Disaggregating Load by Type from Distribution System Measurements in Real-Time

Gregory S. Ledva, Zhe Du, Laura Balzano, and Johanna L. Mathieu

Abstract An electricity distribution network's efficiency and reliability can be improved using real-time knowledge of the total consumption/production of different load/generator types (e.g., air conditioning loads, lighting loads, photovoltaic generation) within the network. This information could be gathered from additional device-level sensors and communication infrastructure. Alternatively, this information can be inferred using existing network measurements and some knowledge of the underlying system. This work applies two online learning algorithms, Dynamic Mirror Descent (DMD) and Dynamic Fixed Share (DFS), to separate (or disaggregate), in real-time, feeder-level active demand measurements into two components: 1) the demand of a population of residential air conditioners and 2) the demand of the remaining loads served by the feeder. The online learning algorithms include models of the underlying load types, which are generated using historical building-level or device-level data. We develop methods to incorporate model prediction error statistics into the algorithms, develop connections between DMD and Kalman filtering, adapt the algorithms for the energy disaggregation application, and present case studies demonstrating that the algorithms perform disaggregation effectively.

G.S. Ledva
Department of Electrical Engineering and Computer Science, University of Michigan, MI, USA, e-mail: gsledv@umich.edu

Z. Du
Department of Electrical Engineering and Computer Science, University of Michigan, MI, USA e-mail: zhedu@umich.edu

L. Balzano
Department of Electrical Engineering and Computer Science, University of Michigan, MI, USA e-mail: girasole@umich.edu

J.L. Mathieu
Department of Electrical Engineering and Computer Science, University of Michigan, MI, USA e-mail: jlmath@umich.edu

This research was funded by NSF Grant #ECCS-1508943. We also thank the Pacific Gas and Electric Company for the commercial building electric load data.

1 Introduction

Power system entities such as utilities and third-party companies can improve an electricity distribution network's reliability and efficiency using real-time knowledge of the mix of load/generation within the distribution network. The mix of load/generation refers to the aggregate consumption/production of different types of load/generators, e.g., air conditioning loads, lighting loads, and photovoltaic generation. For example, a utility can better anticipate and plan for fault induced delayed voltage recovery (FIDVR) caused by motor stalling if it knows the real-time power consumption of small motor loads [2]. Companies that offer power system services via demand response are interested in knowing the time-varying, total electric load available for demand response. This knowledge can help them bid into ancillary services markets, or it could be used as a feedback signal within a load coordination algorithm [6, 12, 23, 26, 30, 32, 42, 43].

We define real-time feeder-level energy disaggregation as the problem of determining the mix of loads/generation connected to a distribution feeder as measurements arrive sequentially in time. This type of energy disaggregation can be accomplished by either computing the mix of loads/generation directly from device-level (i.e., submetering) data or by inferring the mix of loads/generation from distribution network and smart meter data. Acquiring real-time submetering data requires the installation of additional meters for tens of thousands of devices and also requires additional communication infrastructure to transmit the device-level data to a central location for real-time processing. Estimates of the per household costs associated with submetering are $100 to over $1,000 [1], which limits its practicality.

Alternatively, the mix of loads/generation can be inferred using existing infrastructure: a small number of distribution network measurements (e.g., the power demand served by each feeder) and historical data collected by smart meters. Smart meters capable of measuring household demand at frequent intervals[1] have been widely installed [29], but their communication limitations prevent their data from being available in real-time [1]. However, historical data is available. Device-level demand could be estimated by disaggregating the household-level demand [1].

Inferring the mix of loads/generation can be achieved using online learning algorithms, a class of machine learning algorithms. In the single predictor setting, these algorithms use sequential data (or measurements) to update parameters (referred to here as states) within a predictor, which generates predictions about future data. In a setting with multiple predictors, called prediction with expert advice, algorithms use a defined set of predictors (referred to as experts), and they use the measurements to learn the best expert or best combination of experts, e.g., see [17, 22]. Much of the online learning literature assumes that the optimal state or the best expert does not vary in time, e.g., see [3, 7, 11, 22, 33]. Several papers (e.g., see [17–19, 45, 47]) provide performance bounds on these algorithms when the optimal state or best

[1] While most meters are currently configured to measure/record average power demand over 15 minute or hourly intervals, they generally have the ability to measure/record average power over much shorter intervals, for example, every minute.

expert is allowed to be time-varying. However, in this case, the bounds are only meaningful (i.e., they scale sublinearly with respect to time) when the system (i.e., the state or best expert) varies relatively slowly in time.

Online convex programming is a method to solve online learning problems [39], and recent work [15, 38, 41] incorporates dynamic models into the online learning framework. Online convex programming uses a convex objective function to quantify the error between the predicted measurement computed by a predictor and the actual measurement. After each measurement is revealed, the predictor is updated as a function of the (possibly time-varying) convex objective function. Methods to solve convex optimization problems have been adapted to solve online convex programming, e.g., see [3, 11, 33]. Recently, [15, 38, 41] developed online convex optimization algorithms that handle highly time-varying systems by incorporating dynamic models of the systems. These algorithms establish performance bounds that depend on the accuracy of the underlying dynamic models rather than the variability of the state, allowing the algorithms to be effective in situations with highly time-varying states.

In this work, we apply two algorithms from [15], Dynamic Mirror Descent (DMD) and Dynamic Fixed Share algorithms (DFS), to the feeder-level energy disaggregation problem. In our setting, we disaggregate a distribution feeder's demand measurements into two components: 1) the total power demand of a population of air conditioners, and 2) the total power demand of all remaining loads served by the distribution feeder. The contributions of this work are as follows. 1) We summarize the DMD and DFS algorithms and provide simple examples of DMD implementations to provide intuition; 2) we develop methods to include model prediction error statistics into the DMD and DFS algorithms; 3) we establish connections between DMD and a discrete-time Kalman filter; and 4) we present simulations that show the effectiveness of the algorithms on the real-time feeder-level energy disaggregation problem and also show the influence of model prediction error statistics on the performance of the algorithms. We presented preliminary work in [27] and developed the data-driven case studies expanded upon in this chapter in [28].

The remainder of the chapter is organized as follows: Section 2 summarizes the problem framework that we consider for real-time feeder-level energy disaggregation and compares this problem framework to building-level energy disaggregation. Section 3 summarizes the DMD and DFS algorithms, discusses the inclusion of prediction error statistics into DMD, and makes comparisons between DMD and Kalman filtering; an appendix presents two simple example implementations of DMD. Section 4 describes the application of DFS to the feeder-level energy disaggregation problem, including a summary of the models used within DFS, algorithm implementation details, and discussion of a number of case studies. Finally, Section 5 presents the conclusions.

2 Framework for Real-Time Feeder-Level Energy Disaggregaion

In our framework for real-time feeder-level energy disaggregation, we assume that a power system entity has access to real-time measurements of the active power demand served by a distribution feeder, real-time outdoor temperature measurements, and historical feeder, temperature, and load data. We assume that the feeder serves both residential and commercial loads, and we also assume that the power system entity's objective is to determine the real-time demand of a population of residential air conditioners, referred to as the AC demand, from the feeder's total demand measurements. The other component of the feeder demand, consisting of the commercial and remaining residential demand, is referred to as the other load (OL) demand. The demand is measured at one minute intervals, and the measurements are the time-averaged active power demand over the interval. The real-time outdoor temperature measurements correspond to that of the physical area containing the underlying loads; note that we do not use weather data for individual loads. The temperature measurements are used within some models in real-time where the models are parameterized with historical data.

We assume that the power system entity has access to four sources of historical data, which it uses to parameterize models that predict the two demand components during the real-time disaggregation. The historical data includes past feeder demand measurements, past outdoor temperature measurements, historical building-level demand data for both the residences and commercial buildings, and device-level demand data for the residential air conditioners constituting the AC demand. We assume that building- and device-level meters collect demand data at one minute intervals, but the data are not available in real-time due to communication limitations [1].

The feeder-level energy disaggregation problem has similarities with building-level energy disaggregation [1, 8, 9, 24, 25, 36, 44], also known as non-intrusive load monitoring (NILM) [5, 10, 16, 46, 48], which separates the measured power demand of a building into the demand of individual loads or groups of loads within the building. Building-level energy disaggregation algorithms typically use an aggregate signal that is sampled at high frequencies (e.g., 10 KHz to over 1 MHz) and composed of 10-100 component loads. The algorithms generally leverage assumptions stemming from the relatively small number of underlying loads (e.g., a single device turns on or off per time-step [24] or that step changes can be seen in the aggregate signal [9]). Furthermore, disaggregation is generally performed offline. Building-level energy disaggregation algorithms include supervised and unsupervised approaches. In supervised approaches, historical disaggregated signals are available [4, 5, 8, 16, 25], and unsupervised approaches use only the aggregate signal [13, 21, 24, 40].

