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

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

Abstract— **Kalman filtering and online learning are two approaches to estimate the state of a system in the presence of inaccurate (e.g., noisy) measurements. While many online learning algorithms are model-free and data-driven, two recently developed online learning algorithms, Dynamic Mirror Descent (DMD) and Dynamic Fixed Share (DFS), incorporate dynamic models, similarly to Kalman filtering algorithms. Our previous work showed that DMD can be constructed to produce state estimates that are identical to those produced by a discrete-time Kalman filter. This work extends our previous work by exploring connections between a multiple model Kalman filter (MMKF) and DFS, which both incorporate a set of candidate models to address situations in which the underlying model is unknown. We show that the functions/parameters used within DFS can be constructed to produce the same estimates as a MMKF. We then modify DFS to include several heuristics that are used to improve the performance of a MMKF in order to assess whether they can also be used to improve the performance of DFS. Finally, we investigate the performance of the algorithms and their variations in a simulation study. Specifically, we seek to estimate the time-varying power consumption of an aggregation of electric loads, which could be used as the feedback signal within a demand response algorithm. The simulation results empirically show that DFS implementations generally perform better than comparable MMKF implementations since we are able to tune the functions/parameters used within DFS.**

I. INTRODUCTION

State estimation and online learning are two approaches to estimate the state of a dynamic system as measurements arrive at sequential, discrete time-steps. The discrete-time Kalman filter (hereafter referred to simply as a Kalman filter) relies on a model-based update that advances the state estimate in time according to an assumed model of the underlying system, and a measurement-based update that incorporates newly arrived measurements of the system into the state estimate. While a Kalman filter uses a single model of the system to estimate the state, a multiple model Kalman filter (MMKF) [1], [2] uses a set of possible models to compute the state estimate, addressing situations where the model is unknown beforehand. The Kalman filter and MMKF make assumptions on the generative model of the data, i.e., that the system model is linear and that each model's error and measurement noise are normally distributed. The assumptions lead to a fixed structure for the state estimation equations, and a user implementing the algorithms can only tune the model-based parameters within the update equations.

This research was funded by NSF Grant #ECCS-1508943.

The authors are with the Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. `gsledv@umich.edu; girasole@umich.edu; jlmath@umich.edu`

While many online learning algorithms are model-free and data-driven, two recently developed online learning algorithms, Dynamic Mirror Descent (DMD) and Dynamic Fixed Share (DFS) [3], incorporate dynamic models and use both a model-based update and a measurement-based update, similarly to Kalman filtering algorithms. DMD is an online learning analogue to a Kalman filter, and DFS is comparable to a MMKF. DMD and DFS both estimate a dynamic system state without making assumptions of the generative model of the data. These online learning algorithms are based on online convex optimization [4], where a user-defined convex optimization problem is solved online, or at each time-step, to update estimates using each new measurement as it arrives. These algorithms are more flexible and more data-driven than the Kalman filter algorithms. Flexibility arises from the user's ability to use a nonlinear system model within the algorithm and the ability for the user to design the online, convex optimization problem that dictates the way that a new measurement is incorporated into the estimate. The algorithms are more data-driven in the sense that historical data can be used do design this convex optimization problem as well as to select the form of the possibly nonlinear model(s) along with its (their) parameters. Within a Kalman filter, the user can tune the parameters within the update equations to improve performance, but within DMD, the user can tune the equations themselves (as well as the parameters) to improve performance. However, DMD and DFS are not guaranteed to be the best estimator from specific class of estimators, unlike a Kalman filter, which is the best linear estimator out of all linear estimators.

In this paper, we compare the multiple model algorithms, DFS and a MMKF, and apply them to a demand response simulation. Our previous work [5] showed that DMD can be constructed to produce identical state estimates to those produced by a Kalman filter. This work builds on this prior result and shows that the user-defined functions/parameters within DFS can be constructed to produce identical states estimates to those produced by a MMKF. We then modify DFS to include three heuristics that are used to improve the performance of a MMKF in order to assess whether they can also be used to improve the performance of DFS. While these heuristics result in a suboptimal MMKF, they often improve its estimation accuracy in practice, which we will demonstrate empirically.

