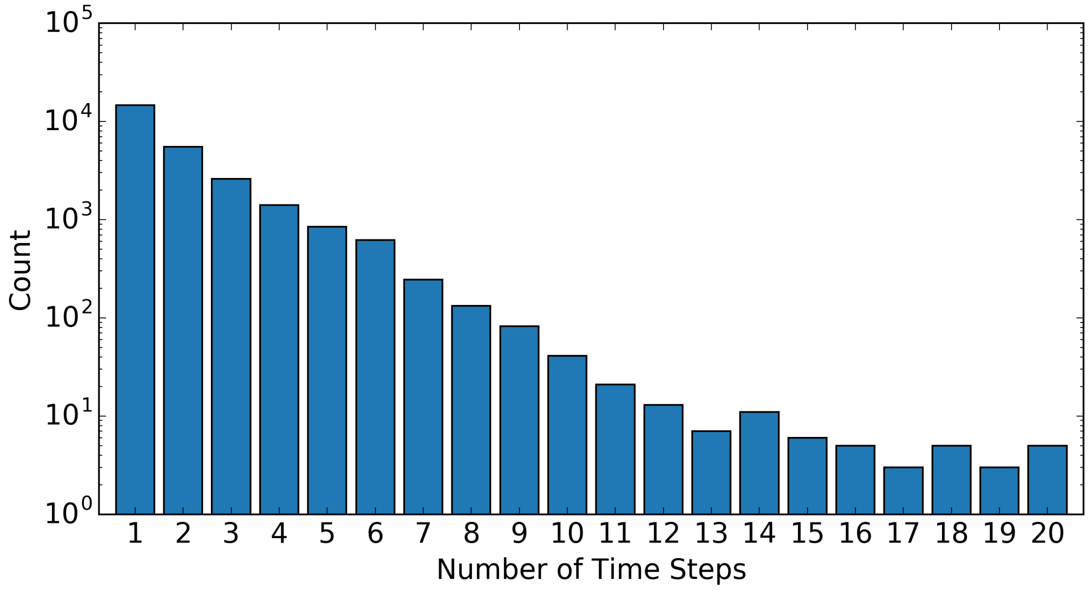

Figure A.16: Optimal Prediction Generation by a Forecaster over Consecutive Time Steps When Applied to Residential Dataset

Lastly, Figure A.16 shows, for all 24 residential buildings, the number of times a forecaster produced the best prediction over multiple consecutive time steps and the lengths of such sequences. Compared to the commercial/university buildings (Figure A.13), we observe larger sequences of optimal prediction generation when applying the ensemble method to the residential buildings. It should be noted that for the residential studies we have not incorporated weather data and are thus using 4 fewer forecasting models than in the commercial/university studies. Nonetheless, the Oracle's forecaster utilization rates (ranging from 5.0% to 14.6%) and the presence of large optimal prediction generation sequences support the underlying assertion of our ensemble learning method: by selecting between multiple models at each time step, we can obtain better results than a single model approach.

A.7 Conclusions

In this appendix, we have presented a gated ensemble learning method for short-term electricity demand forecasting. The contribution of this method is to allow for the incorporation of multiple forecasting models trained in both batch and moving horizon manners. Stated more explicitly, rather than choosing a single approach, an engineer can utilize multiple models with the intention of improving the reliability of the forecaster to produce useful results. This makes the method well suited for real world applications. At deployment, the moving horizon models can be utilized until sufficient data is available to train the batch models. This adaptability also makes the method suitable for control applications. Rather than assuming that demand behaviors are time invariant, the method will work to recognize both long and short-term electricity demand patterns.

The relative performance of the Oracle gate suggests that there is potential for continued development of the gating functions. For instance, none of the gating functions attempt to identify repeated model selection sequences. Given the optimal model in the previous few time steps, a gating method could determine the probability that a model will be optimal in the next time step. Also, in our implementation, the validation step uses the RMSE of 6 multivariate forecasts to measure the performance of each model. It may be more effective to use fewer forecasts or even univariate predictions in order to identify the optimal model at the given time step. Finally, the addition of a feature selection procedure may help to reduce the dimensionality of the regression models or even eliminate certain input types from consideration.

By applying our gated ensemble learning method to 32 unique building electricity demand data sets (8 commercial/university and 24 residential), we demonstrate that the incorporation of multiple models can yield better results than a single model approach. While the development of the gating methods is ongoing, the ability of each gate to perform model validation and selection in real time greatly improves the method's adaptability and ease of use. Utilizing this data-driven approach, we empirically show that the ensemble method is capable of aiding in the production of accurate multivariate electricity demand forecasts for building-level applications.

Appendix B

Model Predictive Control of Residential Baseboard Heaters with Distributed System Architecture

This appendix describes research on the application of model predictive control to the problem of optimizing the energy use of a residential baseboard heating system.

B.1 Motivation & Background

A common system for residential space heating is electric resistance baseboard heaters. Such systems are cheap to produce but costly to operate due to poor primary to end-use energy efficiency ($\sim$30%) and the varying cost of electricity [95]. However, as long as comfort levels are maintained, homeowners are indifferent to precisely *when* energy is used. Additionally, because homeowners often pay time-of-use electricity rates, we can know the price of electricity *a priori* and use the price as a proxy for power grid demand. By forecasting the space heating demand and incorporating the electricity price schedule, residential space heating can be controlled to minimize electricity costs. In practice, this optimal control strategy performs load shifting to avoid high electricity prices, thereby reducing peak loads on the grid.

The potential advantages of such load shifting, as well as models of thermal zones, system identification techniques, and model predictive control (MPC) strategies, are well developed in the literature. For example, [102] details the modelling, parameter estimation, and validation of a variable air volume (VAV) HVAC system. The work in [57] and [80] propose frameworks for online estimation of states and unknown parameters of buildings using extended and unscented Kalman filters. Model predictive controllers are implemented in [59, 58, 56] which optimizes the HVAC system to minimize total energy consumption, peak power consumption, and/or total comfort violations.

In these and other related works, the system architecture employed to implement these components is either briefly mentioned in the results, proposed in the conclusion, or entirely left out. To address this limitation, we have developed a distributed cloud-based system architecture for a residential heating system using readily available electronics and popular application development platforms. While we have selected a basic thermal model and a single system ID and MPC algorithm, future work will focus on comparing various alternatives to determine their relative advantages.

In this work, we demonstrate how these pieces can be linked into a robust and adaptable system. Our cloud-based system architecture enables us to remotely monitor and modify the control algorithm. Using mobile application platforms, we are able to move computationally intensive tasks to the cloud, thereby reducing the cost and complexity of local hardware. By breaking up the system's resources (database, linear program solver, model predictive controller, etc.) into distinct components, it becomes possible to modify sections of the system without reconstructing the whole. These system characteristics enable the implementation and continued development of the thermostatic control strategies presented in literature.

In this appendix, we propose a distributed cloud-based system architecture for a residential heating system. We then deploy a sensor network, local computer, and Internet server to collect real-time temperature data from an apartment. Using the sensor data, we perform system identification on the apartment using the online gradient update law algorithm. Next, we propose a model predictive control algorithm to minimize the cost of the heating system. Finally, we test the cloud-based system with simulated sensory input to demonstrate the economic advantage of the control strategy.

B.2 System Architecture

A fundamental challenge of a "smart" or "predictive" control system is access to the information and computing power needed to forecast and respond to future conditions. Therefore, we begin by proposing a distributed cloud-based system architecture. The term "distributed" refers to the incorporation of several autonomous computers that communicate with each other by passing messages. A distributed network can consist of different types of computers, each with an incomplete view of the system. The term "cloud-based" refers to the use of Internet-connected servers which allow a client computer to connect and perform a task from anywhere in the world.

The system architecture implemented in this work is shown in Figure B.1 where the rectangles represent physical hardware installed in the apartment, the clouds represent Internet servers or services, and the arrows indicate the flow of information. The temperature sensor network consists of six nodes placed throughout the apartment and communicating wirelessly with the ZigBee protocol [112]. Each sensor node, shown in Figure B.2, includes a Honeywell HIH6130 sensor, a Digi XBee S2 radio, and an Arduino Pro Mini board with an ATmega328 microcontroller. The local thermostat is an Internet-connected BeagleBone Black microcomputer that acts as a gateway between the heating system, the sensor network, and the cloud-based system.

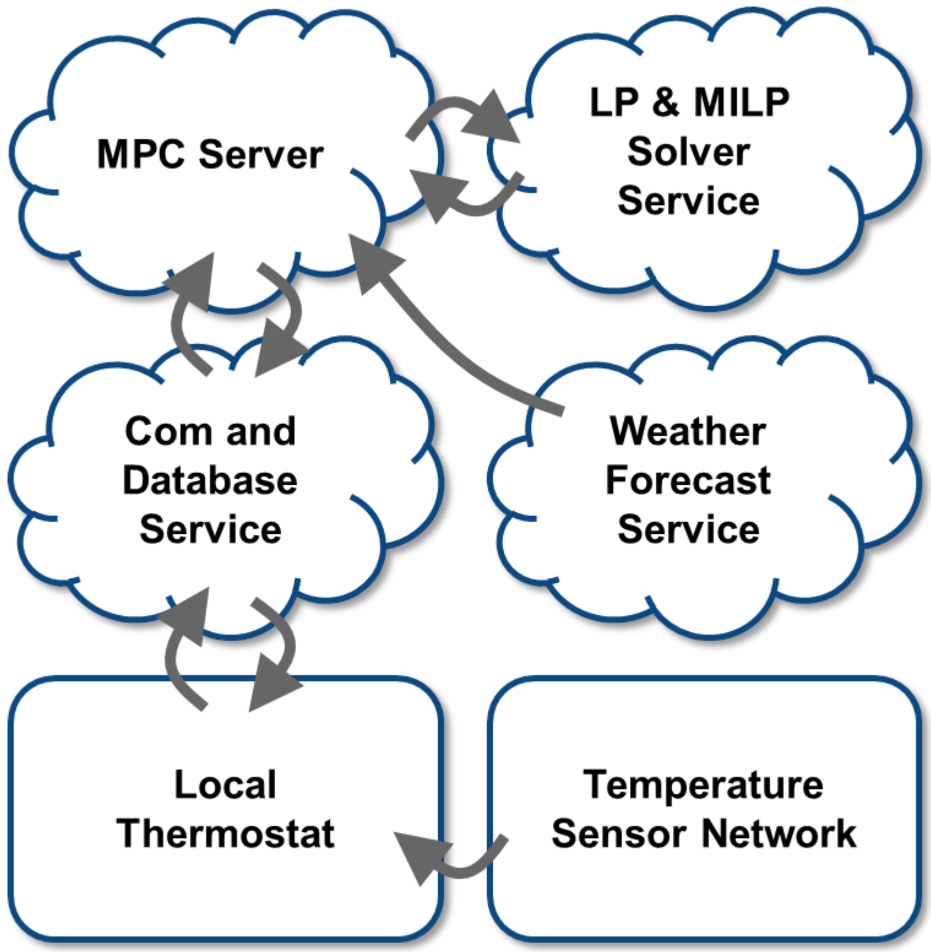

Figure B.1: System Diagram

The system parameters, electricity price schedule, optimization program, and control algorithm are coded into the MPC server and hosted by the mobile-application platform, Google App Engine [28]. Each of the cloud-based services include an application programming interface (API) built on the hypertext transfer protocol (HTTP) with JavaScript Object Notation (JSON) response format. This enables reliable communication across computing platforms and languages, an essential characteristic for our distributed system. The communication and database service, provided by Xively, implements an API to send, store, and retrieve time-series data [106]. This enables the thermostat and MPC server to asynchronously pass messages. The linear program (LP) and mixed integer linear program (MILP) solver is a service developed as part of this work. It consists of a solver hosted on the mobile-application platform, Heroku, and accompanied by an API for sending linear programs in a matrix/vector format [32]. Finally, the weather forecast is provided by the Internet service, Weather Underground [101].

