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

Gregory S. Ledva, *Student Member, IEEE,* Laura Balzano, *Member, IEEE,*
and Johanna L. Mathieu, *Member, IEEE*

***Abstract*—Though distribution system operators have been adding more sensors to their networks, they still often lack an accurate real-time picture of the behavior of distributed energy resources such as demand responsive electric loads and residential solar generation. Such information could improve system reliability, economic efficiency, and environmental impact. Rather than installing additional, costly sensing and communication infrastructure to obtain additional real-time information, it may be possible to use existing sensing capabilities and leverage knowledge about the system to reduce the need for new infrastructure. In this paper, we disaggregate a distribution feeder's demand measurements into: 1) the demand of a population of air conditioners, and 2) the demand of the remaining loads connected to the feeder. We use an online learning algorithm, Dynamic Fixed Share (DFS), that uses the real-time distribution feeder measurements as well as models generated from historical building- and device-level data. We develop two implementations of the algorithm and conduct case studies using real demand data from households and commercial buildings to investigate the effectiveness of the algorithm. The case studies demonstrate that DFS can effectively perform online disaggregation and the choice and construction of models included in the algorithm affects its accuracy, which is comparable to that of a set of Kalman filters.**

***Index Terms*—Online learning, machine learning, energy disaggregation, output feedback, real-time filtering**

I. Introduction

DISTRIBUTED energy resources (DERs) such as demand responsive electric loads and residential solar generation are becoming more common within electricity distribution networks [1], [2]. Sensing infrastructure, such as household smart meters, are also becoming more common [3]. However, distribution system operators still often lack an accurate real-time picture of overall DER characteristics such as i) the total power consumption of the air conditioners connected to a distribution feeder, or ii) the total power production of all solar panels installed on a distribution feeder.

Perfect real-time knowledge of DER characteristics requires a sensor at each of the large number (e.g., thousands) of spatially distributed devices and a communication infrastructure capable of reliably transmitting the data at the necessary frequency (e.g., every few seconds). Rather than installing additional, costly metering and communication infrastructure, in this paper, we show that it is possible to estimate real-time DER characteristics using existing sensing capabilities and some knowledge of the underlying system. Specifically, we show how to separate measurements of the net demand served by a distribution feeder into its components in real-time, using knowledge of the physical processes driving load/generation. We refer to this task as *feeder-level energy disaggregation*.

The authors are with the Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: gsledv@umich.edu; girasole@umich.edu; jlmath@umich.edu). This research was funded by NSF Grant #ECCS-1508943.

Real-time, feeder-level energy disaggregation can help system operators, utilities, and demand response providers improve power system reliability, economic efficiency, and environmental impact. For example, a system operator can 1) estimate the real-time balancing reserve requirement from its estimate of the production of distributed generation resources; 2) estimate the real-time potential for fault induced delayed voltage recovery (FIDVR) caused by stalling in small motor loads [4] from its estimate of the motor load consumption; and 3) optimize conservation voltage recovery (CVR) strategies using its estimate of the mix of constant impedance, constant power, and constant current loads [5]. A utility can 4) better plan demand response actions by knowing the weather forecast and the real-time portion of weather-dependent loads (e.g., air conditioners, heaters, dehumidifiers). A demand response provider can 5) optimize capacity bids into ancillary services markets using its estimate of the real-time, aggregate, demand-responsive load; and 6) use its estimate of the real-time, aggregate, demand-responsive loads as a feedback signal in load coordination algorithms, e.g., [6], [7]. Note that the consumption of demand-responsive loads often needs to be measured for auditing purposes, but the consumption data need not be communicated in real-time.

In this paper, we develop the feeder-level energy disaggregation problem framework and apply an online learning algorithm to separate the active power demand served by a feeder into the active power demand of a population of residential air conditioners and the active power demand of all other loads connected to the feeder. The algorithm [8] incorporates dynamical system models of arbitrary forms, blending aspects of machine learning and state estimation. Building upon our preliminary work [9], the contributions of this paper are to i) frame the feeder-level energy disaggregation problem, ii) adapt the machine learning algorithm in [8] to the feeder-level energy disaggregation problem, iii) develop a variation of the machine learning algorithm that allows it to include models with different underlying states, iv) demonstrate the performance of the online learning algorithm via a realistic data-driven case study, and v) compare the performance of the algorithm to that of a set of Kalman filters. Beyond [9], this paper develops a modified version of the algorithm, compares this modified implementation to a direct implementation of the algorithm, uses only real data (rather than models) to construct the feeder active power signal, uses real data to identify all load models, and compares algorithm performance to that of an aggressive benchmark as opposed to a simple prediction model.

Section II compares our problem and approach to related problems/work. Section III defines the problem framework. Section IV describes the data used to construct the underlying system, and Section V describes the models used within the algorithm. Section VI summarizes the online learning algorithm and

our implementations for the feeder-level energy disaggregation problem. Section VII constructs case studies and summarizes their results. Finally, Section VIII presents the conclusions.

II. Comparison to Related Problems and Work

The feeder-level energy disaggregation problem combines aspects of building-level energy disaggregation and load forecasting. Building-level energy disaggregation, also referred to as nonintrusive load monitoring [10], separates building-level demand measurements into estimates of the demand of individual or small groups of devices [11]. Disaggregation algorithms use data sampled at frequencies ranging from over 1 MHz to 0.3 mHz (i.e., hourly interval data) where higher-frequency data allows separation of more devices [11]. The problem is not usually solved online because the goal is long-term energy efficiency decisions such as identification and replacement of faulty appliances and/or load research. Both unsupervised and supervised learning approaches have been proposed, with the latter often using models developed with submetering data.

Load forecasting predicts the total future demand within a given area over time horizons ranging from hours to years [12]. Whereas energy disaggregation typically deals with small load aggregations, load forecasting typically deals with large aggregations, e.g., thousands to millions of loads. For example, forecasting the load served by a distribution transformer is considered a "small" forecasting problem [12]. Very short term load forecasting, corresponding to intraday forecasts, generally uses 15 min to one hour interval data [12], [13]. Smart meter data enables offline development of detailed load models [14], which may be used online for operational decisions [15], e.g., for predicting the curtailable load [14]. However, load forecasting is typically done offline and is typically used for planning.

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

Machine learning algorithms have been proposed to address a number of problems in power systems including security assessment, forecasting, and optimal operation [16]. A variety of machine learning techniques have been used to forecast load, renewable generation, and prices [17]–[21]. References [22]–[26] apply learning approaches to demand response. However, to the best of our knowledge, this is the first paper to pose and solve the feeder-level energy disaggregation problem, or to apply the approach in [8] to a power systems problem.