In contrast to building-level energy disaggregation, feeder-level energy disaggregation uses an aggregate signal composed of tens of thousands of underlying loads, but we are only interested in disaggregating load by type. We assume that the aggregate signal is sampled less frequently, i.e., on the order of seconds to minutes. The

large number of loads and relatively slow measurement frequency renders many of the building-level approaches inadequate.

3 Summary and Discussion of the DMD and DFS Algorithms

An online learning algorithm predicts a system state $\widehat{\boldsymbol{\theta}}_t \in \Theta$ where the domain of the state $\Theta \subset \mathbb{R}^p$ is a bounded, closed, convex feasible set. After a prediction, denoted $\widehat{\boldsymbol{\theta}}_t \in \Theta$, is formed for time-step t, the system produces a measurement $\boldsymbol{y}_t \in \mathscr{Y} \subset \mathbb{R}^q$. A convex loss function $\ell_t : \Theta \to \mathbb{R}$ measures the accuracy of the prediction $\widehat{\boldsymbol{\theta}}_t$ with respect to $\boldsymbol{y}_t$, and by computing its gradient/subgradient we can determine how to change the prediction to improve its accuracy with respect to the measurement. The loss function may also contain some known, possibly time-varying, function $h_t : \Theta \to \mathscr{Y}$ that computes a predicted measurement from the state prediction, i.e., $\widehat{\boldsymbol{y}}_t = h_t(\widehat{\boldsymbol{\theta}}_t)$. We assume that we can choose the form of the loss function, e.g., $\ell_t(\widehat{\boldsymbol{\theta}}_t) = \|C\widehat{\boldsymbol{\theta}}_t - \boldsymbol{y}_t\|_2^2$ where $h_t(\boldsymbol{\theta}) = C\boldsymbol{\theta}$. The goal of the online learning algorithm is to generate a sequence of predictions that result in low cumulative loss, which is the total achieved loss up to the current time-step.

In this section, we first summarize two online learning algorithms, DMD and DFS, which were originally developed in [15]. From a control systems perspective, DMD and DFS are similar to state estimation algorithms that consider a single model and multiple models, respectively, in forming the estimate of $\boldsymbol{\theta}_t$. Following the algorithm descriptions, we discuss the inclusion of prediction error statistics into DMD. We present a number of simple simulations in the appendix to provide intuition on several parameters and functions that can be chosen by the user within DMD and DFS.

3.1 The DMD Algorithm

The DMD algorithm uses a convex optimization formulation and a model to predict the system state at each time-step. The formulation has similarities to a discrete-time Kalman filter in that the Kalman filter and DMD iteratively use a model to advance the prediction to the next time-step, then adjust the prediction once the new measurement is available. Section 3.3 further develops connections between a Kalman filter and DMD.

The DMD algorithm formulation is

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

$$\widehat{\boldsymbol{\theta}}_{t+1} = \Phi(\widetilde{\boldsymbol{\theta}}_t) \tag{2}$$

where (1) adjusts the prediction using the new measurement and (2) is the model-based update that advances the adjusted prediction. In (1), $\widetilde{\boldsymbol{\theta}}_t$ is the adjusted prediction, and we minimize the right-hand side over the variable $\boldsymbol{\theta}$. The parameter $\eta^{\mathrm{s}} > 0$ is a step-size parameter, $\langle \cdot, \cdot \rangle$ is the standard dot product, $\nabla \ell_t(\widehat{\boldsymbol{\theta}}_t)$ is a sub-gradient of the convex loss function, and $D(\boldsymbol{\theta} \| \widehat{\boldsymbol{\theta}}_t)$ is a Bregman divergence, which penalizes the deviation between the optimizer $\boldsymbol{\theta}$ and the prediction $\widehat{\boldsymbol{\theta}}_t$. In (2), the model $\Phi(\cdot)$ advances the adjusted prediction $\widetilde{\boldsymbol{\theta}}_t$ to the next time-step. The step-size η^{s} influences how aggressively DMD adjusts $\widetilde{\boldsymbol{\theta}}_t$ to match the measurements versus trusting the model-based prediction; see the appendix for an illustrative example.

3.2 The DFS Algorithm

The DFS algorithm incorporates N^{mdl} experts where each expert contains one model from a set of N^{mdl} models $\mathscr{M}^{\mathrm{mdl}} = \{1, \ldots, N^{\mathrm{mdl}}\}$. We use $\mathscr{M}^{\mathrm{mdl}}$ to denote both the set of experts and their corresponding models. DFS assumes that DMD produces each expert's prediction, and then applies the Fixed Share algorithm [17] to form an overall prediction of the system state. The Fixed Share algorithm formulation is

$$w_{t+1}^m = \frac{\lambda}{N^{\mathrm{mdl}}} + (1-\lambda)\, \frac{w_t^m \exp\left(-\eta^r\, \ell_t\left(\widehat{\boldsymbol{\theta}}_t^m\right)\right)}{\sum_{j=1}^{N^{\mathrm{mdl}}} w_t^j \exp\left(-\eta^r\, \ell_t\left(\widehat{\boldsymbol{\theta}}_t^j\right)\right)} \qquad m \in \mathscr{M}^{\mathrm{mdl}} \tag{3}$$

$$\widehat{\boldsymbol{\theta}}_{t+1} = \sum_{m \in \mathscr{M}^{\mathrm{mdl}}} w_{t+1}^m\, \widehat{\boldsymbol{\theta}}_{t+1}^m \tag{4}$$

where (3) advances the weight of each expert and (4) forms the overall prediction as a weighed combination of the individual experts' estimates. In (3), $\widehat{\boldsymbol{\theta}}_t^m$ is the prediction of expert m at time-step t, w_t^m is the weight associated with expert m, $\lambda \in (0,1)$ is a user-defined parameter that influences the weight that is shared amongst experts, and $\eta^{\mathrm{r}} > 0$ is a user-defined parameter that influences how rapidly the algorithm can shift weight between the experts. DFS assumes that $\widehat{\boldsymbol{\theta}}_t^m$ is the value computed in (2) using model $\Phi^m(\cdot)$ for $m \in \mathscr{M}^{\mathrm{mdl}}$. The weight w_t^m is based on each expert's total loss up until time t and the weight that is shared amongst all of the experts. Parameter λ controls the extent to which a single model can dominate the prediction: with λ near zero a single model can dominate, and with λ near one the overall prediction is forced to be close to the average of the individual predictions. For a given sequence of losses, the parameter η^{r} controls how rapidly the weights adjust, with larger values leading to faster weight changes. Setting η^{r} too high may lead to over-fitting, i.e., the weights may become erratic.

3.3 Including Model Prediction Error Statistics into DMD

In this section, we describe how the user can construct the DMD updates to include statistical information about the prediction errors. To support this claim, we design the DMD updates to match those of a Kalman filter, which specifically accounts for zero-mean and normally-distributed errors in its model-based update and measurement predictions. Future work will develop methods to include error statistics corresponding to other probability distributions within DMD.

The loss function and divergence used within DMD must be convex, but their specific form can be chosen by the user. In choosing the convex loss and divergence functions, the user implicitly makes assumptions about statistics describing the model prediction accuracy within the measurement-based update of (1). For example, choosing the divergence function to be a squared ℓ_2-norm, i.e., $D(\boldsymbol{\theta}\|\widehat{\boldsymbol{\theta}}_t) = \|\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t\|_2^2$, treats all errors equally and weights larger errors more. This corresponds to the case where the error covariance matrix is equal to an identity matrix. However, using a Mahalanobis distance, i.e., a weighted squared ℓ_2-norm, such as $(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t)^{\mathrm{T}} \widehat{P}_t^{-1} (\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t)$ with some positive-definite matrix $\widehat{P}_t$, assumes that the errors in the model-based update have a covariance of $\widehat{P}_t$.

A Kalman filter's objective function is to minimize the mean-squared estimation error of the state under assumptions that the system is linear and that state and measurement prediction errors (often referred to as process and measurement noise) are independent in time, zero-mean, normally distributed, and independent from one another. The user must specify the error covariance matrices used within the closed-form update equations. Alternatively, within DMD, the user has the flexibility to select functions (rather than matrices). DMD can produce update equations identical to those of a Kalman filter by making the same assumptions required by a Kalman filter and then appropriately selecting the loss and divergence functions. In the remainder of this section, we summarize the discrete-time Kalman filter and show how to choose the model, divergence function, and loss function within DMD to produce update equations equal to those of a Kalman filter.