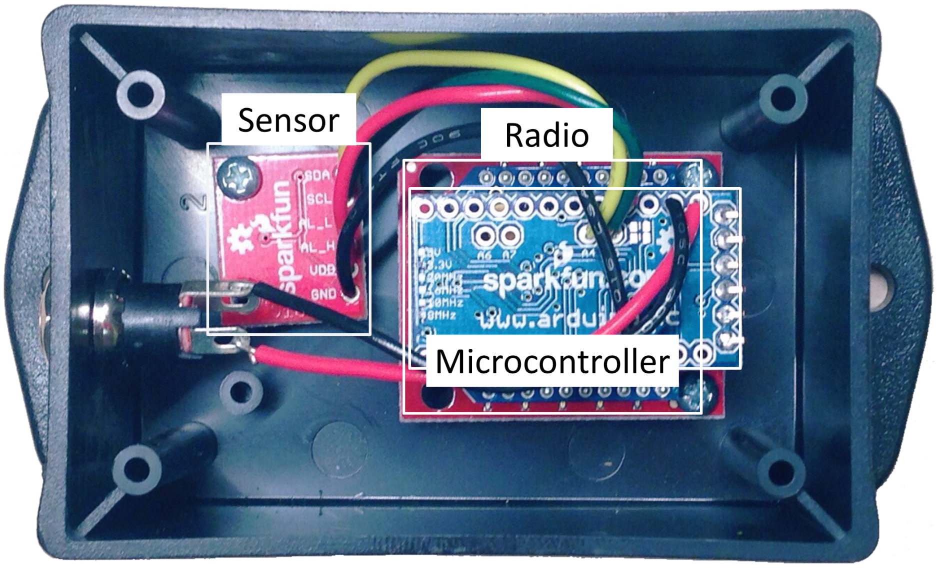

Figure B.2: Sensor Node

The system is executed as follows

- Sensor Network: Once per minute, the sensor network collects temperature measurements and sends them to the thermostat, which stores them in the database service. In this work, only one of the temperature streams is used for system ID and control.
- Weather Forecast: Once every 15 minutes, The MPC server retrieves and parses the weather conditions and temperature forecast.
- MILP: Once the forecast is parsed, the MPC server constructs the matrices of the MILP with the current sensor reading, weather forecast, and electricity price schedule for a 6 hour time horizon (N=24). These matrices are sent to the LP solver service.
- Optimal Solution: Once the problem is solved by the LP solver service, the response is parsed by the MPC server and the optimal heater state (ON or OFF) is sent to the online database service.
- Thermostat: Once every 5 minutes, the local thermostat checks the database service for a change in the optimal heater state record.

A necessity of any distributed computing architecture is fault tolerance. In this case, the local thermostat is programmed to fall back to simple deadband control in the case of communication failure or an infeasible solution from the LP solver service. The deadband control is implemented such that when a lower bound temperature is reached, the apartment is heated for one time step. Because the MPC server is only capable of load shifting (instructing the thermostat to turn the heater on before the lower bound temperature is reached), this fall back does not conflict with our control strategy.

Overall, this work is intended to serve as a demonstration of how distributed and cloud-based systems can aid engineers in developing smarter control systems. By utilizing platforms for electronics hardware, mobile-applications, and Internet services, it is possible to quickly prototype a distributed control system.

B.3 Model Development & Identification

Thermal RC Model

This work employs a simple continuous time RC model to capture the thermal characteristics of the apartment. The modeled dynamics include the heat transfer between the interior and the environment, power from the electric baseboard heaters, and solar heat gain. Radiative heat transfer from the ambient will not be considered. Therefore, the change in temperature within the apartment can be represented by the state equation

$$\dot{T}(t) = \frac{-1}{RC}T(t) + \frac{1}{RC}T_\infty(t) + \frac{P_H}{C}h(t) + \frac{1}{C}P_S(t), \tag{B.1}$$

where $T(t)$, $h(t)$, $T_\infty(t)$, $P_S(t)$ are the indoor temperature (state, °C), heater state (input, binary), outdoor temperature (disturbance input, °C), and solar gain (disturbance input, kW), respectively. The parameters used in this model are R (°C/kW), C (kJ/°C), P_H (kW) which represent the thermal resistance, thermal capacitance, and baseboard heater power, respectively. In this appendix, we will use "solar gain" to refer to power delivered to the apartment attributable to solar irradiance (kW/m^2).

Exogenous Signals

To utilize (B.1) in a model predictive controller, it is necessary to forecast the disturbance inputs, $T_\infty(t)$ and $P_S(t)$. Fortunately, there are several web services that provide searchable application programming interfaces (APIs) for temperature forecasts. Such services include Wundergound.com, OpenWeatherMap.com, and Forecast.io. Unfortunately, at the time of this writing, there is no equivalent service for solar irradiance forecasts, making it difficult to accurately predict solar gain. Therefore, we have created a means of approximating solar gain based on forecasted weather conditions. First, we redefine solar gain

$$P_S(t) = P_{S,max}s(t), \tag{B.2}$$

where $P_{S,max}$ is the maximum solar gain (kW) and $s(t)$ is a solar gain scaling factor. In this appendix, we assume that $P_{S,max}$ is constant, however $P_{S,max}$ will actually vary according to the position of the sun (i.e. season and time of day). Next, we use forecasted cloud cover conditions to define the scaling factor, $s(t)$, shown in Table B.1. Between sunrise and sunset, the input $s(t)$ is equal to a normalized number based on the weather condition. Otherwise (i.e. at night), $s(t) = 0$. This approach provides a crude but useful approximation of the impact of solar irradiance on the temperature inside the apartment.

Table B.1: Solar Gain Scale

Weather Condition		Scale	Normalized, s(t)
Day	Clear	4.5	1
	Scattered Clouds	3.5	0.778
	Partly Cloudy	3	0.667
	Mostly Cloudy	2	0.444
	Overcast	0.5	0.111
Night	N/A	0	0

Finally, (B.1) can be rewritten as

$$\dot{T}(t) = \frac{-T(t) + T_\infty(t)}{RC} + \frac{P_H h(t)}{C} + \frac{P_{S,max} s(t)}{C}. \tag{B.3}$$

Parametric Modeling

We reformulate the temperature dynamics equation into a linear in the parameters form for identification purposes. The linear model is derived from (B.3) as follows,

$$\dot{T}(t) = \begin{bmatrix} \frac{1}{RC} & \frac{P_H}{C} & \frac{P_{S,max}}{C} \end{bmatrix} \begin{bmatrix} T_\infty(t) - T(t) \\ h(t) \\ s(t) \end{bmatrix}. \tag{B.4}$$

Let

$$z(t) = \dot{T}(t), \tag{B.5}$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{RC} \\ \frac{P_H}{C} \\ \frac{P_{S,max}}{C} \end{bmatrix}, \tag{B.6}$$

$$\phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} T_\infty(t) - T(t) \\ h(t) \\ s(t) \end{bmatrix}. \tag{B.7}$$

Therefore,

$$z(t) = \theta^T \phi. \tag{B.8}$$

Identification Algorithm

For parameter estimation, we employ a normalized recursive gradient update law, given by,

$$\frac{d}{dt}\hat{\theta}(t) = \Gamma\epsilon(t)\phi(t), \qquad \hat{\theta}(0) = \hat{\theta}_0, \tag{B.9}$$

$$\epsilon(t) = \frac{z(t) - \hat{\theta}(t)\phi(t)}{m^2(t)}, \tag{B.10}$$

$$m^2(t) = 1 + \hat{\phi}(t)\phi(t), \tag{B.11}$$

System Identification Results and Validation

With $\Gamma = 10^{-4.1} * I$ and initial parameter guesses $\theta_1(0) = 0.00024$, $\theta_2(0) = 0.00045$, and $\theta_3(0) = 0.000145$, the resulting parameter estimates are shown in Figure B.3. The dataset used for identification is shown in Figure B.4, where $\hat{T}(t)$ is the estimate and $T(t)$ is the measurement. Due to the lack of persistent excitation in regressor element $h(t)$, θ_2 required manual tuning to find a suitable estimate [39]. In other words, the heater was not turned on often enough for θ_2 to be identifiable using the recursive gradient descent algorithm.

For model validation, we rewrite the system in state-space form,

$$\dot{T}(t) = \begin{bmatrix} -\theta_1 \end{bmatrix} \begin{bmatrix} T(t) \end{bmatrix} + \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix} \begin{bmatrix} T_\infty(t) \\ h(t) \\ s(t) \end{bmatrix}, \tag{B.12}$$

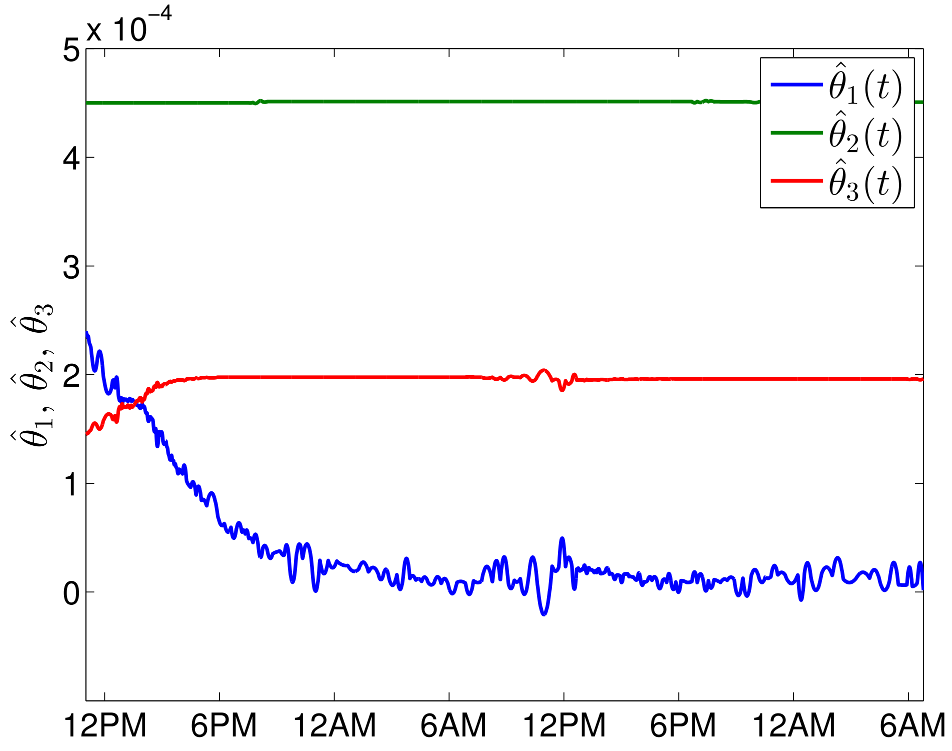

Figure B.3: Parameter Estimates

$$u(t) = \begin{bmatrix} T_\infty(t) \\ h(t) \\ s(t) \end{bmatrix}. \tag{B.13}$$

From Fig. B.3, the exit estimates are

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 1.054 * 10^{-5} \\ 4.51 * 10^{-4} \\ 1.960 * 10^{-4} \end{bmatrix}. \tag{B.14}$$

Using these parameter estimates and the state-space model, we can validate against a different dataset, shown in Fig. B.5. The validation results show a root mean square error of 0.9130 °C.