III. Problem Framework

We assume that a power system entity (e.g., a system operator, utility, or third-party company) has access to real-time

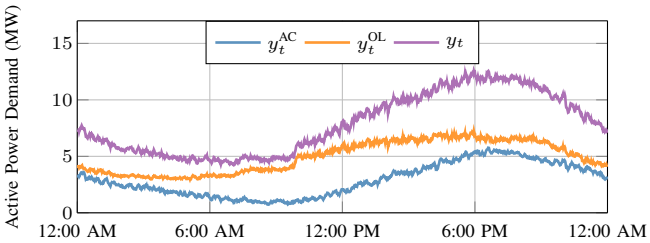
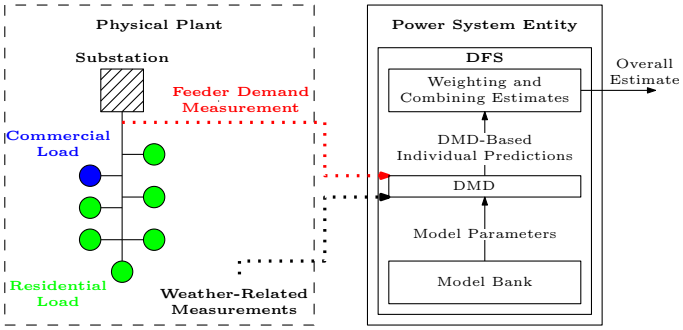


Fig. 1. Example time series of y_t and its components y_t^{OL} and y_t^{AC}.

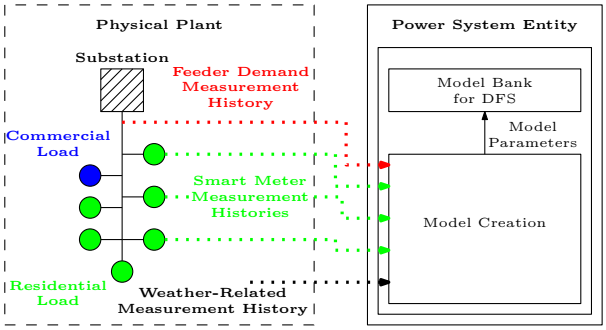
measurements of the electricity demand served by a distribution feeder. The power system entity is interested in separating these measurements into two components in real-time, i.e., at each time-step. The first component is the power demand of a population of residential air conditioners served by the feeder, referred to as the "AC demand." Air conditioners generally draw power periodically to maintain a building's indoor temperature within a range centered at a user-defined temperature set-point. The AC demand varies in time due to each air conditioner's power cycling, weather-related influences, and building occupant influences. The second component is the power demand of the other loads on the feeder, referred to as the "OL demand," which we assume includes both residential and commercial loads. Figure 1 displays example time series for the measured total demand y_t, the AC demand y_t^{AC}, and the OL demand y_t^{OL} over a day. We measure y_t at each time-step and try to estimate y_t^{AC} and y_t^{OL} at each time-step as each measurement arrives.

The power system entity has two distinct modes of operation. The first is the real-time estimation mode depicted in Fig. 2a. The second is the offline model generation mode, depicted in Fig. 2b. During real-time operation, we assume that the power system entity has access to active power measurements corresponding to the demand served by the distribution feeder as well as weather-related measurements. The power measurements are time-averaged active power demands over one min intervals, and they are the sum of the AC and OL demand. The weather-related measurements could include, for example, temperature and humidity, and can be obtained from existing weather sensors; load-specific weather monitoring is not required.

Model generation occurs offline using historical smart meter, feeder, and weather data. To apply DFS to feeder-level energy disaggregation we assume real-time measurements of the demand components are unavailable, but models of the components are available. These models could be created using a variety of techniques (e.g., via system identification using historical measurements obtained from the same system or a different system, or using analytical methods and parameters from the literature). In this work, we assume that smart meters are installed at all houses, and they enable the collection of household-level demand measurements at one minute intervals. The smart meters' communication limitations [11] make real-time communication of this information infeasible, and so we assume that it is only available offline for prior days. Because we do not need real-time, device-level demand measurements, we assume that historical device-level demand estimates can be obtained offline from the historical household-level measurements either by applying non-intrusive load monitoring (NILM) algorithms or by using information from communicating or advanced thermostats, which are becoming more common



(a) Real-time estimation mode



(b) Offline model generation mode

Fig. 2. Problem framework: real-time and offline modes.

within residences. These thermostats can measure and record the on/off mode of a residence's AC unit and measurement histories can be used to estimate the power draw of these devices. The resulting device-level demand estimates may not be exact, but they are accurate enough to be used within the computation of model parameters. As a result, we use historical, device-level measurements to construct the AC demand models. We also assume that the power system entity has access to historical feeder and weather data. Once the models are formed, they are used along with the real-time measurements to estimate the AC and OL demand.

An online learning algorithm, Dynamic Mirror Descent (DMD) [8], uses a single model to generate predictions of the total demand, a loss function to penalize errors between the predicted and measured total demand, and a convex optimization formulation to adjust this prediction based on the measured total demand. Dynamic Fixed Share (DFS) [8], uses DMD within the Fixed Share Algorithm [27] to include predictions from a bank of models. Specifically, DFS applies DMD separately to each model and uses a weighting algorithm to associate a weight with each model's adjusted prediction before combining the predictions into an overall estimate. In DFS, these models are weighted based on their prediction accuracy – better prediction-measurement matching leads to larger weighting and more influence in the overall prediction.

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

IV. Construction of Plant

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

We use feeder model R5-25.00-1 from GridLAB-D's feeder taxonomy [28] to set the average residential and commercial demand on the feeder to 5.8 MW and 2.1 MW, respectively. Ignoring network losses (which, if included, would be treated as part of the OL demand), the total feeder demand measurements are the sum of the AC and OL demand, i.e., $y_t = y_t^{\text{AC}} + y_t^{\text{OL}}$, where y_t^{OL} is the sum of the other residential demand and the commercial demand $y_t^{\text{OL}} = y_t^{\text{OL,res}} + y_t^{\text{OL,com}}$.

Both y_t^{AC} and $y_t^{\text{OL,res}}$ are constructed using residential data from the Pecan Street Dataport [29]. The data consists of historical one min interval household- and device-level demand measurements for a set of single family homes in Texas. Daily household demand signals were randomly drawn with replacement and added together until the total residential signal's mean matched that of the feeder model, resulting in $2,499$ total houses. To construct the AC demand signal, we summed the demand of each household's primary air conditioner and air blower unit. Note that some houses have no/multiple air conditioner and air blower units. We assume that only one unit per household contributes to the AC demand, resulting in $2,269$ units. The remaining demand is the residential OL demand.