A discrete-time Kalman filter assumes an underlying system is [14, p.190]

$$\boldsymbol{\theta}_{t+1} = A_t\, \boldsymbol{\theta}_t + \boldsymbol{w}_t \tag{5}$$

$$\boldsymbol{y}_t = C_t\, \boldsymbol{\theta}_t + \boldsymbol{v}_t \tag{6}$$

where $\boldsymbol{w}_t$ is the process noise (which includes modeling error) and $\boldsymbol{v}_t$ is the measurement noise. The formulation assumes that $\boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, Q_t)$, $\boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{0}, R_t)$, and that A_t, C_t, Q_t, and R_t are known. The notation $\boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{\alpha}, \beta)$ indicates that a random variable $\boldsymbol{\phi}$ is sampled from a normal distribution with mean $\boldsymbol{\alpha}$ and a symmetric, positive-definite covariance β.

A Kalman filter uses the assumptions on the underlying system and the known system parameters to estimate $\boldsymbol{\theta}_t$ at each time-step while minimizing the the mean-squared estimation error. The resulting update equations are [14, p.190]

$$\widetilde{\boldsymbol{\theta}}_t = \widehat{\boldsymbol{\theta}}_t + \widehat{P}_t C_t^T \left[C_t \widehat{P}_t C_t^T + R_t \right]^{-1} \left(\boldsymbol{y}_t - C_t \widehat{\boldsymbol{\theta}}_t \right) \tag{7}$$

$$\widetilde{P}_t = \widehat{P}_t - \widehat{P}_t C_t^{\mathrm{T}} \left[C_t \widehat{P}_t C_t^T + R_t \right]^{-1} C_t \widehat{P}_t \tag{8}$$

$$\widehat{\boldsymbol{\theta}}_{t+1} = A_t \, \widetilde{\boldsymbol{\theta}}_t \tag{9}$$

$$\widehat{P}_{t+1} = A_t \, \widetilde{P}_t \, A_t^{\mathrm{T}} + Q_t \tag{10}$$

where $\widehat{\boldsymbol{\theta}}_t$ is the *a priori* state estimate, $\widetilde{\boldsymbol{\theta}}_t$ is an *a posteriori* state estimate, $\widetilde{P}_t$ is the *a priori* estimation error covariance, and $\widehat{P}_t$ is the *a posteriori* estimation error covariance. If the matrices within the system model are time-invariant, $\widehat{P}_t$ converges to a steady-state value, denoted $\overline{P}$. A steady-state Kalman filter uses $\overline{P}$ in (7).

We choose the model, divergence function, and loss function within DMD to construct the DMD updates (1) and (2). We first set the DMD model to $\Phi(\widetilde{\boldsymbol{\theta}}_t) = A_t \, \widetilde{\boldsymbol{\theta}}_t$, which makes (2) the same as (9). Note that this corresponds to assuming the model is linear with a state-update matrix A_t, as in a Kalman filter. We then set the divergence and loss function to

$$D(\boldsymbol{\theta} \| \widehat{\boldsymbol{\theta}}_t) = \frac{1}{2} \left\| (\widehat{P}_t)^{-\frac{1}{2}} (\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t) \right\|_2^2 \tag{11}$$

$$\ell_t(\widehat{\boldsymbol{\theta}}_t) = \frac{1}{2} \left\| (\widehat{P}_t^{\mathrm{y}})^{-\frac{1}{2}} (C \widehat{\boldsymbol{\theta}}_t - \boldsymbol{y}_t) \right\|_2^2 \tag{12}$$

where $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ are symmetric, positive definite, covariance matrices corresponding to the model prediction errors and the measurement prediction errors, respectively. The quantity $G^{-\frac{1}{2}} = U \left(\Sigma^{-\frac{1}{2}} \right) U^{\mathrm{T}}$ denotes a matrix square root of an arbitrary symmetric positive-definite matrix G, where U is orthonormal and Σ a diagonal matrix with positive entries on the diagonal. The square roots of $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ are also symmetric and positive definite [14]. Given the assumptions thus far, the matrices A_t, $\widehat{P}_t$, and $\widehat{P}_t^{\mathrm{y}}$ can be treated as parameters that are known at each time-step within DMD.

Given our choice of model, divergence function, and loss function, we can use (1) to derive a closed-form DMD update equation, which is the same as (7). We start from (1), substitute the divergence function, and then solve the convex program by finding the value of $\boldsymbol{\theta}$ that sets the gradient of the convex objective function equal to 0. Following this, we substitute the gradient of the loss function. These steps result in

$$\widetilde{\boldsymbol{\theta}}_t = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min} \; \eta^{\mathrm{s}} \left\langle \nabla \ell_t(\widehat{\boldsymbol{\theta}}_t), \, \boldsymbol{\theta} \right\rangle + D \left(\boldsymbol{\theta} \| \widehat{\boldsymbol{\theta}}_t \right) \tag{13}$$

$$= \widehat{\boldsymbol{\theta}}_t + \eta^{\mathrm{s}} \widehat{P}_t \left(-\nabla \ell_t(\widehat{\boldsymbol{\theta}}_t) \right) \tag{14}$$

$$= \widehat{\boldsymbol{\theta}}_t + \eta^{\mathrm{s}} \widehat{P}_t C_t^T \left(\widehat{P}_t^{\mathrm{y}} \right)^{-1} \left(\boldsymbol{y}_t - C \widehat{\boldsymbol{\theta}}_t \right) \tag{15}$$

where $\widehat{P}_t^{\mathrm{y}} = (C_t \widehat{P}_t C_t^T + R_t)$. Finally, setting $\eta^{\mathrm{s}} = 1$ produces the same update as (7). Note that $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$ are the same covariances as used in the Kalman filter. Their

values and updates are assumed known, which is the same assumption we make when we use the Kalman filter.

4 Application of DFS to Real-Time Feeder-Level Energy Disaggregation

In this section, we apply the DFS algorithm to the problem framework described in Section 2. In the following, Section 4.1 details the construction of the underlying system (i.e., the plant), Section 4.2 describes the construction of the models used within the algorithms, Section 4.3 describes the implementation of the online learning algorithms and a Kalman filter, and Section 4.4 presents some case studies investigating the algorithm's performance. Note that we construct these case studies to investigate the effectiveness of DFS within the problem formulation in comparison to the effectiveness of a Kalman filter. Modifying the models developed and used within DFS as well as the user-defined loss and divergence functions within DMD may provide performance improvements.

4.1 Plant Construction

The plant, which is our representation of the underlying physical system, is composed of the active power demand of a set of commercial and residential loads connected to a distribution feeder. The time series consist of n^{steps} one minute time-steps over the course of one day, resulting in $n^{\text{steps}} = 1440$. We denote the measured total demand of the feeder as $y_t \in \mathbb{R}$, the AC demand as $y_t^{\text{AC}} \in \mathbb{R}$, the residential component of the OL demand as $y_t^{\text{OL,res}} \in \mathbb{R}$, and the commercial component of the OL demand as $y_t^{\text{OL,com}} \in \mathbb{R}$. The total demand is $y_t = y_t^{\text{AC}} + y_t^{\text{OL}}$ where $y_t^{\text{OL}} = y_t^{\text{OL,res}} + y_t^{\text{OL,com}}$.

We construct the y_t^{AC}, $y_t^{\text{OL,res}}$, and $y_t^{\text{OL,com}}$ time series using a feeder model, household demand data, air conditioner demand data, and commercial building demand data; additional details on the time series construction can be found in [28]. We assume the average daily active power demand of the commercial and residential loads is 5.8 MW and 2.1 MW, respectively, which is based on the feeder model R5-25.00-1 from GridLAB-D's feeder taxonomy [37]. The residential demand consists of household demand data and air conditioner demand data from the Pecan Street, Inc. Dataport [35], where the aggregate residential demand corresponds to the summed daily demand of a set of 2,499 households. The AC demand y_t^{AC} corresponds to the summed demand of 2,269 primary air conditioner and blower units within those households. The residential OL demand $y_t^{\text{OL,res}}$ consists of the remainder of the aggregate household demand not included within the AC demand. The commercial OL demand $y_t^{\text{OL,com}}$ is the scaled sum of the whole-building demand from a big box retail store and a municipal building in the California Bay Area.

We neglect losses in the power network, which, if included, would be part of the OL demand. We determine the set of houses, the air conditioner population, and the scaling factor for the commercial demand using data from August 3, as detailed in [28].

We also construct time series of the outdoor temperature for the physical area corresponding to the demand data, which is used in some models described in Section 4.2. The residential data corresponds to Austin, TX, and we use outdoor temperature data from [35]. The outdoor temperature for the commercial demand comes from the Concord, CA National Weather Service station [34].

4.2 Model Construction

In this section, we describe the models used within the DFS algorithm, where [28] provide more details. Section 4.2.1 details the linear regression models used to predict both the AC and OL demand while Section 4.2.2 details the linear dynamic system models, specifically linear time-varying (LTV) system models used to predict the AC demand.