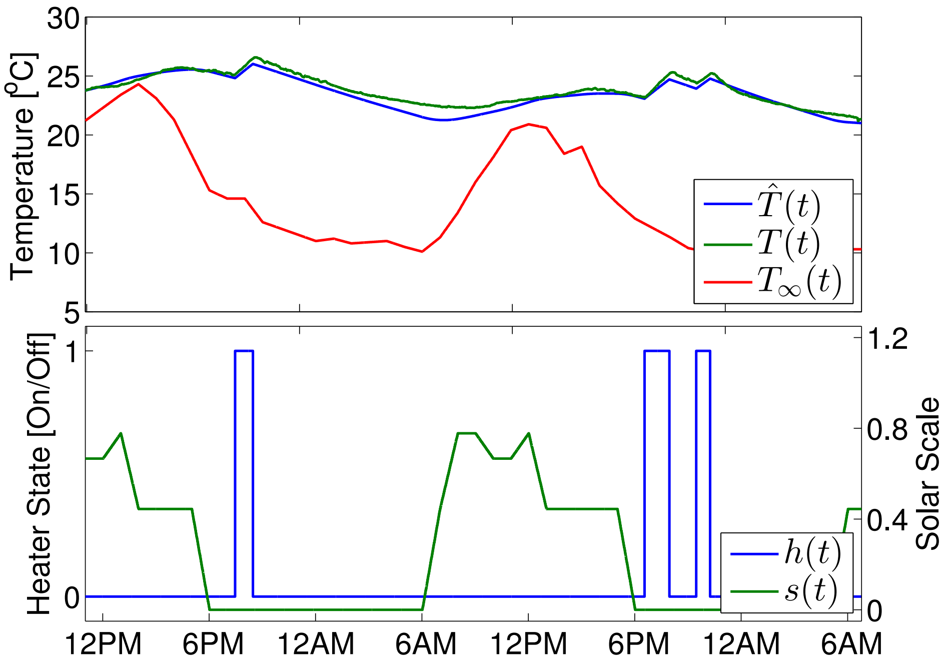

Figure B.4: System Identification Dataset

B.4 Model Predictive Control

Model Discretization

Since the model we identified above is in the continuous time domain, we now transform the model into the discrete time domain for use in the optimization program.

Recall the continuous time state-space model, (B.12), where

$$A = \begin{bmatrix} -1.054 * 10^{-5} \end{bmatrix}, \tag{B.15}$$

$$B = \begin{bmatrix} 1.054 * 10^{-5} & 4.51 * 10^{-4} & 1.960 * 10^{-4} \end{bmatrix}. \tag{B.16}$$

We obtain the discrete time state-space form

$$T(k+1) = A_d \begin{bmatrix} T(k) \end{bmatrix} + B_d \begin{bmatrix} T_\infty(k) \\ h(k) \\ s(k) \end{bmatrix}. \tag{B.17}$$

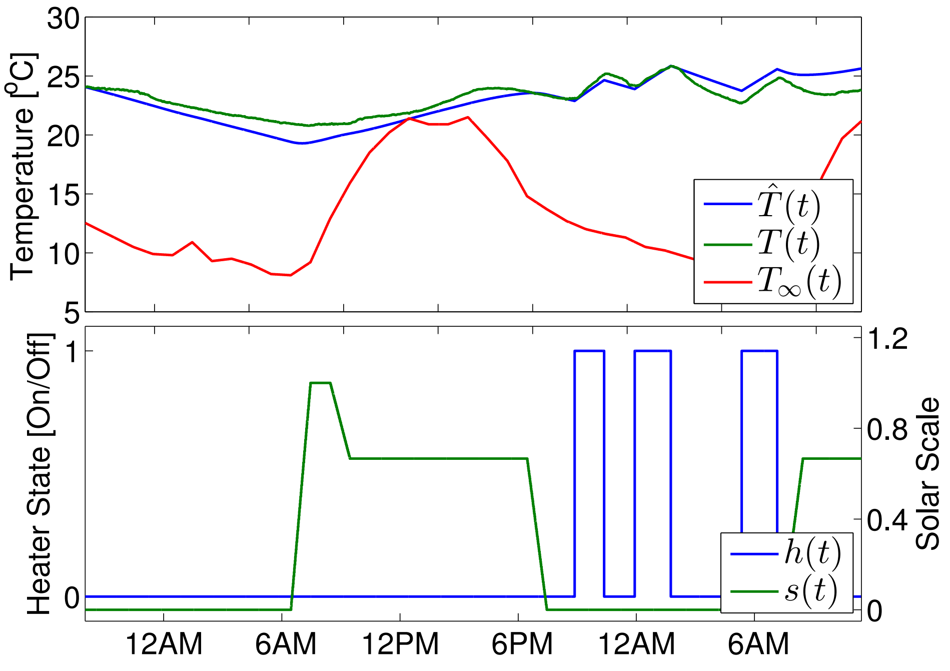

Figure B.5: Validation Dataset

by using the transformations

$$A_d = e^{A\Delta t}, \tag{B.18}$$

$$B_d = \left(\int_0^{\Delta t} e^{A\tau} d\tau\right) B. \tag{B.19}$$

Since the A matrix is non-singular, we can find B_d as

$$B_d = A^{-1}(A_d - I)B. \tag{B.20}$$

We solve for these discrete matrices with Δt = 15 minutes = 900 seconds, resulting in

$$A_d = \begin{bmatrix} 0.9906 \end{bmatrix}, \tag{B.21}$$

$$B_d = \begin{bmatrix} 0.009441 & 0.4040 & 0.17557 \end{bmatrix} \tag{B.22}$$

Optimization Formulation

The optimization program is formulated as

$$J = \min_{h(0)} \sum_{k=1}^{N-1} c(k) P_H h(k) \frac{\Delta t}{3600}, \tag{B.23}$$

subject to

$$T(k+1) = A_d \begin{bmatrix} T(k) \end{bmatrix} + B_d \begin{bmatrix} T_\infty(k) \\ h(k) \\ s(k) \end{bmatrix}, \quad k = 0, ..., N-1, \tag{B.24}$$

$$T(k+1) \geq T_{min}, \qquad k = 0, ..., N-1, \tag{B.25}$$

$$h(k) = \{0, 1\}, \quad k = 0, ..., N-1, \tag{B.26}$$

$$T(0) = T_0. \tag{B.27}$$

The optimization program is a mixed integer linear program (MILP) which minimizes the cost of electricity while keeping the indoor temperature above a minimum setpoint. Since the home has an electric-resistance heating system, the optimal decision variable $h^*(k)$ will be binary (1 or 0) representing an ON or OFF state, respectively. The electricity prices, $c(k)$, are shown in Table B.2. The off, partial, and peak costs are based on PG&E's weekday summer rates for residential time-of-use schedule E-6 [74]. The morning and evening peak rates were added to make the problem more interesting. For the purposes of simulating the system, we have assumed a heater power, P_H, of 1.5kW and a minimum temperature, T_{min}, of 20 °C.

Table B.2: Residential Time-of-Use Price Schedule

	Time	Cost($/kWh)
Off Peak	12:00AM to 7:00AM & 11:00PM to 12:00AM	0.10376
Morning Peak	7:00AM to 9:00AM	0.25913
Partial Peak	9:00AM to 2:00PM & 9:00PM to 11:00PM	0.18054
Peak	2:00PM to 4:00PM & 6:00PM to 9:00PM	0.29581
Evening Peak	4:00PM to 6:00PM	0.44012

As formulated, it is possible for the MILP to return an infeasible solution error. For example, if $T(0)$ is low and $T(k+1)$ cannot be raised above T_{min} in one time step or if the losses to ambient exceed the energy delivered by the heater for several time steps, then the minimum temperature constraint will be violated and the solver will return no solution. This limitation could be resolved by penalizing the MILP for allowing $T(k+1)$ to fall below T_{min} rather than applying T_{min} as a lower bound constraint.

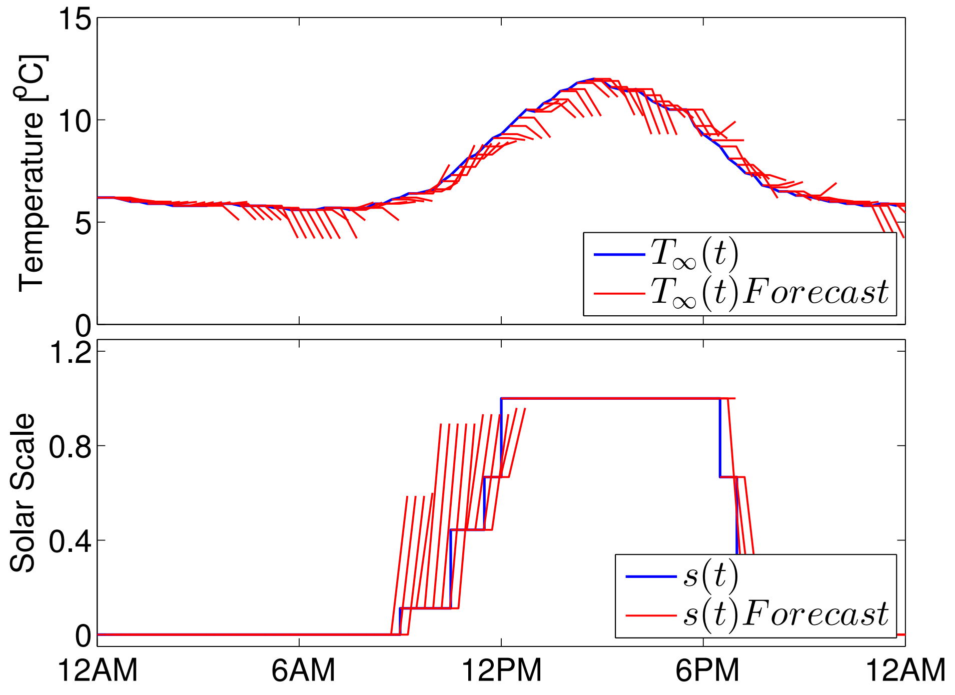

Figure B.6: Weather Forecast Data

Control Algorithm

The MPC algorithm is executed as follows

1. Set the current temperature measurement as the initial state, $T(0) = T_0$.
2. Solve the MILP for the optimal open loop input sequence $h^*(0), h^*(1), .., h^*(N-1)$, given forecasts of disturbances $T_\infty(t)$ and $s(t)$.
3. Implement the first input $h^*(0)$ to advance the system one time step.
4. Repeat the algorithm at the next time step.

An advantage of this control algorithm is that a system can be predictively controlled despite inaccuracies in the identified model parameters. In this work, the MILP was solved for a 6 hour time horizon (N=24). It is unlikely that the model accurately predicts the temperature in the apartment 6 hours into the future. Nonetheless, the MILP is capable of estimating the impact of the system's inputs and helping to determine the optimal course of action in the next time step.

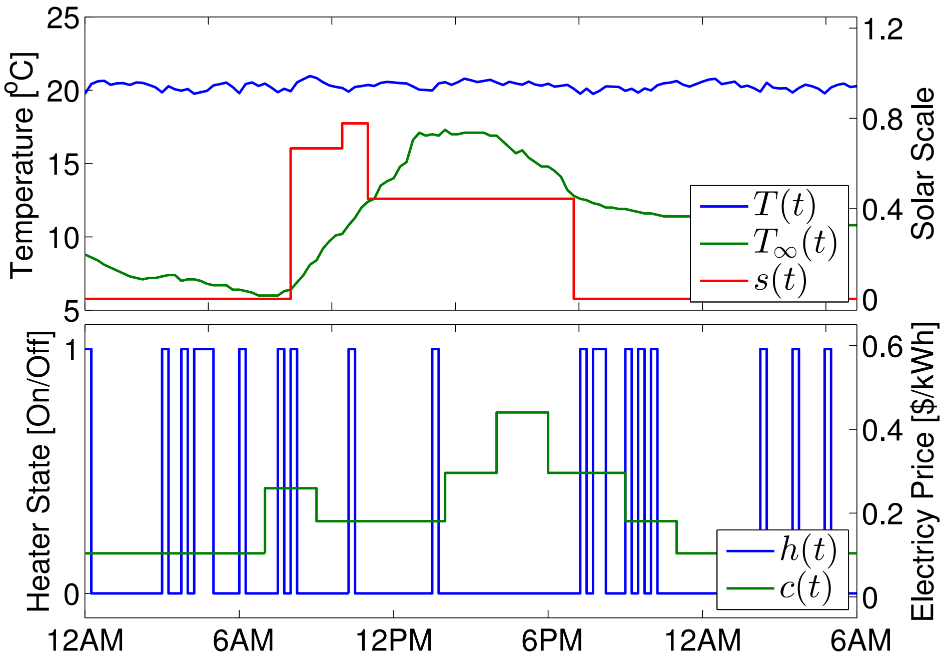

Figure B.7: Deadband Control Simulation

Simulation Results

A real-time software-in-the-loop simulation was conducted to compare the performance of a traditional deadband controller with our model predictive controller. The identified discrete thermal model with added Gaussian noise was used to represent the evolution of the temperature inside the apartment. The simulation was run for approximately 2 weeks using actual weather forecast data, an example of which is shown in Figure B.6. A subset of the results are presented in Figures B.7 and B.8.