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

The plant's weather data is constructed from data obtained from the Pecan Street Dataport [29] and the National Climatic

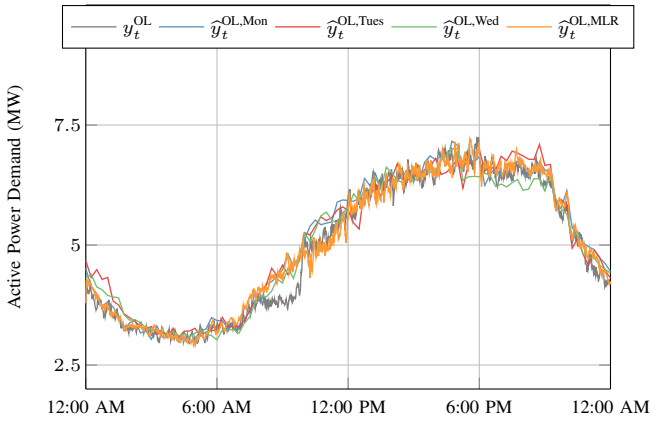


Fig. 3. Example OL demand and several OL demand model predictions.

Data Center [30]. The Pecan Street weather data corresponds to the residential demand. It consists of the outdoor air temperature for Austin, TX, and it is sampled at one hour intervals. We linearly interpolate the data down to one min intervals. The NOAA weather data corresponds to the commercial demand. It consists of outdoor temperature data from the Concord, CA weather station, sampled at one hour intervals. Again, we linearly interpolate the data down to one min intervals. All weather data was taken from the same day as the demand data.

V. System Models

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

A. OL Demand Models

We use two types of regression models to predict the OL demand: time-of-day (TOD) regression models and a multiple linear regression (MLR) model. Figure 3 displays y_t^{OL} for a simulated day, several TOD regression model predictions, e.g., $\widehat{y}_t^{\text{OL,Mon}}$, and the MLR model prediction $\widehat{y}_t^{\text{OL,MLR}}$. We next describe the construction of these models.

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

$$\widehat{y}_t^{\text{OL,TOD}} = \alpha_k^{\text{OL,TOD}} = \alpha^{\text{OL,TOD}}\, x_t^{\text{OL,TOD}}. \tag{1}$$

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

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

$$\widehat{y}_t^{\text{OL,MLR}} = \widehat{y}_t^{\text{OL,res}} + \widehat{y}_t^{\text{OL,com}} \tag{2}$$

$$= \beta^{\text{OL,res}}\, x_t^{\text{OL,res}} + \gamma^{\text{OL,com}}\, x_t^{\text{OL,com}}, \tag{3}$$

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

The MLR model for the residential OL demand uses input features $x_t^{\text{OL,res}} = \left[(x_t^{\text{TOW}})^{\text{T}} \quad T_t^{\text{TX}} \quad y_{t-1}\right]^{\text{T}}$ where x_t^{TOW} is an indicator vector for the time of week in minutes, T_t^{TX} is the outdoor temperature for Austin, TX, and y_{t-1} is the measured total demand of the previous time-step. The commercial regression model corresponds to "Baseline Method 1" from [31]; it uses input features $x_t^{\text{OL,com}} = \left[(x_t^{\text{TOW}})^{\text{T}} \quad T_t^{\text{CA}} \cdot (x_t^{\text{TOW}})^{\text{T}}\right]^{\text{T}}$ where T_t^{CA} is the outdoor temperature for Concord, CA and $T_t^{\text{CA}} \cdot (x_t^{\text{TOW}})$ is a vector that associates the temperature to the corresponding time of week.

B. AC Demand Models

We use three types of models to predict the AC demand: a MLR model, linear time invariant (LTI) system models, and linear time varying (LTV) system models. Figure 4 displays y_t^{AC} for a simulated day, several LTV model predictions, e.g., $\widehat{y}_t^{\text{AC,LTV1}}$, and the MLR regression model prediction $\widehat{y}_t^{\text{AC,MLR}}$. We next describe the construction of these models.

1) MLR Model: The MLR model of the AC demand, denoted $\Phi^{\text{AC,MLR}}$, is similar to the MLR model in Section V-A2 with different input features $x_t^{\text{AC,MLR}} = \left[(x_t^{\text{TOW}})^{\text{T}} \quad T_{t-\tau^{\text{l}}}^{\text{TX}} \quad (T_{t-\tau^{\text{l}}}^{\text{TX}})^2 \quad (T_{t-\tau^{\text{l}}}^{\text{TX}})^3 \quad (T_{t-\tau^{\text{l}}}^{\text{TX}})^4\right]^{\text{T}}$, where $T_{t-\tau^{\text{l}}}^{\text{TX}}$ is the temperature in Austin, TX from τ^{l} time-steps ago and τ^{l} is the time lag that maximizes the cross correlation between the historical AC demand signal and temperature signal (119 min for our plant).

2) LTI Models: We construct a set of LTI models $\mathcal{M}^{\text{LTI}}$, originally developed in [32], [33]. As in [34], each model within

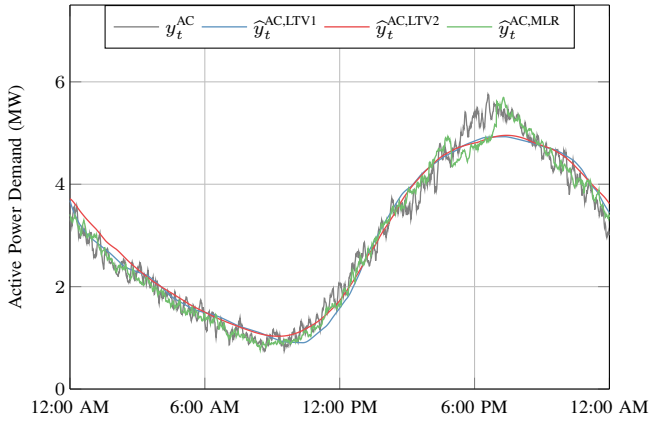


Fig. 4. Examples AC demand and several AC demand model predictions.

the set captures the aggregate behavior of the population of air conditioners at outdoor temperature T^m and has the form

$$\widehat{x}_{t+1}^{\text{LTI},m} = A^{\text{LTI},m}\, \widehat{x}_t^{\text{LTI},m} \tag{4}$$

$$\widehat{y}_t^{\text{AC,LTI},m} = C^{\text{LTI},m}\, \widehat{x}_t^{\text{LTI},m}, \tag{5}$$

with $m \in \mathcal{M}^{\text{LTI}} = \{1, \ldots, N^{\text{LTI}}\}$. The state vector $\widehat{x}_t^{\text{LTI},m} \in \mathbb{R}^{N^{\text{x}} \times 1}$ consists of the portion of the air conditioners within each of N^{x} discrete states. In this paper, we use one state to represent the portion of air conditioners that are drawing power and another to represent those that are not, i.e., $N^{\text{x}} = 2$. The state transition matrix, $A^{\text{LTI},m} \in \mathbb{R}^{N^{\text{x}} \times N^{\text{x}}}$, is a transposed Markov transition matrix. Its entries capture the probabilities that air conditioners maintain their current state or transition to the other state during the time-step. The output matrix $C^{\text{LTI},m}$ estimates the AC demand $\widehat{y}_t^{\text{AC,LTI},m}$ from the portion of air conditioners that are drawing power, i.e., $C^{\text{LTI},m} = N^{\text{ac}} \overline{P}^m \begin{bmatrix} 0 & 1 \end{bmatrix}$, where $\overline{P}^m$ is a parameter approximating of the average power draw of air conditioners drawing power and N^{ac} is the number of air conditioners, which we assume is known.