4.2.1 Linear Regression Models

The linear regression models of the AC and OL demand all have the same general form

$$\widehat{y}_t^\star = \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{\beta}_t$$

where $\widehat{y}_t^\star$ is the prediction of the AC or OL demand, $\boldsymbol{\beta}_t$ is a vector of input features at time t, and $\boldsymbol{\alpha}$ is a vector of coefficients. The input features are the explanatory variables. The vector of coefficients forms a weighted combination of the input features, and their values are determined by applying least-squares error minimization to historical data including the input features and the demand signal. Examples of input features used within the models below include calendar variables such as time-of-week and weather variables such as outdoor temperature.

Below, we summarize the input features used in several regression models, including a simple regression model that forms a lookup table based on the time of day (TOD) and two multiple linear regression (MLR) models that use a vector of input features. The TOD regression models were generated using data from the week preceding August 3. We use residential data from June 24 to August 2, 2015 and commercial data from June 24 to August 2, 2009 to generate the MLR models, and we exclude anomalous data such as those corresponding to holidays.

TOD OL Demand Model The TOD OL demand model corresponds to a lookup table of OL demand predictions based on the time of day, generated by smoothing OL demand data from previous days. We construct TOD models for each weekday

denoted, $\Phi^{\text{OL,Mon}}$, $\Phi^{\text{OL,Tues}}$, $\Phi^{\text{OL,Wed}}$, $\Phi^{\text{OL,Thu}}$, $\Phi^{\text{OL,Fri}}$, and their respective predictions are $\widehat{y}_t^{\text{OL,Mon}}$, $\widehat{y}_t^{\text{OL,Tues}}$, $\widehat{y}_t^{\text{OL,Wed}}$, $\widehat{y}_t^{\text{OL,Thu}}$, $\widehat{y}_t^{\text{OL,Fri}}$.

MLR OL Demand Model The MLR OL demand model forms its predictions using two sets of input features and coefficient vectors, one for the residential OL demand and one for the commercial OL demand, where both sets of input features include calendar- and weather-based values. Two sets of input features are necessary because the OL demand data corresponds to two different physical areas. The input features for the residential OL demand are a time-of-week indicator vector, the outdoor temperature for Austin, TX, and the measured total demand at the last time-step, y_{t-1}. The commercial component of the model corresponds to "Baseline Method 1" from [31]. The input features for the commercial OL demand are a time-of-week indicator vector and the outdoor temperature of Concord, CA at the given time-of-week. Whereas the residential component of the model has a single regression parameter for the outdoor temperature, the commercial component has separate temperature-based coefficients for each time of week. We denote the MLR OL demand model and its predictions as $\Phi^{\text{OL,MLR}}$ and $\widehat{y}_t^{\text{OL,MLR}}$, respectively.

MLR AC Demand Model The input features of the MLR AC demand model $\Phi^{\text{AC,MLR}}$, with predictions $\widehat{y}_t^{\text{AC,MLR}}$, are a time-of-week indicator vector and the lagged outdoor temperature for Austin, TX raised to the first through fourth powers, i.e., the model includes a fourth order polynomial in lagged temperature. The lag was chosen to maximize the cross correlation between the temperature and AC demand in the training data.

4.2.2 Linear Dynamic System Models

We also use two LTV dynamic system models to compute predictions of the AC demand. The on/off cycling of air conditioners varies with the outdoor temperature, and we generate LTV models from sets of linear time-invariant (LTI) models, originally developed in [20, 32], each corresponding to a different outdoor temperature. The first LTV model, denoted $\Phi^{\text{AC,LTV1}}$, generates predictions, denoted $\widehat{y}_t^{\text{AC,LTV1}}$, using a set of LTI models $\mathscr{M}^{\text{LTI1}}$ and the lagged, outdoor temperature. The second LTV model, denoted $\Phi^{\text{AC,LTV2}}$, generates predictions $\widehat{y}_t^{\text{AC,LTV2}}$, using a separate set of LTI models $\mathscr{M}^{\text{LTI2}}$ and the time-averaged outdoor temperature over a window of previous minutes. The lag and the window are chosen to maximize the performance of the models on the training set. Both LTV models have the form

$$\widehat{\boldsymbol{x}}_{t+1}^{\star} = A_t^{\star}\,\widehat{\boldsymbol{x}}_t^{\star}$$
$$\widehat{y}_t^{\text{AC},\star} = C_t^{\star}\,\widehat{\boldsymbol{x}}_t^{\star}$$

where superscript $\star$ is replaced by LTV1 for $\Phi^{\text{AC,LTV1}}$ or LTV2 for $\Phi^{\text{AC,LTV2}}$. The first element of the vector $\widehat{\boldsymbol{x}}_t^{\star} \in \mathbb{R}^2$ captures the portion of air conditioners that are drawing power, i.e., those that are on, and the second element captures the portion of air conditioners not drawing power, i.e., those that are off. The matrix $A_t^{\star}$

includes the probabilities that a given air conditioner: 1) switches on during the time-step, 2) switches off during the time-step, 3) remains on, or 4) remains off. The matrix $C_t^\star$ scales the portion of air conditioners that are drawing power by an average power demand value to compute the prediction $\widehat{y}_t^{\mathrm{AC},\star}$. The LTI models in $\mathscr{M}^{\mathrm{LTI1}}$ and $\mathscr{M}^{\mathrm{LTI2}}$ have the same form as the LTV models, with time-invariant matrices identified from data corresponding to narrow ranges around specific outdoor temperatures. The time-varying matrices $A_t^\star$ and $C_t^\star$ are computed by linearly interpolating the elements of the two closest LTI models (where "closest" is measured in terms of lagged or average temperature). The LTI models are computed using air conditioner demand data from May 2 to August 2, 2015.

4.3 Algorithm Implementation Details

In this section, we describe the implementation of three algorithms used for the feeder-level energy disaggregation problem. First, we describe the implementation of a Kalman filter, which we as the benchmark for the case studies presented in Section 4.4. We then describe an algorithm, referred to as P-DFS, that includes a modified version of DMD, referred to as P-DMD for pseudo-DMD. P-DMD includes measurement-based updates and model-based predictions but modifies the DMD equations (1) and (2) allowing us to include models of various forms, e.g., both LTV and MLR models, within the Fixed Share algorithm. Following this, we describe the DFS implementation. Within DFS, an expert applies DMD when the AC demand is modeled using an LTV model and P-DMD when the AC demand is modeled using an MLR model. Each of the methods detailed below incorporates model prediction error statistics explicitly. Several methods for constructing the covariances are detailed and investigated in Section 4.4. Note that in all implementations, we construct the convex program within DMD to have a closed-form solution. Given this, the computational complexity of the DFS implementation is similar to that of a set of Kalman filters.

In all three algorithms, the model of the feeder consists of one AC demand model $\Phi^{\mathrm{AC}}(\cdot)$ paired with one OL demand model $\Phi^{\mathrm{OL}}(\cdot)$, i.e., the model is $\Phi(\cdot) = \{\Phi^{\mathrm{AC}}(\cdot), \Phi^{\mathrm{OL}}(\cdot)\}$. P-DFS and DFS use a set of models $\mathscr{M}^{\mathrm{DFS}}$ that consists of every pair of AC and OL demand models described in Section 4.2. The Kalman filter implementation applies to a set of models $\mathscr{M}^{\mathrm{KF}}$ that includes every possible pairing of an LTV AC demand model with an OL demand model.

4.3.1 Kalman Filter

The Kalman filter uses an LTV model to describe the underlying system, estimates the state of the AC demand model, i.e., $\boldsymbol{\theta}_t = \boldsymbol{x}_t^\star$, and uses a pseudo-measurement of the AC demand $\widetilde{y}_t^{\mathrm{AC}} = y_t - \widehat{y}_t^{\mathrm{OL}}$ to adjust the model-based estimate where $\widehat{y}_t^{\mathrm{OL}}$ is the predicted OL demand. A time-invariant process noise covariance Q is computed for

each dynamic AC demand model using the historical AC demand measurements. In computing Q, the true state at each time-step is calculated using the measured AC demand and the AC demand model's matrices. The pseudo-measurement $\widetilde{y}_t^{\mathrm{AC}}$ contains noise due to prediction errors in $\widehat{y}_t^{\mathrm{OL}}$, and a separate time-invariant covariance R is computed for each OL model. We compute Q and R using data for the week preceding August 3. We implement one Kalman filter for each model combination within $\mathscr{M}^{\mathrm{KF}}$.

4.3.2 P-DFS Method

The models developed for this work have a variety of forms and different underlying parameters influencing the demand predictions. As a result, it is difficult to define a $\boldsymbol{\theta}_t$ that is common across all models. To overcome this, we modify the DMD algorithm to decouple model-based updates and measurement-based updates, meaning the measurement-based updates do not influence the model-based updates. This allows the algorithm to be applied to the output of a given model, e.g., the demand predictions, rather than some underlying parameter, while operating in the spirit of DMD. We first proposed this idea in [28].