As shown in the Figure B.7, the deadband controller switches on whenever energy is needed to maintain the desired temperature in the apartment. In contrast, the MPC system avoids peak and partial-peak electricity prices, resulting in the more concentrated periods of heating shown in Figure B.8.

Based on the real time simulation, both the MPC and deadband controllers consume close to the same amount of energy (42.0 kWh and 46.4 kWh, respectively) and maintain comparable average temperatures (21.37 °C and 21.06 °C, respectively) during the 2 weeks studied. However, the MPC system showed a 31% reduction in heating costs compared to the traditional deadband controller. This suggests that the MPC system meets the objective of reducing electricity cost by shifting the time of electricity use rather than decreasing the total energy demand.

While "a 31% reduction in heating costs" is an encouraging finding, it is an insufficient metric for representing the performance of our MPC system. Firstly, it compares MPC with deadband control, which is a trusted but ultimately rudimentary control strategy. By

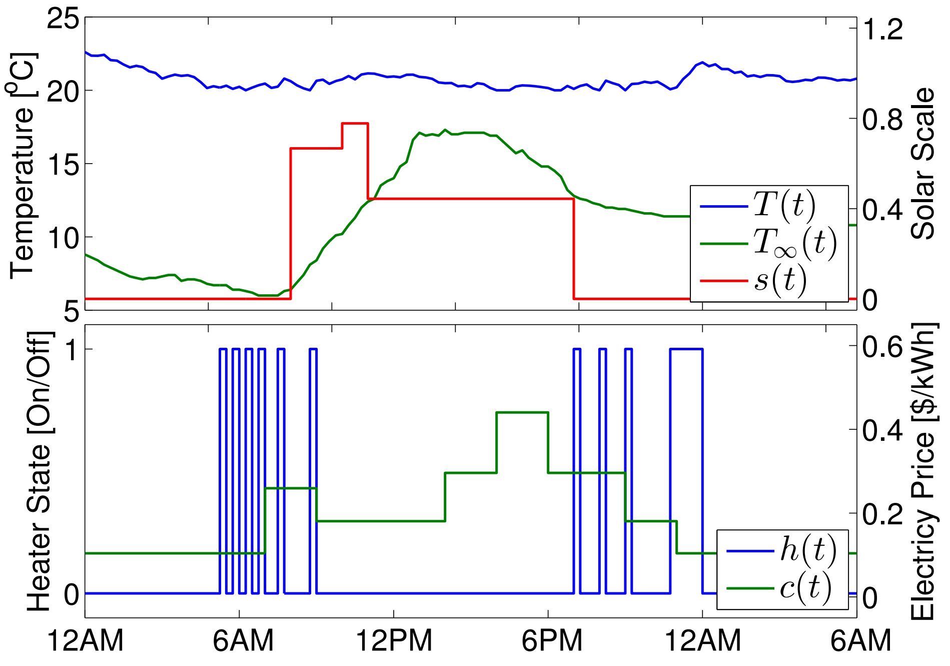

Figure B.8: MPC Simulation

comparing to a strategy that we know to perform poorly, the metric gives an exaggerated impression of success. Secondly, the metric fails to recognize regional or seasonal changes in weather conditions and energy prices. For example, a 5% reduction in cost in a cold climate or during a winter month may be far more significant than a 30% reduction in a temperate climate or during an autumn month. What is needed is a means of defining an optimal control strategy with perfect forecasting. Then, rather than making claims of cost savings, it is possible to express how a system performs relative to the optimal solution.

For the purposes of this work, we have formulated a dynamic program (DP) that implements the same objective and system dynamics as the MILP above. Then, we collected weather forecast data for a period of 3 weeks. For each day, we simulated the thermostatic control system with varying degrees of foresight. The deadband controller (DBand) has no forecasting capability and simply turns on the heater for one time step when the lower bound condition is reached. The MPC controller (MPC) was run using the actual (imperfect) weather forecasts at each time step with forecast horizons of 1, 3, 6, and 10 hours. A Deterministic MPC controller (DMPC) was simulated using the recorded conditions as (perfect) weather forecasts with forecast horizons of 1, 3, 6, and 10 hours. Lastly, the DP was simulated to determine the optimal solution to the given objective and compared with the results of the DBand, MPC, and DMPC simulations, as shown in Figures B.9 and B.10. The cost optimality (CO) for each day simulated is defined as

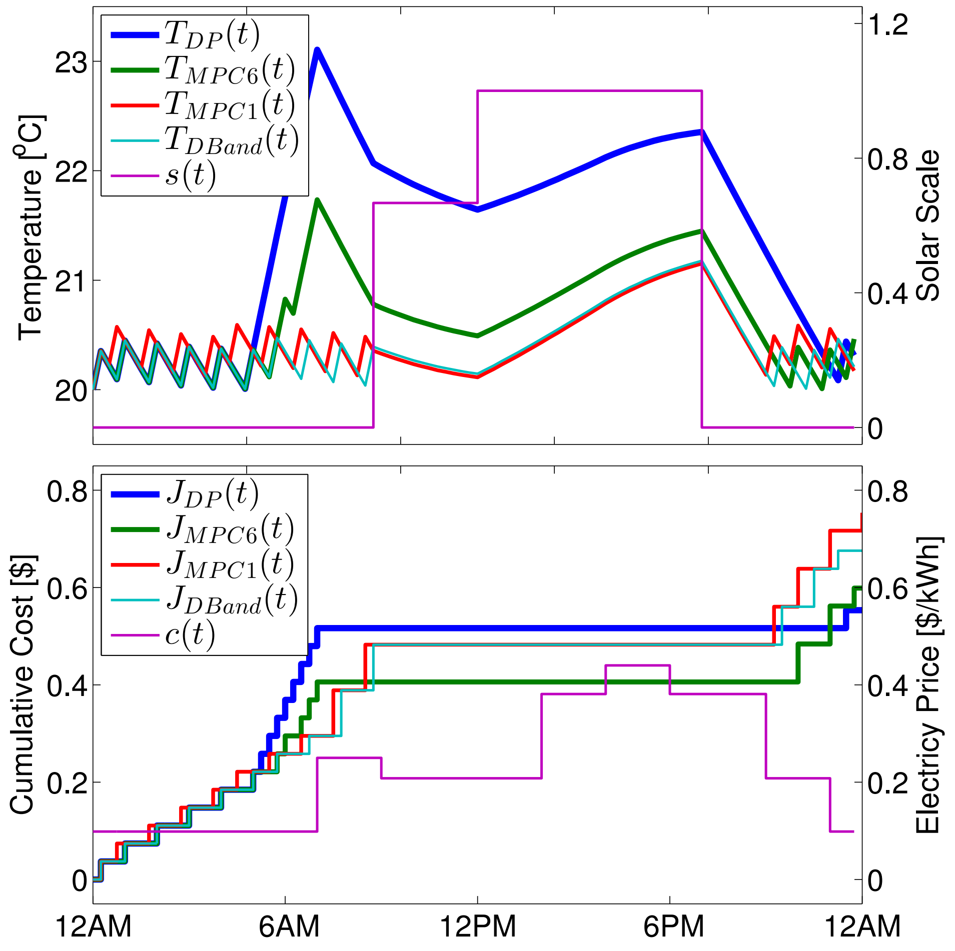

Figure B.9: Simulation Example

$$CO(i,j) = \frac{J(DP,j)}{J(i,j)}, \tag{B.28}$$

where
i = Controller [DBand, MPC, DMPC, DP]
j = Day simulated (out of N=20)
$J(i,j)$ = Cost of heating on day j with controller i

By comparing the results of each controller, we are able to identify some of their behaviours. For example, the MPC and DMPC controllers with 1 hour forecast horizons are not able to see changes in the price of electricity until 1 hour before they take effect. As a result, the controllers are less able to avoid high electricity prices than the DP and, on average, incur higher costs than the non-forecasting DBand controller. For each time horizon, the difference in the cost optimality between the MPC and DMPC simulations indicate the

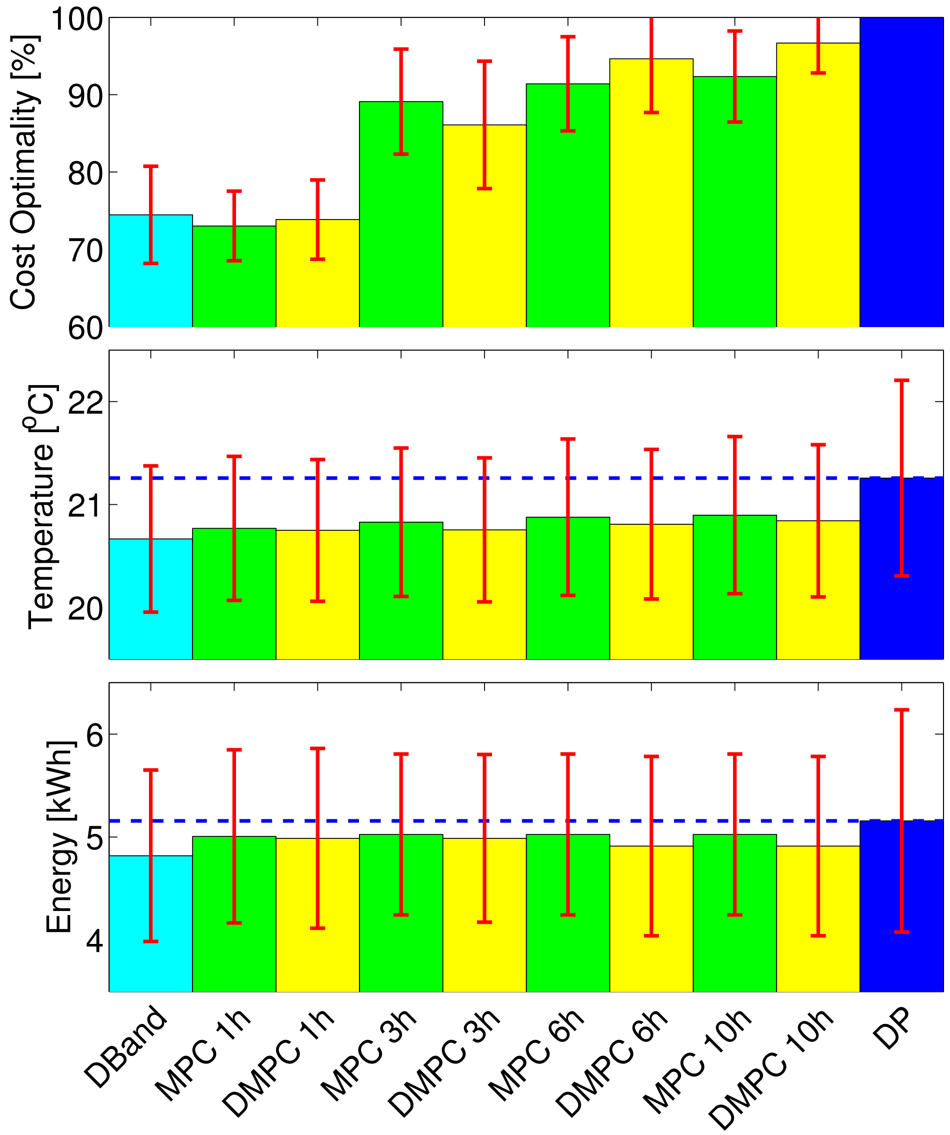

Figure B.10: Mean Cost Optimality, Temperature, and Energy Use per Day over 20 days. Error bars indicate standard deviation over 20 days.

benefit of accurate weather forecasting. By comparing the various forecast horizons of the MPC or DMPC simulations, we can see the diminishing returns of foresight. Additionally, we can see that as the horizon increases, the controller will maintain a slightly higher average temperature so as to minimize heating during on-peak periods.