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

3) LTV Models: We use two LTV models. The first $\Phi^{\text{AC,LTV1}}$ uses the delayed temperature and has the form

$$\widehat{x}_{t+1}^{\text{LTV1}} = A_t^{\text{LTV1}}\, \widehat{x}_t^{\text{LTV1}} \tag{6}$$

$$\widehat{y}_t^{\text{AC,LTV1}} = C_t^{\text{LTV1}}\, \widehat{x}_t^{\text{LTV1}}, \tag{7}$$

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

VI. Online Learning Algorithm

In this section, we first summarize the DFS algorithm developed in [8] and then describe two algorithm implementations, one inspired by DFS and one a direct implementation of it. DFS incorporates DMD, also developed in [8], into the Fixed Share algorithm originally developed in [27]. The Fixed Share algorithm combines a set of predictions that are generated by independent experts, e.g., models, into an estimate of the system parameter using the experts' historical accuracy with respect to observations of the system. DMD extends the traditional online learning framework by incorporating dynamic models, enabling the estimation of time-varying system parameters (or states). DFS uses DMD, applied independently to each of the models, as the experts within the Fixed Share algorithm.

A. The DFS Algorithm

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

DFS uses a set of N^{mdl} models defined as $\mathcal{M}^{\text{mdl}} = \{1, \ldots, N^{\text{mdl}}\}$ to generate the estimate $\widehat{\theta}_t$. To do this, DFS applies the DMD algorithm to each model, forming predictions $\widehat{\theta}_t^m$ for each $m \in \mathcal{M}^{\text{mdl}}$. DMD is executed in two steps (similar to a discrete-time Kalman filter): 1) an observation-based update incorporates the new measurement into the parameter prediction, and 2) a model-based update advances the parameter prediction to the next time-step. DFS then uses the Fixed Share algorithm to form the estimate $\widehat{\theta}_t$ as a weighted combination of the individual model's DMD-based predictions. A weighting algorithm computes the weights based on each model's historical accuracy with respect to the observations y_t. Models that perform poorly have less influence on the overall estimate. The DFS algorithm is [8]

$$\widetilde{\theta}_t^m = \underset{\theta \in \Theta}{\arg\min}\ \eta^{\text{s}} \left\langle \nabla \ell_t(\widehat{\theta}_t^m),\ \theta \right\rangle + D\left(\theta \| \widehat{\theta}_t^m\right) \tag{8}$$

$$\widehat{\theta}_{t+1}^m = \Phi^m(\widetilde{\theta}_t^m) \tag{9}$$

$$w_{t+1}^m = \frac{\lambda}{N^{\text{mdl}}} + (1-\lambda) \frac{w_t^m\, \exp\left(-\eta^r\, \ell_t\left(\widehat{\theta}_t^m\right)\right)}{\sum_{j=1}^{N^{\text{mdl}}} w_t^j\, \exp\left(-\eta^r\, \ell_t\left(\widehat{\theta}_t^j\right)\right)} \tag{10}$$

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

$$\widehat{\theta}_{t+1} = \sum_{m \in \mathcal{M}^{\text{mdl}}} w_{t+1}^m \, \widehat{\theta}_{t+1}^m, \tag{11}$$

where each term is defined below. DMD is applied to each model in (8) and (9) to form the expert predictions, where (8) is a convex program that constructs the measurement-based update to the previous prediction and (9) is the model-based advancement of the adjusted prediction. The Fixed Share algorithm consists of (10) and (11), where (10) computes the weights and (11) computes the estimate as a weighted combination of the individual experts' estimates. We note that the Fixed Share algorithm's updates are independent of the dynamics and only use the experts' predictions and their resulting losses.

In (8), we minimize over the variable θ, $\eta^{\text{s}} > 0$ is a step-size parameter, and $\langle \cdot, \cdot \rangle$ is the standard dot product. The value $\nabla \ell_t(\widehat{\theta}_t)$ is a subgradient of the convex loss function $\ell_t : \Theta \to \mathbb{R}$, which penalizes the error between the predicted and observed values y_t using a known, possibly time-varying, function $h_t : \Theta \to \mathcal{Y}$ that maps θ_t to an observation, i.e., $y_t = h_t(\theta_t)$, to form predictions of the measurements. An example loss function is $\ell_t(\widehat{\theta}_t) = \|C\widehat{\theta}_t - y_t\|_2^2$ where the matrix C is $h_t(\cdot)$. In (9), the function $\Phi^m(\cdot)$ applies model m to advance the adjusted estimate $\widetilde{\theta}_t^m$ in time. Each $\Phi^m(\cdot)$ can have arbitrary form and time-varying parameters. In (10), the weight associated with model m at time-step t is w_t^m, $\lambda \in (0, 1)$ determines the amount of weight that is shared amongst models, and η^r influences switching speed. The weight for model m is based on the loss of each model and the total loss of all models. The term $\eta^{\text{s}} \langle \nabla \ell_t(\widehat{\theta}_t),\ \theta \rangle$ captures the alignment of the variable θ with the positive gradient of $\ell_t(\widehat{\theta}_t)$. To minimize this term alone, we would choose θ to be exactly aligned with the negative gradient direction. The term $D(\theta \| \widehat{\theta}_t)$ is a Bregman divergence that penalizes the deviation between the new variable θ and the old variable $\widehat{\theta}_t$. For simplicity, we have excluded regularization within (8), which DMD readily incorporates [8].