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

The contributions of this paper are as follows: 1) we show that DFS can be constructed to produce identical estimates to those produced by a MMKF; 2) we incorporate three heuristics used within MMKFs into DFS; and 3) we compare the performance of DFS and a MMKF within the demand response simulation. This is in contrast with our previous works [5], [6], where [6] applied DMD and DFS to a feeder-level energy disaggregation problem, which sought to separate measurements of a distribution feeder's aggregate demand into parts in real-time and [5] drew parallels between DMD and Kalman filtering and showed that the design of the online convex program used within DMD/DFS strongly influences its estimation accuracy.

The remainder of this paper is organized as follows. Section II presents the general estimation problem; Section III summarizes the Kalman filter algorithms; Section IV summarizes the DMD and DFS algorithms; Section V reviews the method to construct the functions/parameters within DMD to achieve the same estimates as a Kalman filter from [5], derives a method to construct the functions/parameters within DFS to achieve the same estimates as a MMKF, and derives heuristic adjustments to DFS based on those commonly used in MMKFs; Section VI describes the demand response simulation study and its results; and Section VII concludes.

II. ESTIMATION PROBLEM

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

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

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

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

III. KALMAN FILTER ALGORITHMS

We next summarize the Kalman filter and MMKF algorithms to introduce notation. Section III-A describes the Kalman filter, and Section III-B describes the MMKF.

A. Kalman Filter

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

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

$$y_k = C_k\, x_k + v_k, \tag{2}$$

where (1) advances the state in time and (2) relates the state to a measurement. In these equations, $x_k \in \mathbb{R}^p$ is the state, $\omega_k \in \mathbb{R}^p$ is the process noise (or modeling error), $y_k \in \mathbb{R}^q$ is the output (or observation/measurement), and $v_k \in \mathbb{R}^q$ is the measurement noise. Samples of ω_k and v_k are assumed to be IID and to follow separate zero-mean, normal distributions. Their respective covariances Q_k and R_k are each symmetric, positive definite matrices. The system is assumed to be observable, and the matrices A_k, C_k, Q_k, and R_k are assumed to be known.

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

$$\widetilde{x}_k = \widehat{x}_k + \widehat{P}_k C_k^T \left[C_k \widehat{P}_k C_k^T + R_k \right]^{-1} \left(y_k - C_k \widehat{x}_k \right), \tag{3}$$

where $\widehat{P}_k$ is the estimation error covariance, which is known at each time-step. Equation (3) can be viewed as the solution to the following online, convex optimization problem [7]:

$$\min_{x, v_k} \quad v_k^{\mathrm{T}} R_k^{-1} v_k \,+\, (x - \widehat{x}_k)^{\mathrm{T}}\, \widehat{P}_k^{-1}\, (x - \widehat{x}_k) \tag{4}$$

$$\text{s.t.} \quad y_k = C_k x + v_k. \tag{5}$$

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

B. Multiple Model Kalman Filter

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

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

$$h(y_k|m^i) = \left[(2\pi)^{q/2}\sqrt{|\widehat{P}_k^{\text{y},i}|}\right]^{-1} \exp\left(-\frac{1}{2} d_k^{\text{y}}(\widehat{y}_k^i)\right) \tag{6}$$

$$w_{k+1}^i = \frac{h(y_k|m^i)\; w_k^i}{\sum_{j \in \mathcal{M}^{\text{mdl}}} h(y_k|m^j)\; w_k^j} \tag{7}$$

$$\widehat{x}_{k+1} = \sum_{i \in \mathcal{M}^{\text{mdl}}} w_{k+1}^i \; \widehat{x}_{k+1}^i, \tag{8}$$

with $i \in \mathcal{M}^{\text{mdl}}$ for (6) and (7). In the above, (6) is a conditional probability of the likelihood of the observation y_k given that the underlying system is model m^i, (7) is a weighting function where the weights can be viewed as a probability that model m^i matches the underlying system model, and (8) forms the overall estimate using the individual Kalman filter estimates and their weights.