We modify the model-based and measurement-based updates in DMD, i.e., (1) and (2), to formulate P-DMD, which is used within DFS to form the overall estimate. The P-DMD formulation is

$$\widehat{\boldsymbol{\kappa}}_{t+1}^{m} = \underset{\boldsymbol{\theta}\in\Theta}{\arg\min}\ \eta^{\mathrm{s}} \left\langle \nabla \ell_t(\widehat{\boldsymbol{\theta}}_t^{m}),\ \boldsymbol{\theta} \right\rangle + D\left(\boldsymbol{\theta} \| \widehat{\boldsymbol{\kappa}}_t^{m}\right) \tag{16}$$

$$\check{\boldsymbol{\theta}}_{t+1}^{m} = \Phi(\check{\boldsymbol{\theta}}_t^{m}) \tag{17}$$

$$\widehat{\boldsymbol{\theta}}_{t+1}^{m} = \check{\boldsymbol{\theta}}_{t+1}^{m} + \widehat{\boldsymbol{\kappa}}_{t+1}^{m} \tag{18}$$

for $m \in \mathscr{M}^{\mathrm{DFS}}$. The value $\widehat{\boldsymbol{\kappa}}_t^{m}$ accumulates adjustments to the estimate $\widehat{\boldsymbol{\theta}}_t^{m}$ for model m based on the measurements, and we set $\widehat{\boldsymbol{\theta}}_0^{m} = \mathbf{0}$. The value $\check{\boldsymbol{\theta}}_t^{m}$, is an open-loop state prediction, meaning that the measurements do not influence the prediction (in contrast with DMD), and (17) is the model-based update. Finally, (18) incorporates the accumulated measurement-based adjustment $\widehat{\boldsymbol{\kappa}}_t^{m}$ into the model-based prediction. The AC and OL demand models generate their predictions independently from one another, and so (18) can be rewritten as

$$\widehat{\boldsymbol{\theta}}_{t+1}^{m} = \Phi(\check{\boldsymbol{\theta}}_t^{m}) + \widehat{\boldsymbol{\kappa}}_{t+1}^{m} \tag{19}$$

$$= \begin{bmatrix} \Phi^{\mathrm{AC}}(\check{\boldsymbol{\theta}}_t^{m}) \\ \Phi^{\mathrm{OL}}(\check{\boldsymbol{\theta}}_t^{m}) \end{bmatrix} + \widehat{\boldsymbol{\kappa}}_{t+1}^{m}. \tag{20}$$

The Fixed Share equations (3) and (4) are then applied to the predictions.

In P-DFS, $\boldsymbol{\theta}_t$ is the AC and OL demand, i.e., $\boldsymbol{\theta}_t = \begin{bmatrix} y_t^{\mathrm{AC}} & y_t^{\mathrm{OL}} \end{bmatrix}^{\mathrm{T}}$. We choose the loss function to be (12) and divergence function to be (11) with $\widehat{\boldsymbol{\kappa}}_t^{m}$ as the second

argument rather than $\widehat{\boldsymbol{\theta}}_t^m$. The resulting closed-form update (16) is

$$\widehat{\boldsymbol{\kappa}}_{t+1}^m = \widehat{\boldsymbol{\kappa}}_t^m + \eta^{\mathrm{s}} \widehat{P}_t C^T \left(\widehat{P}_t^{\mathrm{y}}\right)^{-1} \left(y_t - C\widehat{\boldsymbol{\theta}}_t\right) \tag{21}$$

where $C = \begin{bmatrix}1 & 1\end{bmatrix}$. The estimation error covariance $Q_t \in \mathbb{R}^{2\times 2}$ and the measurement noise covariance $R_t \in \mathbb{R}^1$ are used to compute $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$. We set $Q_t = \mathrm{diag}(R_t^{\mathrm{AC}}, R_t^{\mathrm{OL}})$, where $\mathrm{diag}(\cdot)$ forms a diagonal matrix from the scalar arguments. The values $R_t^{\mathrm{AC}} \in \mathbb{R}$ and $R_t^{\mathrm{OL}} \in \mathbb{R}$ correspond to the variances of the AC and OL demand models' prediction errors. We detail several sets of assumptions and methods for computing the parameters R_t, R_t^{AC}, R_t^{OL}, $\widehat{P}_t$, and $\widehat{P}_t^{\mathrm{y}}$ in Section 4.4.

4.3.3 DFS Method

This method also uses the set of models $\mathscr{M}^{\mathrm{DFS}}$. The formulation applies DMD to the LTV AC demand models and P-DMD to all other models, including the OL demand models. The individual model-based estimates are then used as expert predictions within the Fixed Share algorithm. We set $\boldsymbol{\theta}_t = \left[(\boldsymbol{x}_t^\star)^{\mathrm{T}}\ y_t^{\mathrm{OL}}\right]^{\mathrm{T}} \in \mathbb{R}^3$, where $\star$ is LTV1 or LTV2, allowing inclusion of the LTV model dynamics within (1). The model-based update is

$$\widehat{\boldsymbol{\theta}}_{t+1}^m = \begin{bmatrix}\Phi^{\mathrm{AC}}(\widetilde{\boldsymbol{\theta}}_t^m) \\ \Phi^{\mathrm{OL}}(\check{\boldsymbol{\theta}}_t^m)\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix} \widehat{\boldsymbol{\kappa}}_{t+1}^m$$

where we update the AC demand predictions using DMD and the OL demand predictions using P-DMD. The closed-form measurement-based update of the AC demand component is (15). The estimation error covariance $Q_t \in \mathbb{R}^{3\times 3}$, and the measurement noise covariance $R_t \in \mathbb{R}$ are used to compute $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$. The process noise matrix is $Q_t = \mathrm{blkdiag}(Q_t^{\mathrm{AC}}, R_t^{\mathrm{OL}})$ where $\mathrm{blkdiag}(\cdot)$ constructs a block diagonal matrix from the arguments. The matrix $Q_t^{\mathrm{AC}} \in \mathbb{R}^{2\times 2}$ corresponds to the process noise of the AC demand model. We use $A_t = \mathrm{blkdiag}(A_t^\star, 0) \in \mathbb{R}^3$ to update $\widehat{P}_t$ and $\widehat{P}_t^{\mathrm{y}}$, which assumes that errors in the AC demand model are decoupled from errors in the OL demand model, and that the errors of the OL demand model are independent at each time-step. We detail several methods for constructing Q_t^{AC}, R_t^{OL}, $\widehat{P}_t$, and $\widehat{P}_t^{\mathrm{y}}$ in Section 4.4.

4.4 Case Studies

In this section, we describe the setup and summarize the results for a set of case studies investigating the performance of DFS and P-DFS on the feeder-level energy disaggregation problem. We simulate a set of days using data from August 3-5, 10-14, 17, and 18 where we excluded weekends and days on which demand response

Table 1 Parameters η^{s} and η^{r} Used in DFS and P-DFS

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

actions were taken by the commercial buildings. The Kalman filters are used as benchmarks. Specifically, we denote BKF as the "best" Kalman filter achieving the lowest *ex post* root mean square estimation error (RMSEE), and we denote AKF as the average RMSEE of the set of Kalman filters. For an arbitrary time series ψ_t and its estimate $\widehat{\psi}_t$ over n^{steps} time-steps, the RMSEE is defined as

$$\varepsilon^{\mathrm{RMSE}} = \sqrt{\sum_{t=1}^{n^{\mathrm{steps}}} \left(\psi - \widehat{\psi}_t\right)^2 / n^{\mathrm{steps}}}. \tag{22}$$

Table 1 lists the values of the parameters η^{s} and η^{r} used in each scenario; we set $\lambda = 1.0\times10^{-5}$ in all scenarios. We tuned η^{r} using the simulation for August 3, where the goal was to achieve fast weight transitions without over-fitting, i.e., without erratic weights. Qualitative tuning is appropriate as an optimal value for a given simulated day is not necessarily the optimal value for other simulated days. Parameter η^{s} was tuned similarly. In the next subsection, we detail three methods for constructing the covariances, referred to as "Identity", "Historical", and "Real-Time" in Table 1.

4.4.1 Covariances for DFS and P-DFS

In this section, we detail three methods for generating the covariance matrices used within the DFS and P-DFS algorithms. The first method does not explicitly include any model prediction error statistics into the measurement-based updates of DMD and P-DMD. The second method uses historical data from the week preceding August 3 to compute covariance matrices. The third method uses an unrealistic assumption, i.e., that the total, AC, and OL demand are measured at each time-step and used to compute the exact covariance at each time-step. The details of each method are as follows.