B.5 Conclusions

The results of our work show that the system is capable of a) monitoring the real-time thermal conditions in a space, b) reducing peak loads on the power grid by using electricity price as a proxy for peak demand, c) making use of weather forecast data to predict heating requirements according to the apartment's thermal dynamics, and d) enable smart home technology development with a distributed and cloud-based system architecture. Additionally, this simple MPC control system is capable of reducing the consumer's electricity costs while maintaining a given temperature range.

Appendix C

TCL Simulation Studies

This appendix expands on the TCL parameter estimation results in Chapter 3. Below are simulation studies demonstrating the capability of the algorithm to quickly converge to new parameter estimates and evaluating the advantage of employing a recursive algorithm in a demand response context.

C.1 Adaptive Parameter Estimation

In this section, we demonstrate the capability of the single UKF method (presented in Chapter 3, Section 6) to quickly converge to new parameter estimates in response to changes in the system. In this simulation study, we first generate data of a residential refrigerator with time-varying parameters. We then employ the single UKF method to recursively estimate the parameters using the temperature and compressor state signals. We have tuned the values of the covariance matrices Q_v and Q_w to allow the UKF to converge to new parameter estimates more quickly than in the previous TCL studies.

The refrigerator simulated in this study has parameters which vary periodically due to the loss or gain of thermal capacitance and changes in the unmodeled noise process. The change in thermal capacitance is represented by randomly drawing a value for θ_1 from the uniform distribution [0.985, 0.995] once an hour and drawing θ_3 from the uniform distribution [-0.16,-0.08] once every 3 hours. We assume that θ_2 is constant and equal to -40. Internal temperature and compressor state data is generated by simulating the deadband control of the system for 7 days using the random time-varying parameters and employing the ambient temperature data collected by TCL_1.

Next, we apply the single UKF method to recursively estimate the model parameters based on the simulated internal temperature and compressor state signals. Figures 3.21 and 3.22 presents examples of the time-varying system parameters and the parameter estimates produced by the single UKF method. The figure illustrates how the UKF is able to converge to new parameter estimates within a few time steps. We observe that the UKF often updates both θ_1 and θ_3 in response to changes in the thermal capacitance of the system. This

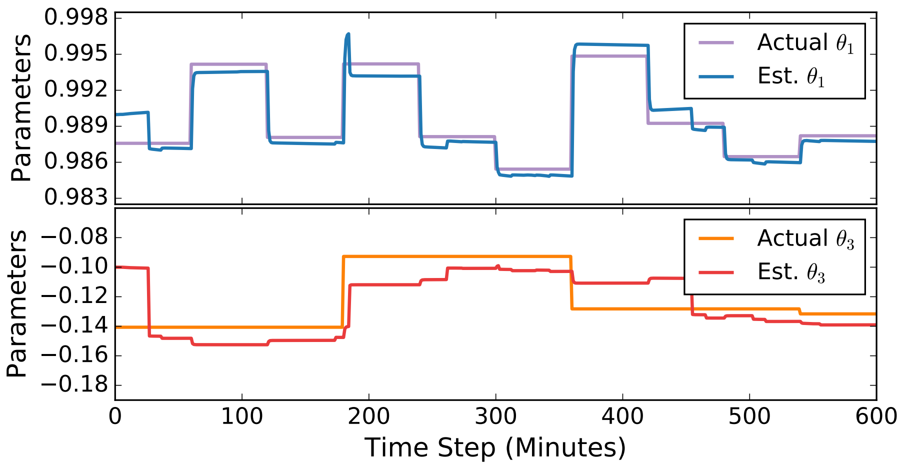

Figure C.1: Adaptive Parameter Estimation of Simulated TCL (estimates over 10 hours)

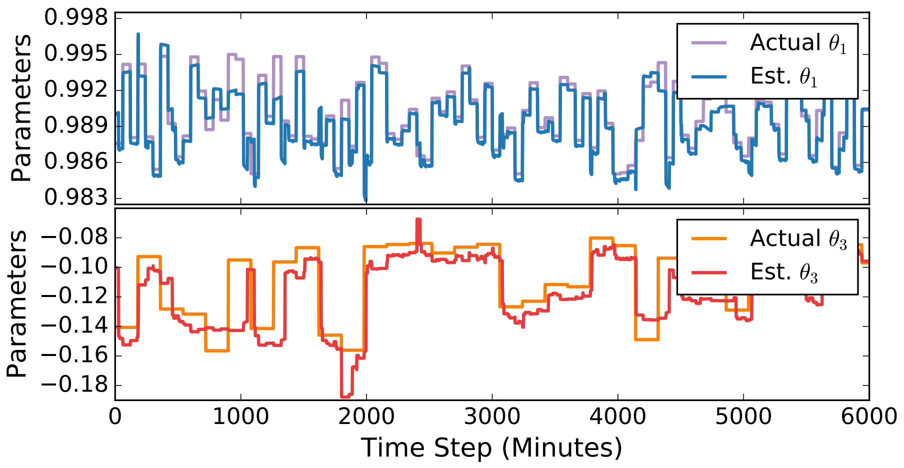

Figure C.2: Adaptive Parameter Estimation of Simulated TCL (estimates over 100 hours)

observation is comparable to how the parameter estimates changed in response to removing thermal mass from TCL_2, as shown in Figure 3.22 (Chapter 3, Section 7).

To quantify the advantage of using an adaptive modeling approach, we compare the performance of 3 observers which model the fridge and produce 1-hour ahead forecasts of the internal temperature and compressor state. The first observer, referred to as the "static" observer, employs a model with fixed parameters ($\theta_1 = 0.99$, $\theta_2 = -40$, and $\theta_3 = -0.12$). The second observer, referred to as the "adaptive" observer, employs the parameter estimates produced by the single UKF method. Note that the adaptive observer uses the model parameters as estimated at the time step that the forecast was produced and that the parameters do not vary within a 1-hour ahead forecast. Finally, the third observer, referred to as the "informed" observer, has perfect knowledge of the system parameters at the time that each forecast is produced. Note that the informed observer does not have knowledge of how the parameters will change in the future and thus the parameters do not vary within a 1-hour ahead forecast.

For each observer, we produce a 1-hour ahead forecast of the internal temperature and compressor state starting at each minute over a 7 day period. The performance of each observer is quantified as the root mean squared error (RMSE) of the forecasts compared to the simulated data with random time-varying parameters. The RMSE of the internal temperature forecasts for the static, adaptive, and informed observers are 1.37°C, 1.12°C, and 0.73°C, respectively, and for the compressor states are 0.402, 0.305, and 0.234, respectively. Based on these results, the adaptive observer reduces the temperature forecast RMSE by 24.1% and the compressor state forecast RMSE by 18.2% relative to the static observer. Assuming that the electrical power demand of the TCL is linearly proportional to the compressor state, the adaptive observer also reduces the power demand forecast RMSE by 18.2% relative to the static observer.

C.2 TCL Demand Response

In this section, we present a simulation study in which a population of residential refrigerators optimizes its power demand according to a price signal from a demand response event using model predictive control (MPC). Because the individual TCLs all respond to the same price signal and do not coordinate with each other or with a grid entity, the demand response approach can be described as decentralized model predictive control. The objective of this study is to illustrate the impact of model fidelity on the decision making of the TCLs. Specifically, we illustrate the advantage of using the single UKF method to recursively estimate a system's parameters rather than employing a fixed set of parameters for model predictive control. By more accurately representing the dynamics of a TCL, a controller with recursive parameter estimation is capable of identifying and implementing a better solution.

To avoid using electricity during the hour long event, the fridges are able to shift their demand by decreasing their temperature setpoints by 1°C for a certain amount of time. Note that the setpoint will never deviate by more than −1°C from the original temperature setpoint. The power demand of a TCL at each time step is defined by

$$p^k = \frac{|P|}{COP} m^k \tag{C.1}$$

where $p^k \in \mathbf{R}$ is the electric power demand (kW) and COP the coefficient of performance.

To determine when to change the setpoints, each refrigerator employs a receding horizon model predictive controller. Every 10 minutes, the controller generates a temperature setpoint schedule for the next 2 hours that minimizes the electricity costs. The refrigerator then implements the first 10 minutes of the 2-hour setpoint schedule. This procedure is repeated every 10 minutes over one day.

Parameter	Refrigerator
Thermal resistance, R (°C/kW)	[80, 100]
Thermal capacitance, C (kWh/°C)	[0.4, 0.8]
Energy transfer rate, P (kW)	[-1, -0.2]
Coefficient of performance, COP	2
Temperature setpoint, T_{set} (°C)	[1.7, 3.3]
Deadband width, δ (°C)	[1, 2]
Ambient temperature, T_a (°C)	21

Table C.1: Refrigerator thermal parameter ranges adopted from [64]

Setpoint changes are applied over 10 consecutive time steps. In other words, the 2-hour horizon is divided into 12 segments of 10 minutes. For each segment, the controller must decide whether to employ the original setpoint temperature or to decrease the setpoint by 1°C. Therefore the controller must consider 2^{12} possible 2-hour setpoint schedules. To find the optimal setpoint schedule, the controller simulates the refrigerator with each of the 2^{12} setpoint schedules and selects the schedule that has the lowest electricity cost.

In this study, we simulate the dynamics of 1000 refrigerators. To generate a diverse population of refrigerators, we employ published model parameter ranges, given in Table C.1 and adopted from [64]. For each refrigerator in the population, we randomly draw the R, P, T_{set}, and δ values from a uniform distribution between the maximum and minimum values shown in the table. These values remained fixed for a specific TCL. To represent periodic loss or gain of thermal capacitance and changes in the unmodeled noise process, we randomly vary the C and θ_3 parameters of each TCL. Once an hour, a new value of C is randomly drawn from the uniform distribution in Table C.1. Once every 3 hours, a new θ_3 is drawn from the uniform distribution [-0.16,-0.08]. Finally, we assume a constant COP of 2 and ambient temperature T_∞ of 21°C.

To quantify the impact of model accuracy on the decision making, we implement 3 model predictive controllers. The first controller, referred to as the "static" controller, employs a model with fixed parameters ($C = 0.6$ and $\theta_3 = -0.12$). The second controller, referred to as the "adaptive" controller, employs parameter estimates produced recursively using the single UKF method. The third controller, referred to as the "informed" controller, has perfect knowledge of the system parameters at the time that each forecast is produced. We employ each of these controllers to simulate the TCLs in the population and to minimize the electricity costs by adjusting the temperature setpoints using the procedure described above.