B. Algorithm Implementations

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

1) Update Method 1: The models used within this paper have different underlying parameters, dynamic variables, and/or structures, which makes it difficult to define a common θ_t across all of the models used. Therefore, we develop a variation of the DMD algorithm that adjusts the demand predictions directly, rather than applying the updates to quantities influencing the demand predictions. This allows us to include a diverse set of models. Specifically, we modify the DMD formulation to

$$\widehat{\kappa}_{t+1} = \underset{\theta \in \Theta}{\arg\min}\ \eta^{\text{s}} \left\langle \nabla \ell_t(\widehat{\theta}_t),\ \theta \right\rangle + D\left(\theta \| \widehat{\kappa}_t\right) \tag{12}$$

$$\breve{\theta}_{t+1} = \Phi(\breve{\theta}_t) \tag{13}$$

$$\widehat{\theta}_{t+1} = \breve{\theta}_{t+1} + \widehat{\kappa}_{t+1}. \tag{14}$$

The AC and OL demand models generate their predictions independently from one another, and so (14) can be rewritten as

$$\widehat{\theta}_{t+1} = \Phi(\breve{\theta}_t) + \widehat{\kappa}_{t+1} = \begin{bmatrix} \Phi^{\text{AC}}(\breve{\theta}_t) \\ \Phi^{\text{OL}}(\breve{\theta}_t) \end{bmatrix} + \widehat{\kappa}_{t+1}. \tag{15}$$

The convex program (12) is now used to update a value $\widehat{\kappa}_t$ that accumulates the deviation between the predicted and actual measurements. The model-based update (13) computes an open-loop prediction $\breve{\theta}_{t+1}$, meaning that the measurements do not influence $\breve{\theta}_{t+1}$. The measurement-based updates and model-based, open-loop predictions are combined in (14). In contrast, DMD uses a closed-loop model-based update where the convex program adjusts the parameter estimate to $\widetilde{\theta}_t$, which is used to compute the next parameter estimate $\widehat{\theta}_{t+1}$.

In this method, we define θ_t as the AC and OL demand, i.e., $\theta_t = \begin{bmatrix} y_t^{\text{AC}} & y_t^{\text{OL}} \end{bmatrix}^{\text{T}}$. The mapping from the parameter to the measurement is $h_t(\theta_t) = C_t \theta_t$ where the matrix $C_t = \begin{bmatrix} 1 & 1 \end{bmatrix}$. While the mapping and matrix are time-invariant, they may be time-varying in Section VI-B2, and so we use the more general notation. We choose the loss function as $\ell_t(\widehat{\theta}_t) = \frac{1}{2}\|C_t\widehat{\theta}_t - y_t\|_2^2$ and the divergence as $D(\theta \| \widehat{\kappa}_t) = \frac{1}{2}\|\theta - \widehat{\kappa}_t\|_2^2$. We can then write (12) in closed form as

$$\widehat{\kappa}_{t+1} = \widehat{\kappa}_t + \eta^{\text{s}} C_t^T \left(y_t - C_t \widehat{\theta}_t \right). \tag{16}$$

2) Update Method 2: This method applies only to dynamic system models with dynamic states, i.e., in this paper the LTI or LTV AC demand models, which have dynamic states x_t. We set $\theta_t = \begin{bmatrix} x_t^{\text{T}} & y_t^{\text{OL}} \end{bmatrix}^{\text{T}}$, where x_t is $\widehat{x}_t^{\text{LTI},m}$ in (4), $\widehat{x}_t^{\text{LTV1}}$ in (6), or $\widehat{x}_t^{\text{LTV2}}$ in an update equation similar to (6). The mapping from the parameter to the measurement is then $C_t = \begin{bmatrix} C_t^{\text{AC}} & 1 \end{bmatrix}$ where C_t^{AC} is the output matrix of the LTI or LTV AC demand model, i.e., $C^{\text{LTI},m}$, C_t^{LTV1}, or C_t^{LTV2}. Defining the system parameter in this way allows us to update the dynamic states of the LTI and LTV AC demand models, rather than just the output as in Update Method 1. The model-based update is then

$$\widehat{\theta}_{t+1} = \begin{bmatrix} \Phi^{\text{AC}}(\widetilde{\theta}_t) \\ \Phi^{\text{OL}}(\breve{\theta}_t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \widehat{\kappa}_{t+1}, \tag{17}$$

where we update the AC demand model using the adjusted parameter estimate, as in DMD. Because the OL demand models do not include dynamic states, we continue to update their estimates according to Update Method 1. We again use (16) as the measurement-based update.

VII. Case Studies

In this section, we define the scenarios, describe the benchmark, summarize the parameter settings, and present the results. In [8], performance bounds for DMD and DFS were established in terms of a quantity called regret. Regret is the total (or cumulative) loss of an online learning algorithm's prediction sequence versus that of a comparator sequence, often a best-in-hindsight offline algorithm. In [8], the DMD regret bound uses a comparator that can take on an arbitrary sequence of values from the feasible domain Θ. The DFS regret bound uses a

comparator that chooses the best-in-hindsight possible sequence of models chosen from the same model collection used by DFS, where the number of model switches is a predefined number. In lieu of developing formal performance bounds for the given problem scenario, we benchmark the algorithms' performance using Kalman filters, which is described in Section VII-B. Future work will investigate regret bounds for our particular problem.

A. Scenario Definitions

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

1) $\mathcal{M}^{\text{Full}}$, all of the models developed in Section V, i.e., every combination of AC and OL demand models from the AC demand model set $\mathcal{M}^{\text{AC,Full}} = \{\mathcal{M}^{\text{LTI}}, \Phi^{\text{AC,MLR}}, \Phi^{\text{AC,LTV1}}, \Phi^{\text{AC,LTV2}}\}$ and the OL demand model set $\mathcal{M}^{\text{OL}} = \{\Phi^{\text{OL,Mon}}, \Phi^{\text{OL,Tues}}, \Phi^{\text{OL,Wed}}, \Phi^{\text{OL,Thurs}}, \Phi^{\text{OL,Fri}}, \Phi^{\text{OL,MLR}}\}$;
2) $\mathcal{M}^{\text{Red}}$, a reduced set that excludes the LTI models, which are not accurate over the course of the day;
3) $\mathcal{M}^{\text{KF}}$, a further reduced set that excludes the MLR AC demand model, which can not be used in a Kalman filter;

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

B. Kalman filter benchmark

A set of Kalman filters are used to establish a benchmark for the DFS algorithm. A Kalman filter uses measurements, an assumed system model, and known statistics of random variables, which are assumed to be zero-mean and normally distributed, to estimate the value of dynamic system parameters, i.e., the system state, at each time-step. Additional background on Kalman filters can be found in [35].

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

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

C. Data

We test the methods on data from Aug 3-5, 10-14, 17, and 18, where the commercial data is from 2009 and the residential data is from 2015. Note that the dates in both years pertain to the same days of the week. The household and commercial demand data for Aug 3 were used to determine the set of houses included on the feeder and construct the plant. To generate the MLR regression models of the AC and OL demand, we use data from June 24 to Aug 2, 2015 and commercial data from June 24 to Aug 2, 2009. The LTI and LTV models of the AC demand were constructed using device-level data from individual air conditioners from May 2 to Aug 2, 2015. The TOD regression models and Kalman filter covariance matrices were generated using data from the week preceding Aug 3. All testing and training data are available at [36].