1. Identity: we assume that $\widehat{P}_t$ and $\widehat{P}_t^{y}$ are appropriately sized identity matrices for both DFS and P-DFS.
2. Historical: DFS and P-DFS assume that the process noise covariance is time-invariant, i.e., $Q_t = Q$ and that the measurement noise covariance is $R_t \approx 0$ as the total demand measurements are accurate. The covariances Q^{AC}, R^{AC}, R^{OL} used within the two variations of Q are computed using historical estimation errors, and Q^{AC} is used within the Kalman filter. DFS updates $\widehat{P}_t$ according to (8) and (10), and P-DFS sets $\widehat{P}_t = Q$. Both methods set $\widehat{P}_t^{y} = (C\widehat{P}_t C^T + R_t)$.

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

4.4.2 Results

We next summarize the results for each scenario described above. Figure 1 presents time series of the total demand, OL demand, AC demand, their respective estimates, and the model weights from the August 4 simulation while running P-DFS with covariances generated from historical data. In Fig. 1d, Φ^{Other} is used to denote the combined weight of all model combinations not explicitly specified. Table 2 summarizes the mean, minimum, and maximum RMSEE for each demand component across the simulated days and scenarios. Figure 2 presents time series of the AC demand and various estimates across several scenarios from the August 11 simulations.

From Fig. 1c, it is clear that, in this case, the P-DFS algorithm effectively estimates the AC demand in real-time. In this scenario, BKF achieves an RMSEE of 148.4 kW for the AC demand, and the P-DFS algorithm performs similarly, achieving an RMSEE of 155.0 kW for the AC demand. It should be noted that the P-DFS algorithm is determining the model of the underlying system in real-time, as can be seen in Fig. 1d. Alternatively, the BKF algorithm selects the most accurate model after the simulation, which is not feasible in practice. For comparison, AKF achieves an RMSEE of 173.1 kW. The weights within P-DFS is initially dominate by Φ^{Other}, which makes sense as the weight of each model combination is initialized to the same value. As the simulation progresses, the weight shifts to $\{\Phi^{\text{AC,LTV1}}, \Phi^{\text{OL,MLR}}\}$, which is the most accurate model. At points of the simulation, it loses accuracy, and the weight shifts to other model combinations during those times. The total demand is estimated closely, which can be achieved based on the parameter settings as discussed in Section 5. Finally, it should be noted that while P-DFS did not achieve lower RMSEE than BKF in this case, in some cases it does outperform BKF.

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

Figure 2c provides example time series of the AC demand estimates for P-DFS and DFS when using historical covariances, and Fig. 2d provides similar example time series when using real-time covariance data. As can be seen in Fig. 2c, the P-DFS algorithm generally achieves better RMSEE for the AC demand than the DMD algorithm. However, as can be seen in Fig. 2d DFS achieves lower RMSEE

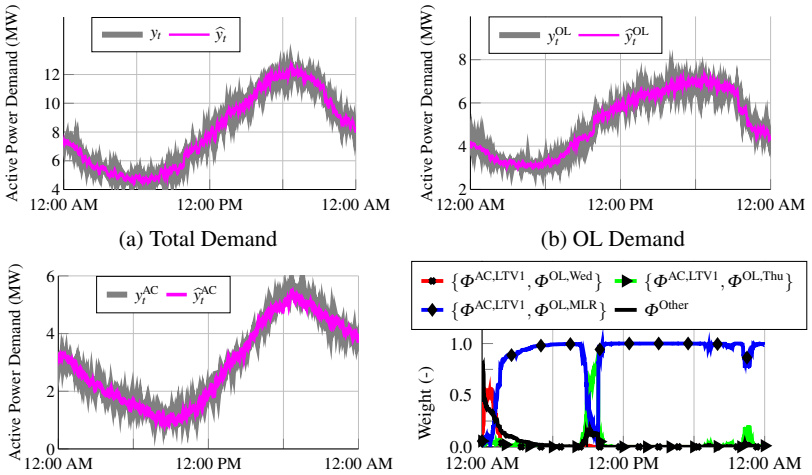


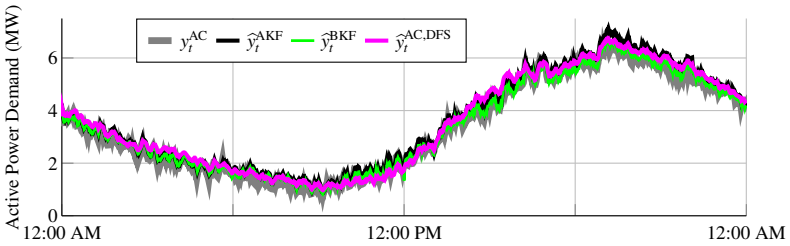
Fig. 1 Time series of the total, OL, and AC demands versus their estimates as well as times series of the weights from the August 4 simulation while running P-DFS with historical covariances

Table 2 Mean, Minimum (Min), and Maximum (Max) RMSEE in kW over 10 Simulated Days for each Algorithm and Covariance Computation Method

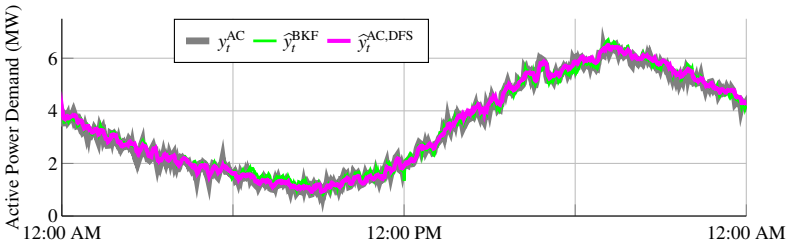
Method	Covariance	Total Demand			AC Demand			OL Demand		
		Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
P-DFS	Identity	88.9	100.0	110.5	151.0	220.6	325.8	150.8	222.3	327.2
P-DFS	Historical	98.4	114.8	123.2	155.0	252.2	371.5	150.2	250.1	372.5
P-DFS	Real-Time	146.6	154.3	168.4	120.2	125.3	131.8	104.8	114.5	130.5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

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

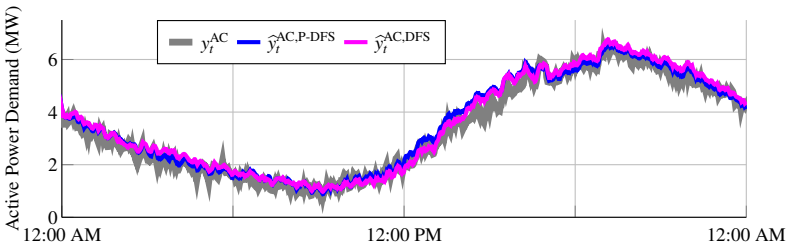
Also, our method of computing the covariances with historical data degrades performance. This implies that our assumptions regarding the errors are overly coarse. However, the inclusion of unrealistically accurate covariance information, which is done when using real-time covariance data, the DFS and P-DFS algorithms' performance improves dramatically.



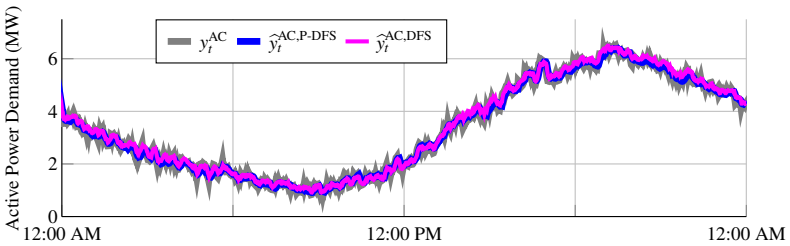
(a) AC demand estimates for BKF, AKF, and DFS while using historical covariances



(b) AC demand estimates for BKF and DFS while using real-time covariances



(c) AC demand estimates for P-DFS and DFS when using historical covariances



(d) AC demand estimates for P-DFS and DFS when using real-time covariances

Fig. 2 Time series of the AC demand and various estimates from the August 11 simulations

5 Conclusions

In this chapter, we summarized the real-time feeder-level energy disaggregation problem and an online learning algorithm, Dynamic Fixed Share (DFS), that we adapted and applied to the problem. It was shown that the Dynamic Mirror Descent (DMD), which is used within the DFS algorithm, can be constructed to be

equivalent to a discrete-time Kalman filter through proper choice of user-defined functions and parameters. In addition, simple examples were constructed to illustrate aspects of the DMD algorithm. Two implementations of DFS-based algorithms were described. The first modifies the DFS algorithm to incorporate combinations of models with different model structures resulting in estimates of output, rather than the state. The second implemented the original DFS algorithm. Finally, case studies were presented that indicate the online learning algorithms are capable of performing real-time feeder-level energy disaggregation and that model prediction error statistics can be effectively incorporated into these algorithms.