For each of the 1000 TCLs in the population, we simulate the receding horizon model

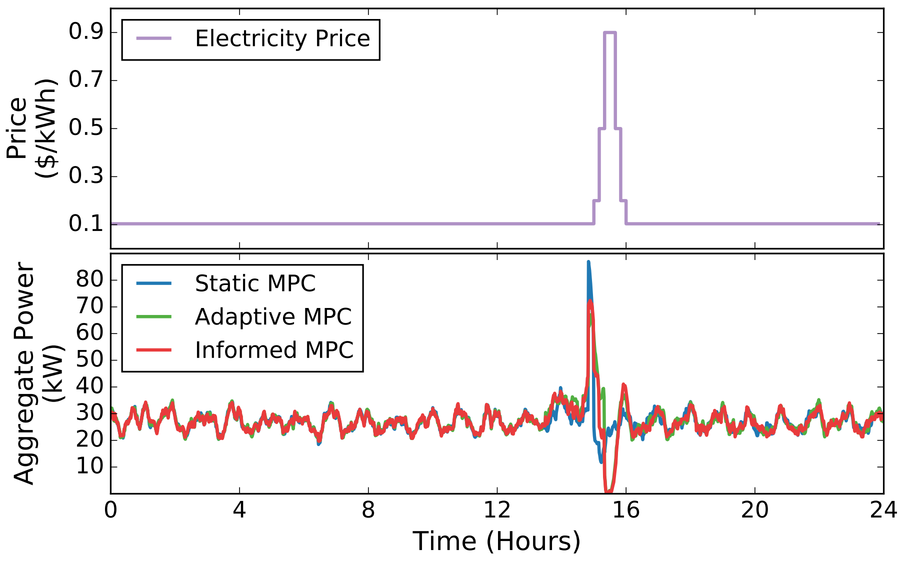

Figure C.3: Demand Response of Simulated TCL Population (Full Day)

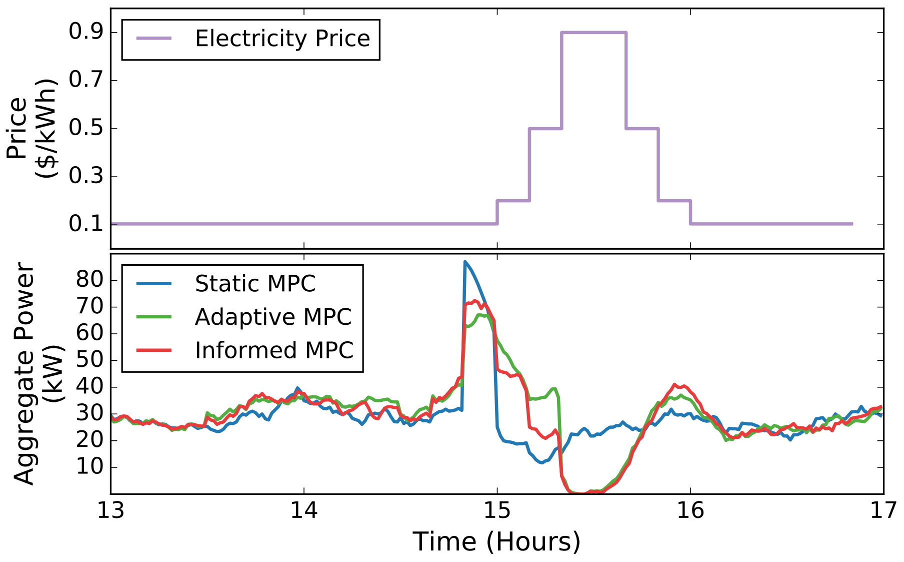

Figure C.4: Demand Response of Simulated TCL Population (Price Event)

predictive control (MPC) over one day using the static, adaptive, and informed controllers. The results of this study are illustrated in Figures C.3 and C.4. The top subplots show the demand response event pricing and the bottom subplots show the aggregate power demand of the refrigerator population. In Figure C.4, we can observe how the controllers respond to the demand response pricing event and how the model accuracy impacts the capability of the controllers to optimally schedule the temperature setpoints. In other words, the static, adaptive, and informed controllers each produce an optimal temperature setpoint schedule using their respective TCL models. However, if the models do not accurately represent the dynamics of the system, the setpoint schedule implemented by the controller is likely to be suboptimal. This is illustrated in Figure C.4 by the fact that the static controller does not avoid the peak pricing as well as the adaptive and informed controllers. The average cost of operating a TCL in the population is \$0.0769 per day using the static controller, \$0.0740 per day using the adaptive controller, and \$0.0725 per day using the informed controller. Based on these results, the adaptive controller achieves a 3.71% cost savings compared to the static controller.

Additionally, the adaptive controller is more capable of achieving the desired demand response behavior of shifting load away from the peak price period. Within the 1-hour event, the average electricity costs of operating a TCL are \$0.01175 per hour, \$0.00855 per hour, and \$0.00702 per hour using the static, adaptive, and informed controllers, respectively. Within the 1-hour event, the adaptive controller achieves a 27.2% cost savings compared to the static controller. By recursively estimating the parameters using the single UKF method, the adaptive controller better enables the TCL to participate in demand response programs and other smart grid applications.

C.3 Conclusions

In this appendix, we presented 2 simulation studies. The first study demonstrates the capability of the single UKF method to quickly converge to new parameter estimates in response to changes in the system dynamics. The second study presents simulation results for a population of refrigerators which optimize their power demand based on a demand response electricity price event. These studies show the advantage of using model predictive control with the single UKF method rather than employing a fixed set of model parameters.

Appendix D

Distributed Optimization of Thermostatically Controlled Loads

D.1 Notation

To simplify equations, we employ the following notation and abbreviations throughout the chapter.

ℓ_1-norm:

$$\|x\|_1 = \sum_{i=1}^{N} |x_i| \tag{D.1}$$

ℓ_2-norm:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2} \tag{D.2}$$

Root Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2} \tag{D.3}$$

Mean:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{D.4}$$

Sum:

$$\sum x_i = \sum_{i=1}^{N} x_i \tag{D.5}$$

Inner product:

$$\langle \lambda, x \rangle = \lambda^T x \tag{D.6}$$

with variable x, $\lambda \in \mathbf{R}^N$.

D.2 Sharing ADMM Optimality and Residuals

In this section, we derive the sharing ADMM residuals, which are required to define the stopping criteria. The necessary and sufficient optimality conditions for the sharing ADMM problem (6.9) are given by the primal feasibility,

$$x_i^* - z_i^* = 0 \tag{D.7}$$

and dual feasibility,

$$0 = \nabla f_i(x^*) + \lambda_i^* \tag{D.8}$$

$$0 = \nabla g(\sum z_i^*) - \sum \lambda_i^* \tag{D.9}$$

for $i = 1, \dots, N$ assuming f_i and g are differentiable.

Since z^{k+1} minimizes (6.9b) by definition, we can show that z^{k+1} and λ^{k+1} always satisfy (D.9),

$$\begin{aligned} 0 &= \nabla g(\sum z_i^{k+1}) - (\sum \lambda_i^k + \sum \rho(x_i^{k+1} - z_i^{k+1})) \\ &= \nabla g(\sum z_i^{k+1}) - \sum (\lambda_i^k + \rho(x_i^{k+1} - z_i^{k+1})) \\ &= \nabla g(\sum z_i^{k+1}) - \sum \lambda_i^{k+1} \end{aligned}$$

Therefore, optimality is achieved by satisfying (D.7) and (D.8). From (D.7), we can define the primal residual as

$$r_i^{k+1} = x_i^{k+1} - z_i^{k+1} \tag{D.10}$$

Since x_i^{k+1} minimizes (6.9a) by definition, we can show

$$\begin{aligned} 0 &= \nabla f_i(x_i^{k+1}) + \lambda_i^k + \rho(x_i^{k+1} - z_i^k) \\ &= \nabla f_i(x_i^{k+1}) + \lambda_i^k + \rho(x_i^{k+1} - z_i^k + z_i^{k+1} - z_i^{k+1}) \\ &= \nabla f_i(x_i^{k+1}) + (\lambda_i^k + \rho(x_i^{k+1} - z_i^{k+1})) + \rho(z_i^{k+1} - z_i^k) \\ &= \nabla f_i(x_i^{k+1}) + \lambda_i^{k+1} + \rho(z_i^{k+1} - z_i^k) \end{aligned}$$

Thus, we can define the dual residual as

$$s_i^{k+1} = \nabla f_i(x_i^{k+1}) + \lambda_i^{k+1} = -\rho(z_i^{k+1} - z_i^k) \tag{D.11}$$

D.3 Averaged Sharing ADMM

In this section, we derive the averaged form of the sharing ADMM algorithm. The sharing ADMM algorithm (6.9) requires the local calculation of a z_i^k, λ_i^k, and r_i^k term for each agent $i = 1, \dots, N$ in the network. Next, we will show that we can simplify the algorithm by introducing global variables $\bar{x}^k$, $\bar{z}^k$, and $\bar{\lambda}^k$ representing the arithmetic mean of all x_i^k, z_i^k, and λ_i^k, respectively.

We begin by introducing $\bar{z}^k$ into the z-update equation (6.9b), which can be rewritten as

$$\begin{aligned}\min_{z,\bar{z}}\ & g(N\bar{z}) \\ & + \sum \left(\langle \lambda_i^k, -z_i\rangle + \frac{\rho}{2}\|x_i^{k+1} - z_i\|_2^2\right) \\ \text{s.t. }\ & \bar{z} = \frac{1}{N}\sum z_i\end{aligned} \tag{D.12}$$

or in augmented Lagrangian form

$$\begin{aligned}\mathscr{L}(z, \bar{z}, \mu) = g(N\bar{z}) &+ \sum \langle \lambda_i^k, -z_i\rangle \\ &+ \sum\left(\frac{\rho}{2}\|x_i^{k+1} - z_i\|_2^2\right) \\ &+ \mu^T\left(\bar{z} - \frac{1}{N}\sum z_i\right)\end{aligned}$$

Thus, for every iteration of the sharing ADMM algorithm, the optimal value of each z_i is

$$\begin{aligned}0 &= \frac{\partial \mathscr{L}}{\partial z_i}(z_i^*, \bar{z}^*, \mu^*) \\ &= \lambda_i^k + \rho(x_i^{k+1} - z_i^*) + \frac{\mu^*}{N} \\ &= \frac{1}{\rho}\left(\lambda_i^k + \frac{\mu^*}{N}\right) + x_i^{k+1} - z_i^* \\ z_i^* &= \frac{\mu^*}{N\rho} + \frac{\lambda_i^k}{\rho} + x_i^{k+1}\end{aligned} \tag{D.13}$$

Finally, we can calculate the optimal value of $\bar{z}$

$$\begin{aligned}\bar{z}^* &= \frac{1}{N}\sum z_i^* \\ &= \frac{1}{N}\sum\left(\frac{\mu^*}{N\rho} + \frac{\lambda_i^k}{\rho} + x_i^{k+1}\right) \\ &= \frac{1}{N}\left(\frac{\mu^*}{\rho} + \frac{1}{\rho}\sum\lambda_i^k + \sum x_i^{k+1}\right) \\ &= \frac{\mu^*}{N\rho} + \frac{\bar{\lambda}^k}{\rho} + \bar{x}^{k+1}\end{aligned} \tag{D.14}$$

Thus, substituting $\mu^*/N\rho$ from (D.14) into (D.13),

$$z_i^* = \bar{z}^* - \frac{\bar{\lambda}^k}{\rho} - \bar{x}^{k+1} + \frac{\lambda_i^k}{\rho} + x_i^{k+1} \tag{D.15}$$

or equivalently

$$z_i^{k+1} = \bar{z}^{k+1} + (x_i^{k+1} - \bar{x}^{k+1}) + \frac{1}{\rho}(\lambda_i^k - \bar{\lambda}^k) \tag{D.16}$$

Next, we can replace z_i^{k+1} in the λ_i-update equation (6.9c)

$$\begin{aligned} \lambda_i^{k+1} &= \lambda_i^k + \rho(x_i^{k+1} - z_i^{k+1}) \\ &= \lambda_i^k \\ &\quad + \rho(x_i^{k+1} - (\bar{z}^{k+1} + x_i^{k+1} - \bar{x}^{k+1})) \qquad \text{(D.17)} \\ &\quad - (\lambda_i^k - \bar{\lambda}^k) \\ &= \bar{\lambda}^k + \rho(\bar{x}^{k+1} - \bar{z}^{k+1}) \end{aligned}$$

which shows that the dual variables λ_i^k are all equal to the global $\bar{\lambda}^k$ and thus

$$z_i^{k+1} = \bar{z}^{k+1} + (x_i^{k+1} - \bar{x}^{k+1}) \tag{D.18}$$

Therefore, the unscaled form of the averaged sharing ADMM algorithm is

$$\begin{aligned} x_i^{k+1} &= \underset{x_i}{\operatorname{argmin}}\ f_i(x_i) + \langle \bar{\lambda}^k, x_i \rangle \qquad \text{(D.19a)} \\ &\quad + \frac{\rho}{2}\|x_i - x_i^k + \bar{x}^k - \bar{z}^k\|_2^2 \\ \bar{z}^{k+1} &= \underset{\bar{z}}{\operatorname{argmin}}\ g(N\bar{z}) + \langle \bar{\lambda}^k, -N\bar{z} \rangle \qquad \text{(D.19b)} \\ &\quad + \frac{N\rho}{2}\|\bar{x}^{k+1} - \bar{z}\|_2^2 \\ \bar{\lambda}^{k+1} &= \bar{\lambda}^k + \rho(\bar{x}^{k+1} - \bar{z}^{k+1}) \qquad \text{(D.19c)} \end{aligned}$$

With this averaged sharing ADMM form, the individual agents no longer update their own λ_i variable. Instead, a single aggregator updates $\bar{\lambda}$, along with $\bar{x}$ and $\bar{z}$, and reports these global variables to every agent in the network.