TABLE I
PARAMETER η^{s} USED IN THE DFS SCENARIOS IN SECTION VII-D

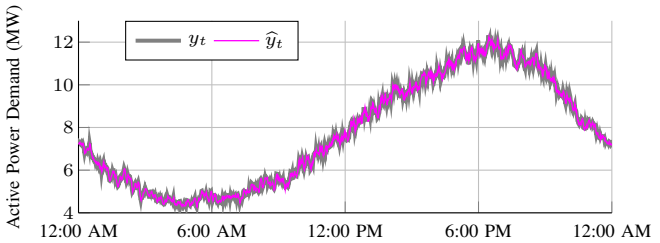
Model Set	$\mathcal{M}^{\text{Full}}$	$\mathcal{M}^{\text{Full}}$	$\mathcal{M}^{\text{Red}}$	$\mathcal{M}^{\text{Red}}$	$\mathcal{M}^{\text{KF}}$	$\mathcal{M}^{\text{KF}}$
Update Method	1	2	1	2	1	2
η^{s}	0.013	0.015	0.4	0.013	0.4	0.5

D. Investigation of Algorithm Performance

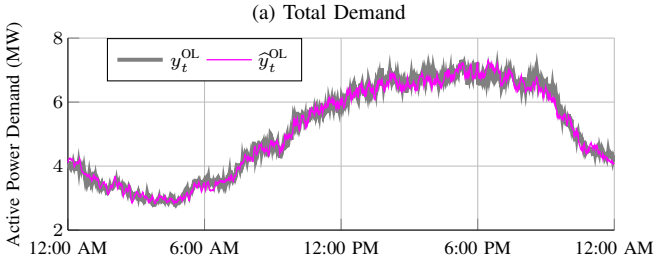
Table I gives the settings of η^{s}, and we set $\lambda = \eta^{\text{r}} = 1.0 \times 10^{-5}$ across all simulations (with the exception of sensitivity simulations in Sections VII-E and VII-F). Parameter λ dictates the amount of weight shared amongst the models, where values near 1 force the DFS algorithm to generate estimates that are close to an average of the predictions of all models. By using a λ value near 0, a single model can dominate the estimate if one model is more accurate than the rest. Parameters η^{r} and η^{s} were roughly tuned to achieve qualitative characteristics of fast switching between models without over-fitting. The optimal η^{r} and η^{s} for a given day will generally not be optimal across all days, and so tuning to achieve the desired qualitative characteristics is appropriate. In practice, η^{r} and η^{s} can be tuned based on recent historical data, and λ can be tuned based on the historical accuracy of the models within the algorithm. An avenue for future research is to develop methods for online parameter tuning using real-time data.

Figure 5 depicts time series for the Aug 17 simulation with $\mathcal{M}^{\text{Red}}$ while using Update Method 1. Figure 6 shows the evolution of the dominant model weights. The weights of the remaining models are summed and referred to as "Other Models." In this scenario, the total demand is accurately separated into its AC demand and OL demand components in real-time, where the RMSE of the total demand, AC demand, and OL demand is 93.2 kW, 151.0 kW, and 150.8 kW, respectively. In this scenario, DFS produces a more accurate AC demand estimate than BKF, which has an AC demand RMSE of 177.3 kW. The RMSE of the AC demand for AKF is 214.0 kW. The majority of the weight is initially given to the "Other Models," because we initialize all models with the same weight. As the simulation progresses, the weight shifts between different model combinations. Since the combinations $\{\Phi^{\text{AC,LTV2}}, \Phi^{\text{OL,MLR}}\}$ and $\{\Phi^{\text{AC,LTV1}}, \Phi^{\text{OL,MLR}}\}$ perform best, they eventually earn more weight and dominate the predictions. It is unsurprising that the $\Phi^{\text{OL,MLR}}$ is the most accurate OL demand model as it captures weather and time variables with the most detail, and it is unsurprising that $\Phi^{\text{AC,LTV1}}$ and $\Phi^{\text{AC,LTV2}}$ are the most accurate AC demand models as they capture the physical phenomenon driving changes in the AC demand as the outdoor temperature changes.

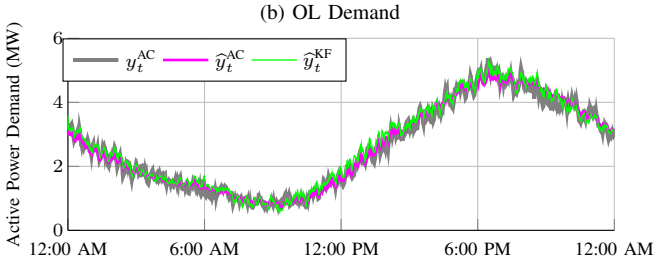
Figure 7 presents the minimum, mean, and maximum RMSE across the full set of testing days for the total demand, the



(a) Total Demand



(b) OL Demand



(c) AC Demand

Fig. 5. Total, OL, and AC demands versus their DFS estimates (Aug 17, $\mathcal{M}^{\text{Red}}$, Update Method 1). The best Kalman filter estimate of the AC demand is also shown.

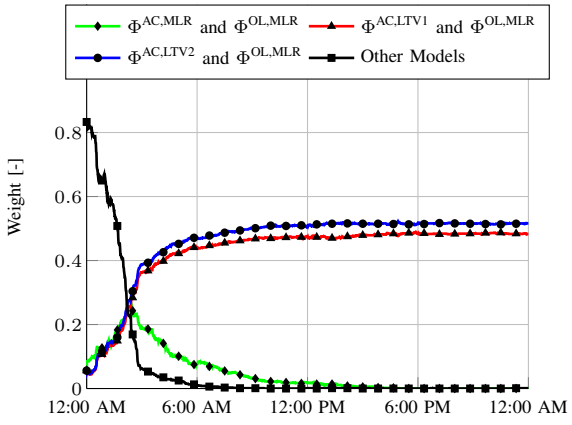
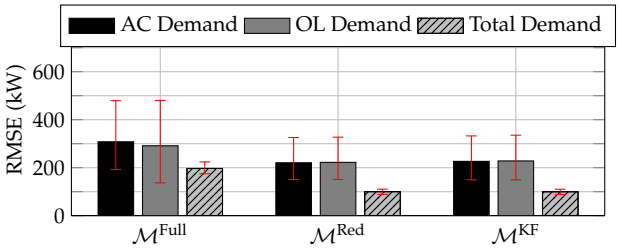
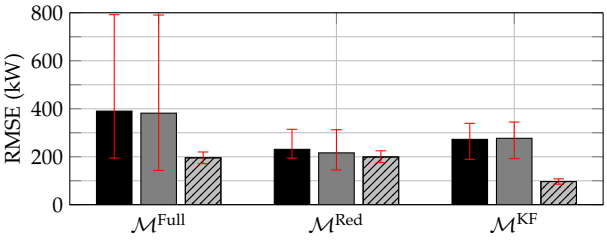


Fig. 6. Model weights (Aug 17, $\mathcal{M}^{\text{Red}}$, Update Method 1).