Future work will further explore will further explore connections between Kalman filtering and online learning methods, enabling application of results across both well-studied fields. Another topic of future work is addressing the simultaneous problems of active manipulation and online estimation of the AC demand, e.g., in a demand response program.

Appendix

In this appendix, we present two examples that demonstrate the influence of several of the user-defined functions and parameters within DMD. The first example shows how the choice of η^{s} impacts the estimate in the presence of measurement noise. The second example illustrates how the choice of divergence and loss functions impact the estimates generated by (1). These examples are constructed to isolate impact of the component of interest. In reality, the various parameters and function choices influence each other in nontrivial ways, which generally cannot be known *a priori*.

In the examples below, the plant model, whose state we are trying to estimate, consists of (5) and (6), where $C = \begin{bmatrix} 0 & 1 \end{bmatrix}$ and

$$A = \begin{bmatrix} \cos(\pi/500) & -\sin(\pi/500) \\ \sin(\pi/500) & \cos(\pi/500) \end{bmatrix}. \tag{23}$$

The the state is $\boldsymbol{\theta}_t \in \mathbb{R}^2$, its initial value is $\boldsymbol{\theta}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{w}_t \in \mathbb{R}^2$, and $v_t \in \mathbb{R}$. We assume that $\boldsymbol{w}_t$ and v_t satisfy the assumptions of a Kalman filter, and their covariances are $Q \in \mathbb{R}^{2\times 2}$ and $R \in \mathbb{R}$, respectively, where we detail their values in each example.

Varying the Gradient Descent Step Size η^{s} The parameter η^{s} influences how closely DMD adjusts the state estimate $\widehat{\boldsymbol{\theta}}_t$ to match the (possibly noisy) measurement versus trusting the predictions of the system model $\Phi(\cdot)$. In this example, we assume the plant model contains no process noise, i.e., $Q = 0$, and the measurement noise covariance is $R = 1$. The DMD model $\Phi(\widehat{\boldsymbol{\theta}}_t)$ is set to the plant model (5) and (6) excluding $\boldsymbol{w}_t$ and v_t. The divergence is set to $D(\boldsymbol{\theta} \| \widehat{\boldsymbol{\theta}}_t) = \frac{1}{2}\|\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t\|_2^2$ and the loss function is set to $\ell_t(\widehat{\boldsymbol{\theta}}_t) = \frac{1}{2}\|C\widehat{\boldsymbol{\theta}}_t - y_t\|_2^2$. The resulting closed-form measurement-based update (1) is

$$\widetilde{\boldsymbol{\theta}}_t = \widehat{\boldsymbol{\theta}}_t + \eta^{\mathrm{s}} C^{\mathrm{T}}(y_t - C\widehat{\boldsymbol{\theta}}_t). \tag{24}$$

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

Figure 3 presents the resulting time series of the second elements of $\boldsymbol{\theta}_t$, $\widehat{\boldsymbol{\theta}}_t^0$, and $\widehat{\boldsymbol{\theta}}_t^1$; we exclude time series of the first elements as they exhibit similar characteristics. The second term of (24) is 0 for $\widehat{\boldsymbol{\theta}}_t^0$, and so there is no adjustment to the state estimate based on the measurement. As a result, $\widehat{\boldsymbol{\theta}}_t^0$ matches $\boldsymbol{\theta}_t$ exactly because the model within DMD exactly matches the plant model. Alternatively, for $\widehat{\boldsymbol{\theta}}_t^1$, the convex program adjusts the state estimate to match the noisy measurements rather than trusting DMD's model, resulting in significant estimation error.

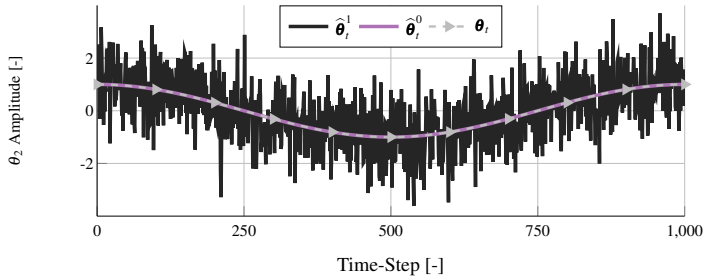


Fig. 3 Time series of the second element of $\boldsymbol{\theta}_t$, $\widehat{\boldsymbol{\theta}}_t^0$, and $\widehat{\boldsymbol{\theta}}_t^1$ for the example in Section 5

Varying the Choice of Divergence and Loss Functions The choice of the divergence and loss functions within DMD influences the algorithm's measurement-based adjustments. In this example, we vary DMD's measure-ment-based update by using two choices for the divergence and loss functions – one that includes covariance matrices explicitly and one that does not. We also simulate a Kalman filter to empirically show that the DMD estimates match those of a Kalman filter when the divergence and loss functions are constructed as described in Section 3.3.

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

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

We construct a steady-state discrete-time Kalman filter, whose estimates are denoted $\widehat{\boldsymbol{\theta}}_t^{\mathrm{KF}}$, using the underling system model and covariances. Both DMD formulations use $\eta^{\mathrm{s}} = 1$. The first DMD formulation, whose estimates are denoted $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$, uses the same loss function, divergence function, and resulting measurement-based update equation as in the previous example. The second DMD formulation, whose esti-

mates are denoted $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$, uses the divergence and loss functions needed to produce measurement-based updates that are equivalent to those of the Kalman filter, i.e., (15), and we set $\widehat{P}_t = \overline{P}$ and $\widehat{P}_t^{\mathrm{y}} = \left[C\overline{P}C^T + R\right]$. Note that the second formulation explicitly includes accurate model prediction error statistics via the covariances, whereas the first estimate implicitly assumes the covariances are identity matrices.

Figure 4 presents the time series of $\boldsymbol{\theta}_t$, $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$, $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ and $\widehat{\boldsymbol{\theta}}_t^{\mathrm{KF}}$. Note that the estimates $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ and $\widehat{\boldsymbol{\theta}}_t^{\mathrm{KF}}$ coincide exactly, empirically supporting our claim that we can choose the DMD model, divergence function, and loss function to achieve a measurement-based update equivalent to that of Kalman filter. In estimating the second element of $\boldsymbol{\theta}_t$, we first note that both $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$ and $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ follow the general trajectory of the true state, but $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ is noticeably smoother than $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$. By including the covariance matrices into the measurement-based update, $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ is better able to account for the measurement noise resulting in a less erratic estimate and reduced estimation error versus $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$. In estimating the first element of $\boldsymbol{\theta}_t$, both methods have significant deviations from the true state value; however, the root mean square estimation error in $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ is smaller that of $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$, indicating that $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$ is more accurate over the duration of the simulation. Again, the inclusion of accurate statistical information into the measurement-based update has led to a more accurate estimate.

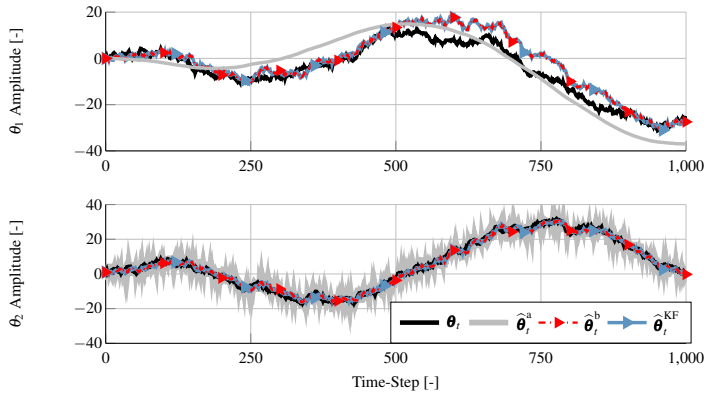


Fig. 4 Time series of $\boldsymbol{\theta}_t$, $\widehat{\boldsymbol{\theta}}_t^{\mathrm{a}}$, $\widehat{\boldsymbol{\theta}}_t^{\mathrm{b}}$, and $\widehat{\boldsymbol{\theta}}_t^{\mathrm{KF}}$ for the example in Section 5

It should be noted that this example was constructed such that the Kalman filter is the optimal estimator. In reality, the assumptions of the Kalman filter rarely hold, as in the case studies presented in Section 4.4. A Kalman filter can still be applied with varying degrees of success, but it may not be the optimal estimator. The DMD algorithm relaxes some of the underlying assumptions, which allows greater flexibility in designing the updates, but the theoretical guarantees of the Kalman filter do not apply. Additionally, it should be noted that in this example we assume we have a perfect estimate of the covariance matrices. In Section 4.4, we show that including inaccurate model prediction error statistics into the DMD algorithms within DFS

can degrade the performance of DFS, while including accurate statistics (which can be hard to obtain in practice) can substantially improve the estimation accuracy of DFS.