D.4 Averaged Sharing Residuals

In order to apply the stopping criteria, we must redefine the primal and dual residuals for the averaged form. We can substitute z_i^{k+1} from (D.18) into (D.10) and (D.11) in order to define the primal residual r_i^k and dual residual s_i^k in terms of $\bar{z}$

$$r_i^{k+1} = \bar{x}^{k+1} - \bar{z}^{k+1} \tag{D.20}$$

$$\begin{aligned} s_i^{k+1} = \rho((&\bar{x}^{k+1} - \bar{x}^k) - (x_i^{k+1} - x_i^k) \\ &- (\bar{z}^{k+1} - \bar{z}^k)) \end{aligned} \tag{D.21}$$

and the corresponding ℓ_2-norms of the stopping criteria

$$\begin{aligned} \|r^k\|_2 &= N\|\bar{x}^k - \bar{z}^k\|_2 \\ \|s^k\|_2 &= \textstyle\sum \|s_i^k\|_2 \end{aligned} \tag{D.22}$$

Bibliography

[1] Clarence Agbi, Zhen Song, and Bruce Krogh. “Parameter identifiability for multi-zone building models”. In: *Decision and Control (CDC), 51st Annual Conference on.* IEEE. 2012, pp. 6951–6956.

[2] Charu C. Aggarwal. “Outlier ensembles”. In: *ACM SIGKDD Explorations Newsletter* 14.2 (2012), pp. 49–58.

[3] H. K. Alfares and M. Nazeeruddin. “Electric load forecasting: literature survey and classification of methods”. In: *International Journal of Systems Science* 33.1 (2002), pp. 23–34. DOI: 10.1080/00207720110067421.

[4] David Angeli and Panagiotis-Aristidis Kountouriotis. “A stochastic approach to dynamic-demand refrigerator control”. In: *Control Systems Technology, IEEE Transactions on* 20.3 (2012), pp. 581–592.

[5] Anil Aswani et al. “Energy-efficient building HVAC control using hybrid system LBMPC”. In: *arXiv preprint arXiv:1204.4717* (2012).

[6] Anil Aswani et al. “Identifying models of HVAC systems using semiparametric regression”. In: *American Control Conference (ACC).* IEEE. 2012, pp. 3675–3680.

[7] A. Azadeh et al. “Integration of artificial neural networks and genetic algorithm to predict electrical energy consumption”. In: *Applied Mathematics and Computation* 186.2 (2007), pp. 1731–1741. DOI: 10.1016/j.amc.2006.08.093.

[8] Saeid Bashash and Hosam K Fathy. “Modeling and control insights into demand-side energy management through setpoint control of thermostatic loads”. In: *American Control Conference (ACC).* IEEE. 2011, pp. 4546–4553.

[9] Stephen Boyd et al. “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers”. In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[10] Leo Breiman. “Stacked regressions”. In: *Machine learning* 24.1 (1996), pp. 49–64.

[11] Rich Brown. “US building-sector energy efficiency potential”. In: *Lawrence Berkeley National Laboratory* (2008).

[12] Eric M. Burger and Scott J. Moura. "Gated Ensemble Learning Method for Demand-Side Electricity Load Forecasting". In: *Energy and Buildings* 109 (2015), pp. 23–34. DOI: `10.1016/j.enbuild.2015.10.019`.

[13] Eric M. Burger and Scott J. Moura. "Generation Following with Thermostatically Controlled Loads via Alternating Direction Method of Multipliers Sharing Algorithm". In: *Electric Power Systems Research* 146 (2017), pp. 141–160. DOI: `10.1016/j.epsr.2016.12.001`. URL: `http://escholarship.org/uc/item/2m5333xx`.

[14] Eric M. Burger and Scott J. Moura. "Recursive Parameter Estimation of Thermostatically Controlled Loads via Unscented Kalman Filter". In: *Sustainable Energy, Grids and Networks* 8 (2016), pp. 12–25. DOI: `10.1016/j.segan.2016.09.001`. URL: `http://escholarship.org/uc/item/7t453713`.

[15] Eric M. Burger, Hector E. Perez, and Scott J. Moura. "Piecewise Linear Thermal Model and Recursive Parameter Estimation of a Residential Heating System". 2015. URL: `http://escholarship.org/uc/item/8kx450mg`.

[16] *California Independent System Operator*. URL: `oasis.caiso.com`.

[17] Duncan S. Callaway. "Tapping the energy storage potential in electric loads to deliver load following and regulation, with application to wind energy". In: *Energy Conversion and Management* 50.5 (2009), pp. 1389–1400.

[18] Duncan S. Callaway, Ian Hiskens, et al. "Achieving controllability of electric loads". In: *Proceedings of the IEEE* 99.1 (2011), pp. 184–199.

[19] E. J. Cands, M. Wakin, and S. Boyd. "Enhancing sparsity by reweighted L1 minimization". In: *Journal of Fourier Analysis and Applications* 14.5 (2008), pp. 877–905.

[20] C. Y. Chong and A. Debs. "Statistical synthesis of power system functional load models". In: *Decision and Control (CDC) including the Symposium on Adaptive Processes, 18th Conference on*. 18. IEEE. 1979, pp. 264–269.

[21] G. A. Darbellay and M. Slama. "Forecasting the short-term demand for electricity - Do neural networks stand a better chance?" In: *International Journal of Forecasting* 16.1 (2000), pp. 71–83. DOI: `10.1016/S0169-2070(99)00045-X`.

[22] Judith E. Dayhoff and James M. DeLeo. "Artificial neural networks". In: *Cancer* 91.S8 (2001), pp. 1615–1635.

[23] Glenn De'ath and Katharina E. Fabricius. "Classification and regression trees: a powerful yet simple technique for ecological data analysis". In: *Ecology* 81.11 (2000), pp. 3178–3192.

[24] Saso Dzeroski and Bernard Zenko. "Is Combining Classifiers with Stacking Better than Selecting the Best One?" In: *Machine Learning* 54.3 (2004), pp. 255–273.

[25] J. H. Eto et al. “Demand Response Spinning Reserve Demonstration. Lawrence Berkeley National Laboratory, Berkeley, CA. Prepared for Energy Systems Integration”. In: *Public Interest Energy Research Program, California Energy Commission* (2007).

[26] Azad Ghaffari, Scott Moura, and Miroslav Krstić. “Modeling, Control, and Stability Analysis of Heterogeneous Thermostatically Controlled Load Populations Using Partial Differential Equations”. In: *Journal of Dynamic Systems, Measurement, and Control* 137.10 (2015), p. 101009.

[27] Tom Goldstein et al. “Fast Alternating Direction Optimization Methods”. In: *UCLA Computational and Applied Mathematics Report* (2012), pp. 12–35.

[28] *Google App Engine.* URL: `appengine.google.com`.

[29] Siddharth Goyal and Prabir Barooah. “A method for model-reduction of non-linear thermal dynamics of multi-zone buildings”. In: *Energy and Buildings* 47 (2012), pp. 332–340.

[30] ASHRAE Handbook-Fundamentals. “American society of heating, refrigerating and air-conditioning engineers”. In: *Inc., NE Atlanta, GA* 30329 (2009).

[31] He Hao et al. “Aggregate flexibility of thermostatically controlled loads”. In: *Power Systems, IEEE Transactions on* 30.1 (2015), pp. 189–198.

[32] *Heroku App Development Platform.* URL: `heroku.com`.

[33] H. S. Hippert, C. E. Pedreira, and R. C. Souza. “Neural networks for short-term load forecasting: A review and evaluation”. In: *Power Systems, IEEE Transactions on* 16.1 (2001), pp. 44–55. DOI: `10.1109/59.910780`.

[34] E. Hirst and B. Kirby. “Separating and measuring the regulation and load-following ancillary services”. In: *Utilities Policy* 8 (1999), pp. 75–81.

[35] Wei-Chiang Hong. “Electric load forecasting by support vector model”. In: *Applied Mathematical Modelling* 33.5 (2009), pp. 2444–2454. DOI: `10.1016/j.apm.2008.07.010`.

[36] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95.

[37] Sandro Iacovella et al. “Cluster Control of Heterogeneous Thermostatically Controlled Loads Using Tracer Devices”. In: *Smart Grid, IEEE Transactions on* (2015), pp. 1–9.

[38] S. Ihara and F. C. Schweppe. “Physically based modeling of cold load pickup”. In: *Power Apparatus and Systems, IEEE Transactions on* 100.9 (1981), pp. 4142–4250.

[39] P.A. Ioannou and J. Sun. *Robust Adaptive Control.* Prentice-Hall, 1996.

[40] Fahad Javeda et al. “Forecasting for demand response in smart grids: An analysis on use of anthropologic and structural data and short term multiple loads forecasting”. In: *Applied Energy* 96 (2012), pp. 150–160.

[41] Jorjeta G. Jetcheva, Mostafa Majidpour, and Wei-Peng Chen. "Neural Network Model Ensembles for Building-level Electricity Load Forecasts". In: *Energy and Buildings* 84 (2014), pp. 214–223. DOI: `10.1016/j.enbuild.2014.08.004`.

[42] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *AeroSense*. International Society for Optics and Photonics. 1997, pp. 182–193.

[43] S. Karatasou, M. Santamouris, and V. Geros. "Modeling and predicting building's energy use with artificial neural networks: Methods and results". In: *Energy and Buildings* 38.8 (2006), pp. 949–958.

[44] Abdollah Kavousi-Fard, Haidar Samet, and Fatemeh Marzbani. "A new hybrid Modified Firefly Algorithm and Support Vector Regression model for accurate Short Term Load Forecasting". In: *Expert Systems with Applications* 41.13 (2014), pp. 6047–6056. DOI: `10.1016/j.eswa.2014.03.053`.

[45] Brendan J. Kirby. "Frequency Regulation Basics and Trends". In: *ORNL/TM 2004/291, Oak Ridge National Laboratory* (2004).

[46] Stephan Koch, Johanna L. Mathieu, and Duncan S. Callaway. "Modeling and control of aggregated heterogeneous thermostatically controlled loads for ancillary services". In: *Proc. PSCC*. 2011, pp. 1–7.

[47] Matt Kraning et al. "Message passing for dynamic network energy management". In: *arXiv preprint arXiv:1204.1106* (2012).

[48] S. Kundu et al. "Modeling and control of thermostatically controlled loads". In: *Proceedings of the 17th Power Systems Computation Conference*. Stockholm, Sweden, 2011.

[49] Caroline Le Floch, Francois Belletti, and Scott Moura. "Optimal Charging of Electric Vehicles for Load Shaping: A Dual-Splitting Framework With Explicit Convergence Bounds". In: *Transportation Electrification, IEEE Transactions on* 2.2 (2016), pp. 190–199.

[50] Caroline Le Floch, Florent Di Meglio, and Scott Moura. "Optimal charging of vehicle-to-grid fleets via PDE aggregation techniques". In: *American Control Conference (ACC)*. IEEE. 2015, pp. 3285–3291.

[51] Caroline Le Floch et al. "Distributed optimal charging of electric vehicles for demand response and load shaping". In: *Decision and Control (CDC), 54th Annual Conference on*. IEEE. 2015, pp. 6570–6576.