AC demand, and the OL demand for each DFS scenario. For comparison, BKF achieves a minimum RMSE of 148.4 kW, a mean RMSE of 195.3 kW, and a maximum RMSE of 318.9 kW for the AC demand, and AKF achieves a minimum RMSE of 173.1 kW, a mean RMSE of 259.4 kW, and a maximum RMSE of 357.5 kW for the AC demand. The model corresponding to the BKF varies from day to day and so it is not possible to obtain a single Kalman filter that always outperforms DFS.[1]To demonstrate the value of the measurement-based updates, we generated results for the full set of days using the model set $\mathcal{M}^{\text{Red}}$ and with $\eta^s = 0$; the measurement-based update is irrelevant with this parameter setting. The resulting total demand, AC demand, and OL demand RMSEs were 260.4 kW, 254.2 kW, and 245.2 kW, respectively. These are significant increases over the DFS scenarios using $\mathcal{M}^{\text{Red}}$.



(a) Update Method 1



(b) Update Method 2

Fig. 7. Minimum, mean, and maximum RMSE (kW) for the DFS scenarios in Section VII-D.

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

The scenarios using $\mathcal{M}^{\text{Red}}$ generally do better, in terms of AC demand RMSE, than AKF and worse than BKF. On some simulated days DFS also outperforms BKF, as was shown in Fig. 5. Within this set of simulations, Update Method 2 results in higher AC demand RMSE than Update Method 1. The increased RMSE in Update Method 2 versus Update Method 1 can be explained due to the usage of only two discrete states within the LTV models. Specifically, the states reach their steady-state values rapidly, and so the measurement-based updates to the state do not persist for very long, whereas the measurement-based updates to the output used in Update Method 1 do. Using LTV models with more discrete states may allow Update Method 2 to achieve better RMSE, but this would complicate system identification.

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

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

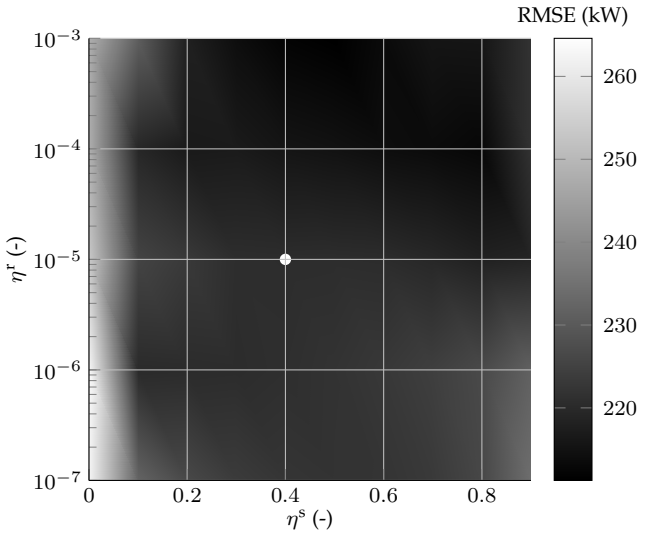


Fig. 8. Average RMSE of the estimated AC demand across all days as a function of η^{s} and η^{r}, using Update Method 1, $\mathcal{M}^{\mathrm{Red}}$, and $\lambda = 1.0 \times 10^{-5}$ where the marker indicates the parameter values used in Section VII-D.

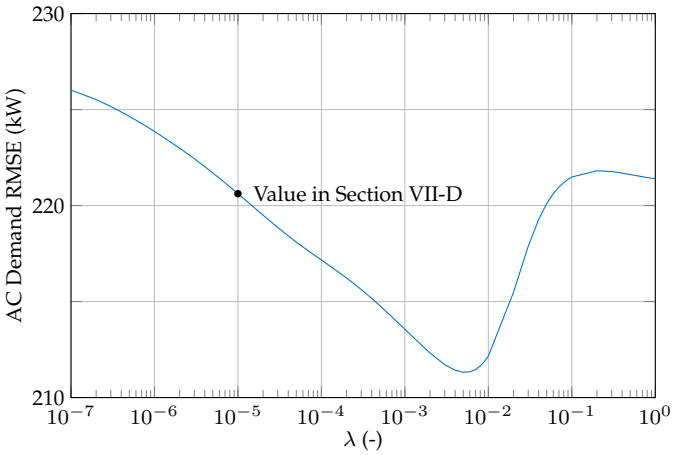


Fig. 9. Average RMSE of the estimated AC demand across all days as a function of λ, using Update Method 1, $\mathcal{M}^{\mathrm{Red}}$, $\eta^{\mathrm{s}} = 0.4$, and $\eta^{\mathrm{r}} = 1.0{\times}10^{-5}$.

E. Sensitivity to the Parameters η^{r} and η^{s}

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

Figure 8 provides the average RMSE of the AC demand across the full set of testing days for each parameter value combination. With η^{s} near zero the RMSE is relatively large as DFS makes small adjustments to the model predictions based on the realized prediction errors. The RMSE with η^{s} near zero decreases slightly as η^{r} increases because this allows for faster transitions in the weighting of the models. However, it should be noted that at larger η^{r} values (e.g., 10^{-3}), the model weights within DFS become erratic or noisy, and overfitting is possible. The RMSE is also relatively high with large η^{s} (e.g., 0.9) and small η^{r} as DFS adjusts the model predictions too aggressively and the model weights change slowly. Alternatively, as η^{r} increases with large η^{s}, the RMSE decreases, but again the weights become vulnerable to overfitting. The RMSE using moderate η^{s} values (e.g., from 0.2-0.7) are similar. The η^{s} and η^{r} values used within Section VII-D do not achieve the lowest RMSE, but they achieve low RMSE while ensuring changes in the weights are reasonably fast but not erratic.

F. Sensitivity to the Parameter λ

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

Figure 9 gives the average RMSE of the estimated AC demand across all days as a function of λ. The average RMSE of the AC demand decreases as λ is increased from 1.0×10^{-7} and reaches a minimum RMSE of 211.3 kW with $\lambda = 0.005$. As λ increases from 0.005, the RMSE increases until it remains relatively constant from 0.1 to 1.0. While we set λ in Section VII-D to allow a single model to dominate the DFS estimate if one model proved to be more accurate than the rest, Fig. 9 indicates that tuning λ on a set days that are similar to the testing days may allow a reduction in the RMSE.

VIII. Conclusions

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

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

Acknowledgments

We thank the Pacific Gas & Electric Company for the commercial building electric load data.

REFERENCES

[1] GTM Research/SEIA: U.S. Solar Market Insight, "Solar market insight report 2015 Q1," 2015. [Online]. Available: http://www.seia.org/research-resources/solar-market-insight-report-2015-q1

[2] Navigant Research, "Direct load control and dynamic pricing programs, DR markets, and DR management systems for residential customers: Global market analysis and forecasts," 2015. [Online]. Available: https://www.navigantresearch.com/research/residential-demand-response#

[3] M. P. Lee, O. Aslam, B. Foster, S. Hou, D. Kathan, C. Pechman, and C. Young, "Assessment of Demand Response & Advanced Metering," FERC, Staff Report, Dec. 2015, https://www.ferc.gov/legal/staff-reports/2015/demand-response.pdf.