Note that other algorithms exist for situations where the system models switches as time progresses, e.g., the first- and second-order generalized pseudo-Bayesian algorithms and the interacting multiple model algorithm, all from [2]. However, these algorithms require the probability of transitioning from one model to another during a time-step to be known *a priori*; here, we assume that this is unknown *a priori*. In addition, whereas online learning assumes that the underlying models (or "experts") operate independently, the first- and second-order generalized pseudo-Bayesian algorithms and the interacting multiple model algorithm do not make this assumption. As a result, we do not include further discussion of these algorithms.

IV. ONLINE LEARNING ALGORITHMS

In this section we summarize the DMD and DFS algorithms, which were originally developed in [3]. Section IV-A describes DMD, and Section IV-B describes DFS.

A. Dynamic Mirror Descent

The DMD algorithm uses a user-defined convex optimization formulation and a single model of the underlying system to estimate its state. Specifically, DMD uses the convex optimization formulation to adjust the estimate $\widehat{x}_k$ using the new measurement y_k, and then applies the model to advance the adjusted estimate to the next time-step. The DMD algorithm formulation is [3]

$$\widetilde{x}_k = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\; \eta^{\text{s}} \left(\nabla \ell_k(\widehat{x}_k)\right)^T \; x + D\left(x \| \widehat{x}_k\right) \tag{9}$$

$$\widehat{x}_{k+1} = \Phi(\widetilde{x}_k), \tag{10}$$

where (9) computes the adjusted estimate and (10) applies the model. In these equations, $\Phi(\cdot)$ is the (possibly nonlinear) model, $\eta^{\text{s}} > 0$ is a user-defined step size, and $\nabla \ell_k(\widehat{x}_k)$ is the gradient or subgradient of the convex loss function $\ell_k(\widehat{x}_k)$, which computes the error on the output estimate. The function $D(x\|\widehat{x}_k)$ is a Bregman divergence, which is similar to a distance function. As an example, we could use $D(x\|\widehat{x}_k) = \|x - \widehat{x}_k\|_2^2$ and $\ell_t(\widehat{x}_k) = \|C\widehat{x}_k - y_k\|_2^2$, where the matrix C translates the state estimate into an output estimate. The choice of loss function establishes the relationship between output estimate errors and state estimate errors, since the gradient of this function helps to determine how the state estimate is adjusted based on the output estimate errors. The choice of the Bregman divergence helps to establish the relationship between estimation errors within the different components of the state. The parameter η^{s} controls how closely the algorithm matches the output estimates with the measurements (by adjusting the state estimate) versus trusting the models.

B. Dynamic Fixed Share

The DFS algorithm uses a set of DMD algorithms, each using a separate (possibly nonlinear) model from $\mathcal{M}^{\text{mdl}}$, as experts (i.e., algorithms that generate estimates) into the Fixed Share Algorithm developed in [8]. Similar to a MMKF, DFS forms an overall state estimate from the N^{mdl} experts. The DFS algorithm's weight updates and overall estimate are those of the Fixed Share Algorithm where the estimates $\widehat{x}_k^i$ for $i \in \mathcal{M}^{\text{mdl}}$ are generated using DMD:

$$w_{k+1}^i = \frac{\lambda}{N^{\text{mdl}}} + (1-\lambda)\; \frac{w_k^i \; \exp\left(-\eta^r \; \ell_k\left(\widehat{x}_k^i\right)\right)}{\sum_{j=1}^{N^{\text{mdl}}} w_k^j \; \exp\left(-\eta^r \; \ell_k\left(\widehat{x}_k^j\right)\right)} \tag{11}$$

$$\widehat{x}_{k+1} = \sum_{i \in \mathcal{M}^{\text{mdl}}} w_{k+1}^i \; \widehat{x}_{k+1}^i \tag{12}$$

with $i \in \mathcal{M}^{\text{mdl}}$ for (11). Equation (11) updates the weight of each expert, where w_k^i is the weight of expert i, $\lambda \in (0,1)$ is a user-defined parameter that sets the minimum weight of each expert, and η^{r} is a user-defined parameter that scales the total accumulated loss (which is related to output estimation errors). Equation (12) combines the individual estimates into an overall estimate $\widehat{x}_k$. Setting η^{r} to larger values forces $\exp\left(-\eta^r \; \ell_k\left(\widehat{x}_k^i\right)\right)$ to be near one regardless of $\ell_k\left(\widehat{x}_k^i\right)$, and this results in faster changes to the weights.