[52] Kangji Li, Hongye Su, and Jian Chu. "Forecasting building energy consumption using neural networks and hybrid neuro-fuzzy system: A comparative study". In: *Energy and Buildings* 43 (2011), pp. 2893–2899.

[53] Yashen Lin, Timothy Middelkoop, and Prabir Barooah. "Issues in identification of control-oriented thermal models of zones in multi-zone buildings". In: *Decision and Control (CDC), 51st Annual Conference on*. IEEE. 2012, pp. 6932–6937.

[54] Mingxi Liu and Yang Shi. "Distributed model predictive control of thermostatically controlled appliances for providing balancing service". In: *Decision and Control (CDC), 53rd Annual Conference on*. 2014, pp. 4850–4855.

[55] Chi-Jie Lu, Tian-Shyug Lee, and Chih-Chou Chiu. "Financial time series forecasting using independent component analysis and support vector regression". In: *Decision Support Systems* 47.2 (2009), pp. 115–125.

[56] Y. Ma et al. "Predictive Control for Energy Efficient Buildings with Thermal Storage: Modeling, Simulation, and Experiments". In: *IEEE Control Systems Magazine* 31.1 (2012), pp. 44–64.

[57] M. Maasoumy et al. "Online Simultaneous State Estimation and Parameter Adaptation for Building Predictive Control". In: *ASME Dynamic Systems and Control Conference (DSCC)*. Palo Alto, California, 2013.

[58] Mehdi Maasoumy and Alberto Sangiovanni-Vincentelli. "Total and peak energy consumption minimization of building HVAC systems using model predictive control". In: *IEEE Design and Test of Computers* 29.4 (2012), pp. 26 –35. DOI: 10.1109/MDT.2012.2200871.

[59] Mehdi Maasoumy et al. "Model Predictive Control Approach to Online Computation of Demand-Side Flexibility of Commercial Buildings HVAC Systems for Supply Following". In: *American Control Conference (ACC)*. Portland, Oregon, 2014, pp. 1082–1089.

[60] Yuri V Makarov et al. "Operational impacts of wind generation on California power systems". In: *Power Systems, IEEE Transactions on* 24.2 (2009), pp. 1039–1050.

[61] Roland Malhame and Chee-Yee Chong. "Electric load model synthesis by diffusion approximation of a high-order hybrid-state stochastic system". In: *Automatic Control, IEEE Transactions on* 30.9 (1985), pp. 854–860.

[62] Joaquim Massana et al. "Short-term load forecasting in a non-residential building contrasting models and attributes". In: *Energy and Buildings* 92 (2015), pp. 322–330. DOI: 10.1016/j.enbuild.2015.02.007.

[63] Johanna L. Mathieu and Duncan S. Callaway. "State estimation and control of heterogeneous thermostatically controlled loads for load following". In: *System Science (HICSS), 45th Hawaii International Conference on*. IEEE. 2012, pp. 2002–2011.

[64] Johanna L. Mathieu, Mark Dyson, and Duncan S. Callaway. "Using Residential Electric Loads for Fast Demand Response: The Potential Resource and Revenues, the Costs, and Policy Recommendations". In: *ACEEE Summer Study on Energy Efficiency in Buildings* (2012).

[65] Johanna L. Mathieu, Stephan Koch, and Duncan S. Callaway. "State estimation and control of electric loads to manage real-time energy imbalance". In: *Power Systems, IEEE Transactions on* 28.1 (2013), pp. 430–440.

[66] Johanna L. Mathieu et al. "Arbitraging Intraday Wholesale Energy Market Prices With Aggregations of Thermostatic Loads". In: *Power Systems, IEEE Transactions on* 30.2 (2015), pp. 763–772.

[67] Johanna L. Mathieu et al. "Energy arbitrage with thermostatically controlled loads". In: *European Control Conference (ECC)*. IEEE. 2013, pp. 2519–2526.

[68] K. Metaxiotis et al. "Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher". In: *Energy Conversion and Management* 44.9 (2003), pp. 1525–1534. DOI: `10.1016/S0196-8904(02)00148-6`.

[69] R. E. Mortensen and K. P. Haggerty. "A stochastic computer model for heating and cooling loads." In: *Power Systems, IEEE Transactions on* 3.3 (1998), pp. 1213–1219.

[70] Scott J. Moura, Victor Ruiz, and Jan Bendsten. "Modeling heterogeneous populations of thermostatically controlled loads using diffusion-advection PDEs". In: *Dynamic Systems and Control Conference (DSCC)*. American Society of Mechanical Engineers. 2013.

[71] Guy R. Newsham and Benjamin J. Birt. "Building-level occupancy data to improve ARIMA-based electricity use forecasts". In: *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*. ACM. 2010, pp. 13–18.

[72] Hiroyasu Okuyama and Yoshinori Onishi. "System parameter identification theory and uncertainty analysis methods for multi-zone building heat transfer and infiltration". In: *Building and Environment* 54 (2012), pp. 39–52.

[73] Stephen Malvern Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[74] Pacific Gas & Electric. *Residential Electricity Rates (MAR 1 - APR 30, 2014)*. Accessed May. 2, 2014. URL: `http://www.pge.com/tariffs/electric.shtml`.

[75] Anil Pahwa and CW Brice III. "Modeling and system identification of residential air conditioning load". In: *Power Apparatus and Systems, IEEE Transactions on* 6 (1985), pp. 1418–1425.

[76] P. F. Pai and W. C. Hong. "Support vector machines with simulated annealing algorithms in electricity load forecasting". In: *Energy Conversion and Management* 46.17 (2005), pp. 2669–2688. DOI: `10.1016/j.enconman.2005.02.004`.

[77] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[78] Luis Pérez-Lombard, José Ortiz, and Christine Pout. "A review on buildings energy consumption information". In: *Energy and Buildings* 40.3 (2008), pp. 394–398.

[79] Cristian Perfumo et al. “Load management: Model-based control of aggregate power for populations of thermostatically controlled loads”. English. In: *Energy Conversion and Management* 55 (2012), pp. 36 –48. ISSN: 01968904. DOI: 10.1016/j.enconman.2011.10.019.

[80] Peter Radecki and Brandon Hencey. “Online building thermal parameter estimation via unscented Kalman filtering”. In: *American Control Conference (ACC).* IEEE. 2012, pp. 3056–3062.

[81] S. Rahman and O. Hazim. “A generalized knowledge-based short-term load-forecasting technique”. In: *Power Systems, IEEE Transactions on* 8.2 (1993), pp. 508–514. DOI: 10.1109/59.260833.

[82] Amir Safdarian et al. “Domestic EWH and HVAC management in smart grids: Potential benefits and realization”. In: *Electric Power Systems Research* 134 (2016), pp. 38–46.

[83] Tom Schaul et al. “PyBrain: an open source machine-learning library written in Python”. In: *Journal of Machine Learning Research* 11 (2010), pp. 743–746.

[84] Joe A. Short, David G. Infield, and Leon L. Freris. “Stabilization of grid frequency through dynamic demand control”. In: *Power Systems, IEEE Transactions on* 22.3 (2007), pp. 1284–1293.

[85] Alex J. Smola and Bernhard Schölkopf. “A tutorial on support vector regression”. In: *Statistics and Computing* 14.3 (2004), pp. 199–222.

[86] Richard L. Snyder. “California irrigation management information system”. In: *American Journal of Potato Research* 61.4 (1984), pp. 229–234. URL: http://www.cimis.water.ca.gov/.

[87] G. Strbac. “Demand side management: Benefits and challenges”. In: *Energy Policy* 36.12 (2008), pp. 4419–4426.

[88] J. W. Taylor, L. M. de Menezes, and P. E. McSharry. “A comparison of univariate methods for forecasting electricity demand up to a day ahead”. In: *International Journal of Forecasting* 22.1 (2006), pp. 1–16. DOI: 10.1016/j.ijforecast.2005.06.006.

[89] Simon H. Tindemans, Vincenzo Trovato, and Goran Strbac. “Decentralized Control of Thermostatic Loads for Flexible Demand Response”. In: *Control Systems Technology, IEEE Transactions on* 23.5 (2015), pp. 1685–1700.

[90] Luminita Cristiana Totu, John Leth, and Rafael Wisniewski. “Control for large scale demand response of thermostatic loads*”. In: *American Control Conference (ACC).* IEEE. 2013, pp. 5023–5028.

[91] Luminita Cristiana Totu, Rafael Wisniewski, and John Leth. “Modeling populations of thermostatic loads with switching rate actuation”. In: *arXiv preprint arXiv:1411.2864* (2014).

[92] Geoffrey K. F. Tso and Kelvin K. W. Yau. "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks". In: *Energy* 32.9 (2007), pp. 1761–1768. DOI: `10.1016/j.energy.2006.11.010`.

[93] Canbolat Ucak and Ramazan Çağlar. "The effects of load parameter dispersion and direct load control actions on aggregated load". In: *Power System Technology (POWERCON), International Conference on.* Vol. 1. IEEE. 1998, pp. 280–284.

[94] U.S. Department of Energy. *2010 Buildings Energy Data Book.* Accessed May. 2, 2014. URL: `http://buildingsdatabook.eren.doe.gov`.

[95] U.S. Department of Energy. *Electric Resistance Heating.* Accessed May. 2, 2014. 2012. URL: `http://energy.gov/energysaver/articles/electric-resistance-heating`.

[96] U.S. Energy Information Administration. *Monthly Energy Review.* U.S. Department of Energy. 2015. URL: `http://www.eia.gov/mer`.

[97] Evangelos Vrettos and Goran Andersson. "Combined load frequency control and active distribution network management with thermostatically controlled loads". In: *Smart Grid Communications (SmartGridComm), International Conference on.* IEEE. 2013, pp. 247–252.

[98] Eric A. Wan and Rudolph Van Der Merwe. "The Unscented Kalman Filter". In: *Kalman Filtering and Neural Networks.* Wiley, 2001, pp. 221–280.

[99] Eric A. Wan and Rudolph Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC).* IEEE. 2000, pp. 153–158.

[100] Eric A. Wan, Rudolph Van Der Merwe, and Alex T Nelson. "Dual Estimation and the Unscented Transformation". In: *NIPS.* Citeseer. 1999, pp. 666–672.

[101] *Weather Underground Web Service and API.* URL: `wunderground.com/weather/api/`.

[102] J. Wen and T. F. Smith. "Development and Validation of Online Parameter Estimation for HVAC Systems". In: *Journal of Solar Energy Engineering* 125.3 (2003), pp. 324–330.

[103] Michael Wetter. "Modelica-based modelling and simulation to support research and development in building energy and control systems". In: *Journal of Building Performance Simulation* 2.2 (2009), pp. 143–161.

[104] David H. Wolpert. "Stacked generalization". In: *Neural Networks* 5.2 (1992), pp. 241–259.

[105] Hao Xing et al. "Fast distributed power regulation method via networked thermostatically controlled loads". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 5439–5444.

[106] *Xively Web Service and API*. URL: `xively.com`.

[107] Zhao Xu et al. “Design and modelling of thermostatically controlled loads as frequency controlled reserve”. In: *Power Engineering Society General Meeting*. IEEE. 2007, pp. 1–6.

[108] H. T. Yang, C. M. Huang, and C. L. Huang. “Identification of ARMAX model for short term load forecasting: An evolutionary programming approach”. In: *Power Systems, IEEE Transactions on* 11.1 (1996), pp. 403–408.

[109] Wei Zhang et al. “Aggregate model for heterogeneous thermostatically controlled loads with demand response”. In: *Power and Energy Society General Meeting*. IEEE. 2012, pp. 1–8.

[110] Wei Zhang et al. “Aggregated modeling and control of air conditioning loads for demand response”. In: *Power Systems, IEEE Transactions on* 28.4 (2013), pp. 4655–4664.

[111] Wei Zhang et al. “Reduced-order modeling of aggregated thermostatic loads with demand response”. In: *Decision and Control (CDC), 51st Annual Conference on*. 2012, pp. 5592–5597.

[112] *ZigBee Protocol*. URL: `zigbee.org/Specifications/ZigBee/Overview.aspx`.