[4] C. Baone, Y. Xu, and J. Kueck, "Local voltage support from distributed energy resources to prevent air conditioner motor stalling," in *Innovative Smart Grid Technologies (ISGT)*, 2010.

[5] S. Paul and W. Jewell, "Impact of load type on power consumption and line loss in voltage reduction program," in *North American Power Symposium (NAPS)*, Sept 2013.

[6] E. Can Kara, Z. Kolter, M. Berges, B. Krogh, G. Hug, and T. Yuksel, "A moving horizon state estimator in the control of thermostatically controlled loads for demand response," in *Proceedings of SmartGridComm*, Vancouver, BC, 2013.

[7] S. Esmaeil Zadeh Soudjani and A. Abate, "Aggregation and control of populations of thermostatically controlled loads by formal abstractions," *IEEE Transactions on Control Systems Technology (in press)*, 2014.

[8] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 647–662, 2015.

[9] G. S. Ledva, L. Balzano, and J. L. Mathieu, "Inferring the behavior of distributed energy resources with online learning," in *Allerton Conference on Communication, Control, and Computing*, 2015, pp. 187–194.

[10] M. E. Berges, E. Goldman, H. S. Matthews, and L. Soibelman, "Enhancing electricity audits in residential buildings with nonintrusive load monitoring," *J. Industrial Ecology*, vol. 14, no. 5, pp. 844–858, 2010.

[11] K. C. Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? The case of electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013.

[12] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *Int J. Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[13] T. Hong, "Short term electric load forecasting," Ph.D. dissertation, North Carolina State University, 2010.

[14] T. Byers, "How Comverge is using machine learning to improve demand response forecasts," Feb. 2016. [Online]. Available: http://blog.comverge.com/expert-insights/how-comverge-is-using-machine-learning-to-improve-demand-response-forecasts/

[15] J. W. Taylor, "An evaluation of methods for very short-term load forecasting using minute-by-minute British data," *Int J Forecasting*, vol. 24, no. 4, pp. 645–658, 2008.

[16] N. Hatziargyriou, "Machine learning applications to power systems," in *Machine Learning and Its Applications*. Springer, 2001, pp. 308–317.

[17] M. Negnevitsky, P. Mandal, and A. K. Srivastava, "Machine learning applications for load, price and wind power prediction in power systems," in *Intelligent System Applications to Power Systems*, 2009.

[18] D. Niu, Y. Wang, and D. D. Wu, "Power load forecasting using support vector machine and ant colony optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2531–2539, 2010.

[19] M.-G. Zhang, "Short-term load forecasting based on support vector machines regression," in *Machine Learning and Cybernetics*, vol. 7, 2005, pp. 4310–4314.

[20] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting solar generation from weather forecasts using machine learning," in *SmartGridComm*, 2011.

[21] T. Teo, T. Logenthiran, and W. Woo, "Forecasting of photovoltaic power using extreme learning machine," in *Innovative Smart Grid Technologies-Asia*, 2015.

[22] J. A. Taylor and J. L. Mathieu, "Index policies for demand response," *IEEE Trans on Power Systems*, vol. 29, no. 3, pp. 1287–1295, 2014.

[23] D. Kalathil and R. Rajagopal, "Online learning for demand response," in *Allerton Conference on Communication, Control, and Computing*, 2015.

[24] F. Ruelens, B. Claessens, S. Vandael, B. De Schutter, R. Babuska, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Transactions on Smart Grid*, 2016.

[25] K. Khezeli and E. Bitar, "Risk-sensitive learning and pricing for demand response," *arXiv preprint arXiv:1611.07098*, 2016.

[26] A. Lesage-Landry and J. A. Taylor, "Learning to shift thermostatically controlled loads," in *Hawaii International Conference on System Sciences*, 2017.

[27] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.

[28] K. P. Schneider, Y. Chen, D. P. Chassin, R. G. Pratt, D. W. Engel, and S. E. Thompson, "Modern grid initiative distribution taxonomy final report," Pacific Northwest National Laboratory, Tech. Rep., 2008.

[29] Pecan Street Inc., "Dataport," https://dataport.cloud/, 2017.

[30] NOAA, "NNDC Climatic Data Online," 2009, Satellite and Information Service National Climate Data Center. [Online]. Available: http://www7.ncdc.noaa.gov/CDO/dataproduct

[31] J. Mathieu, A. Gadgil, D. Callaway, P. Price, and S. Kiliccote, "Characterizing the response of commercial and industrial facilities to dynamic pricing signals from the utility," in *ASME International Conference on Energy Sustainability*, Phoenix, AZ, May 2010.

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

[33] K. Kalsi, M. Elizondo, J. Fuller, S. Lu, and D. Chassin, "Development and validation of aggregated models for thermostatic controlled loads with demand response," in *Hawaii International Conference on Systems Science*, 2012.

[34] J. L. Mathieu, M. Kamgarpour, J. Lygeros, G. Andersson, and D. S. Callaway, "Arbitraging intraday wholesale energy market prices with aggregations of thermostatic loads," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 763–772, 2015.

[35] M. S. Grewal and A. P. Andrews, *Kalman filtering*. Wiley, 2015.

[36] G. S. Ledva, L. Balzano, and J. L. Mathieu, "Data Files for Real-Time Energy Disaggregation of a Distribution Feeder's Demand," http://dx.doi.org/10.7302/Z2668BBC, Deep Blue Data at The University of Michigan, 2017.

Gregory S. Ledva (S'16) received the B.S. degree in Mechanical Engineering from the Pennsylvania State University, State College, PA, USA, in 2010. He received the M.S. degree in Energy Science and Technology at ETH Zürich (Swiss Federal Institute of Technology) in 2014. He is currently a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, MI, USA. His research interests include modeling, estimation, control, and optimization of distributed energy resources as well as the adaptation of machine learning algorithms for control applications.

Laura Balzano Laura Balzano is an assistant professor in Electrical Engineering and Computer Science at the University of Michigan. She is an Intel Early Career Faculty Honor Fellow. She received her Ph.D. in Electrical and Computer Engineering from the University of Wisconsin, Madison along with the ECE Best Dissertation Award. Her main research focus is online learning and optimization for low-rank models, with high-dimensional data that are highly incomplete or corrupted. She works on applications in networks, environmental monitoring, and computer vision.

Johanna L. Mathieu (S'10, M'12) received the B.S. degree in ocean engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2004 and the M.S. and Ph.D. degrees in mechanical engineering from the University of California, Berkeley, USA, in 2008 and 2012, respectively. She is an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, MI, USA. Prior to joining the University of Michigan, she was a postdoctoral researcher at the Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. Her research interests include modeling, estimation, control, and optimization of distributed energy resources.