V. Connections Between the Kalman Filtering and Online Learning Algorithms

Section V-A reviews the method developed in [5] to construct the functions/parameters within DMD to produce estimates identical to those produced by a Kalman filter. Section V-B builds on this result and presents our main result: a method to construct the functions/parameters used within DFS to produce estimates identical to those produced by a MMKF. Finally, Section V-C adapts several heuristics commonly used within MMKFs to DFS.

A. Producing Identical Estimates with DMD and a Kalman Filter

We construct DMD by choosing the model, user-defined parameters, and user-defined functions. As with a Kalman filter, we assume as linear model and that the model matrices A_k, C_k, Q_k, and R_k are known, and additionally, that $\widehat{P}_k$ is known. Choosing the model used within DMD to be identical to that used within the Kalman filter results in the same model-based update. The remaining step is to construct the convex program (9) such it corresponds to (3).

In (9) we have the ability to choose the η^{s}, $D(x\|\widehat{x}_k)$, and $\ell_k(\widehat{x}_k)$. Recall that the measurement-based update in a Kalman filter is

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

and the measurement-based update in DMD is

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

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

$$\widetilde{x}_k = \widehat{x}_k + \widehat{P}_k\left(-\nabla\ell_k(\widehat{x}_k)\right). \tag{13}$$

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

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

and then $-\nabla\ell_k(\widehat{x}_k) = C_k^T(\widehat{P}_k^{\mathrm{y}})^{-1}\left(y_k - C_k\widehat{x}_k\right)$. Plugging $-\nabla\ell_k(\widehat{x}_k)$ into (13) gives the same measurement-based update as the Kalman filter.

B. Producing Identical Estimates with DFS and a MMKF

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

$$h^{\mathrm{DFS}}(y_k|m^i) = \exp\left(-\ell_k(\widehat{x}_k^i)\right) \tag{14}$$

$$w_{k+1}^i = \frac{h^{\mathrm{DFS}}(y_k|m^i)\; w_k^i}{\sum_{j\in\mathcal{M}^{\mathrm{mdl}}} h^{\mathrm{DFS}}(y_k|m^j)\; w_k^j} \tag{15}$$

for $i \in \mathcal{M}^{\mathrm{mdl}}$. Note that (15) corresponds exactly to (7), and the remaining step is to construct $h^{\mathrm{DFS}}(y_k|m^i)$ to equal $h(y_k|m^i)$.

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

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

$$1 = \frac{1}{(2\pi)^{q/2}\sqrt{|\beta^i\widehat{P}_k^{\mathrm{y},i}|}} \tag{16}$$

$$\implies \beta^i = \sqrt[q]{(2\pi)^{-q}\,|\widehat{P}_k^{\mathrm{y},i}|^{-1}}. \tag{17}$$

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

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

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

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

$$\overline{K}_k^i = \widehat{P}_k^i\alpha^i(C_k^i)^T\left[\alpha^i C_k^i\widehat{P}_k^i\alpha^i(C_k^i)^T + \beta^i R_k^i\right]^{-1} \tag{18}$$

$$= \frac{\alpha^i}{\beta^i}K_k^i. \tag{19}$$

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

$$\widetilde{x}_k^i = \widehat{x}_k^i + \frac{\alpha^i}{\beta^i}K_k^i\left(\alpha^i y_k - \alpha^i C_k^i\widehat{x}_k^i\right) \tag{20}$$

$$= \widehat{x}_k^i + K_k^i\left(y_k - C_k^i\widehat{x}_k^i\right) \tag{21}$$

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

C. Adapting MMKF Heuristics to DFS

We next show that several heuristic adjustments that are commonly used within MMKFs can be readily incorporated into DFS by modifying the weight update. The heuristics include 1) setting a minimum weight such that a model's weight does not go to zero [2], 2) using exponential decay within the likelihood function [9], and 3) using a sliding window within the likelihood function [9].