References

[1] Armel K, Gupta A, Shrimali G, Albert A (2013) Is disaggregation the holy grail of energy efficiency? The case of electricity. Energy Policy 52:213–234
[2] Baone C, Xu Y, Kueck J (2010) Local voltage support from distributed energy resources to prevent air conditioner motor stalling. In: Innovative Smart Grid Technologies (ISGT), Gothenburg, Sweden
[3] Beck A, Teboulle M (2003) Mirror descent and nonlinear projected subgradient methods for convex optimization. Operations Research Letters 31(3):167–175
[4] Berges M, Goldman E, Matthews H, Soibelman L (2009) Learning systems for electric consumption of buildings. In: International Workshop on Computing in Civil Engineering, Austin, TX
[5] Berges M, Goldman E, Matthews H, Soibelman L (2010) Enhancing electricity audits in residential buildings with nonintrusive load monitoring. Journal of Industrial Ecology 14(5):844–858
[6] Can Kara E, Kolter Z, Berges M, Krogh B, Hug G, Yuksel T (2013) A moving horizon state estimator in the control of thermostatically controlled loads for demand response. In: Proceedings of SmartGridComm, Vancouver, BC
[7] Cesa-Bianchi N, Long PM, Warmuth MK (1996) Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. IEEE Transactions on Neural Networks 7(3):604–619
[8] Dong R, Ratliff L, Ohlsson H, Sastry S (2013) A dynamical systems approach to energy disaggregation. In: Proceedings of the IEEE Conference on Decision and Control, Firenze, Italy
[9] Dong R, Ratliff L, Ohlsson H, Sastry S (2013) Energy disaggregation via adaptive filtering. In: Proceedings of the Allerton Conference, Monticello, IL
[10] Dong R, Ratliff L, Ohlsson H, Sastry S (2014) Fundamental limits of nonintrusive load monitoring. In: Proceedings of the HiCoNS, Berlin, Germany
[11] Duchi JC, Shalev-Shwartz S, Singer Y, Tewari A (2010) Composite Objective Mirror Descent. In: 23rd Annual Conference on Learning Theory (COLT), Haifa, Israel, pp 14–26
[12] Esmaeil Zadeh Soudjani S, Abate A (2015) Aggregation and control of populations of thermostatically controlled loads by formal abstractions. IEEE Transactions on Control Systems Technology 23(3):975–990
[13] Gonçalves H, Ocneanu A, Bergés M, Fan R (2011) Unsupervised disaggregation of appliances using aggregated consumption data. In: Proceedings of the 1st KDD Workshop on Data Mining Applications in Sustainability (SustKDD), San Diego, CA

[14] Grewal MS, Andrews AP (2015) Kalman filtering. Wiley
[15] Hall EC, Willett RM (2015) Online convex optimization in dynamic environments. IEEE Journal of Selected Topics in Signal Processing 9(4):647–662
[16] Hart G (2010) Nonintrusive appliance load monitoring. Proceedings of the IEEE 80(12):1870–1891
[17] Herbster M, Warmuth MK (1998) Tracking the best expert. Machine Learning 32(2):151–178
[18] Herbster M, Warmuth MK (2001) Tracking the Best Linear Predictor. Journal of Machine Learning Research 1(Sep):281–309
[19] Jadbabaie A, Rakhlin A, Shahrampour S, Sridharan K (2015) Online optimization: Competing with dynamic comparators. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, pp 398–406
[20] Kalsi K, Elizondo M, Fuller J, Lu S, Chassin D (2012) Development and validation of aggregated models for thermostatic controlled loads with demand response. In: Proceedings of Hawaii International Conference on Systems Science, Wailea, HI
[21] Kim H, Marwah M, Arlitt MF, Lyon G, Han J (2011) Unsupervised disaggregation of low frequency power measurements. In: Proceedings of the 2011 SIAM International Conference on Data Mining, vol 11, pp 747–758
[22] Kivinen J, Warmuth MK (1999) Averaging expert predictions. In: European Conference on Computational Learning Theory, Springer, pp 153–167
[23] Koch S, Mathieu J, Callaway D (2011) Modeling and control of aggregated heterogeneous thermostatically controlled loads for ancillary services. In: Proceedings of the Power Systems Computation Conference, Stockholm, Sweden
[24] Kolter J, Jaakkola T (2012) Approximate inference in additive factorial HMMs with application to energy disaggregation. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, La Palma, Canary Islands
[25] Kolter J, Batra S, Ng A (2010) Energy disaggregation via discriminative sparse coding. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), Vancouver, Canada
[26] Ledva G, Vrettos E, Mastellone S, Andersson G, Mathieu J (2015) Applying networked estimation and control algorithms to address communication bandwidth limitations and latencies in demand response. In: Hawaii International Conference on Systems Science (HICSS), Grand Hyatt, Kauai, HI
[27] Ledva GS, Balzano L, Mathieu JL (2015) Inferring the behavior of distributed energy resources with online learning. In: Proceedings of the Allerton Conference, Monticello, IL, pp 187–194
[28] Ledva GS, Balzano L, Mathieu JL (2017) Real-time energy disaggregation of a distribution feeders demand using online learning. (Under Review) Available at: https://arxiv.org/abs/1701.04389
[29] Lee MP, Aslam O, Foster B, Hou S, Kathan D, Pechman C, Young C (2015) Assessment of Demand Response & Advanced Metering. Staff report, Federal Energy Regulatory Commission, https://www.ferc.gov/legal/staff-reports/2015/demand-response.pdf

[30] Mathieu J, Callaway D (2012) State estimation and control of heterogeneous thermostatically controlled loads for load following. In: Proceedings of the Hawaii International Conference on Systems Science, Wailea, HI, pp 2002–2011

[31] Mathieu J, Gadgil A, Callaway D, Price P, Kiliccote S (2010) Characterizing the response of commercial and industrial facilities to dynamic pricing signals from the utility. In: Proceedings of ASME 2010 4th International Conference on Energy Sustainability, Phoenix, AZ

[32] Mathieu J, Koch S, Callaway D (2013) State estimation and control of electric loads to manage real-time energy imbalance. IEEE Transactions on Power Systems 28(1):430–440

[33] Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization. In: Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, New York

[34] NOAA (2009) NNDC Climatic Data Online, URL `http://www7.ncdc.noaa.gov/CDO/dataproduct`, Satellite and Information Service National Climate Data Center

[35] Pecan Street Inc (2016) Dataport. URL `https://dataport.pecanstreet.org/`

[36] Powers J, Margossian B, Smith B (1991) Using a rule-based algorithm to disaggregate end-use load profiles from premise-level data. IEEE Computer Applications in Power 4(2):42–47

[37] Schneider KP, Chen Y, Chassin DP, Pratt RG, Engel DW, Thompson SE (2008) Modern grid initiative distribution taxonomy final report. Tech. rep., Pacific Northwest National Laboratory (PNNL), Richland, WA (US)

[38] Shahrampour S, Jadbabaie A (2016) Distributed online optimization in dynamic environments using mirror descent. arXiv preprint arXiv:160902845

[39] Shalev-Shwartz S (2011) Online learning and online convex optimization. Foundations and Trends in Machine Learning 4(2):107–194

[40] Shao H, Marwah M, Ramakrishnan N (2012) A temporal motif mining approach to unsupervised energy disaggregation. In: Proceedings of the 1st International Workshop on Non-Intrusive Load Monitoring, Pittsburgh, PA, USA, vol 7

[41] Simonetto A, Mokhtari A, Koppel A, Leus G, Ribeiro A (2016) A class of prediction-correction methods for time-varying convex optimization. IEEE Transactions on Signal Processing 64(17):4576–4591

[42] Vrettos E, Mathieu J, Andersson G (2014) Control of thermostatic loads using moving horizon estimation of individual load states. In: Proceedings of the Power Systems Computation Conference (PSCC), Wroclaw, Poland

[43] Vrettos E, Mathieu J, Andersson G (2014) Demand response with moving horizon estimation of individual thermostatic load states from aggregate power measurements. In: Proceedings of the American Control Conference (ACC), Portland, OR

[44] Wytock M, Kolter J (2014) Contextually supervised source separation with application to energy disaggregation. In: Proceedings of the Conference on Artificial Intelligence (AAAI), Québec City, Canada, pp 486–492
[45] Yang T, Zhang L, Jin R, Yi J (2016) Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient. In: Proceedings of the 33rd International Conference on Machine Learning, New York, NY
[46] Zeifman M, Roth K (2011) Nonintrusive appliance load monitoring: review and outlook. IEEE Transactions on Consumer Electronics 57(1):76–84
[47] Zinkevich M (2003) Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the International Conference on Machine Learning (ICML), Washington DC
[48] Zoha A, Gluhak A, Imran M, Rajasegarar S (2012) Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. Sensors 12:16,838–16,866