To incorporate these methods into DFS, we can change the weighting equation (11). The first heuristic is equivalent to setting λ to a value greater than zero within DFS. The second heuristic is equivalent to adjusting (11) to

$$w_{k+1}^{i} = \frac{\lambda}{N^{\text{mdl}}} + (1-\lambda)\frac{(w_k^i)^{\gamma}\,\exp\left(-\eta^r \ell_k(\widehat{x}_k^i)\right)}{\sum_{j=1}^{N^{\text{mdl}}} (w_k^j)^{\gamma}\,\exp\left(-\eta^r \ell_k(\widehat{x}_k^j)\right)}. \tag{22}$$

The parameter $\gamma \in (0,1)$ reduces the impact of the previously accumulated loss on the model's weight, where smaller values of γ reduce the effect more dramatically. The third method is equivalent to adjusting (11) to

$$w_{k+1}^{i} = \frac{\lambda}{N^{\text{mdl}}} + (1-\lambda)\,\frac{\prod_{t=k-N^{\ell}}^{k} \exp\left(-\eta^r\,\ell_t\left(\widehat{x}_t^i\right)\right)}{\sum_{j=1}^{N^{\text{mdl}}} \prod_{t=k-N^{\ell}}^{k} \exp\left(-\eta^r\,\ell_t\left(\widehat{x}_t^j\right)\right)}, \tag{23}$$

where N^{ℓ} is the number of time-steps within the sliding window. By using (22) and (23), it is possible to discount and exclude historical estimation errors (and their resulting losses), which leads to a more dynamic set of weights that depend on the recent estimation accuracy. Simulations presented in Section VI investigate how these weighting functions affect the overall estimates within DFS and a MMKF.

VI. CASE STUDIES

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

A. Problem Setup and Simulation Details

Figure 1 gives the block diagram of the estimation problem. The plant is our representation of the physical system. It consists of a set of n^{AC} residential AC models along with other loads within a distribution network. The demand response provider would like an estimate of the aggregate AC demand (i.e., the flexible demand), but it only has a measurement of the total demand. It subtracts an estimate of the demand of the other loads from the measurement of the total demand to obtain a noisy estimate of the aggregate AC demand. This noisy estimate is used as a measurement within the DFS/MMKF algorithms to obtain a better estimate of the aggregate AC demand.

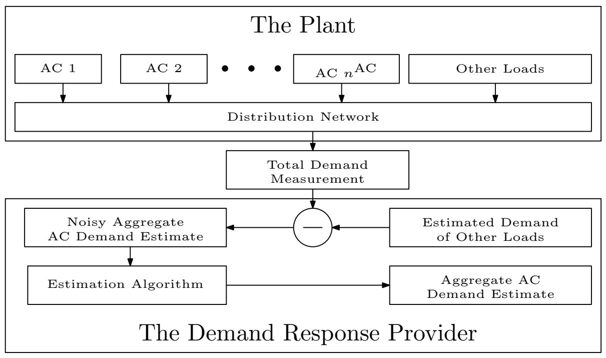

The Plant
AC 1
AC 2
AC n^{AC}
Other Loads
Distribution Network
Total Demand Measurement
Noisy Aggregate AC Demand Estimate
Estimated Demand of Other Loads
Estimation Algorithm
Aggregate AC Demand Estimate
The Demand Response Provider

Fig. 1. Block diagram of the estimation problem.

Each AC within the plant is modeled with the following equations [10]:

$$\theta_{k+1} = a\,\theta_k + (1-a)\left(\theta_k^{\text{o}} - m_k \Lambda P\right) \tag{24}$$

$$m_{k+1} = \begin{cases} 0 & \text{if } \theta_{k+1} < \theta^{\text{set}} - \frac{\theta^{\text{db}}}{2} \\ 1 & \text{if } \theta_{k+1} > \theta^{\text{set}} + \frac{\theta^{\text{db}}}{2} \\ m_k & \text{otherwise,} \end{cases} \tag{25}$$

where $\theta_k \in \mathcal{R}$ is the internal air temperature of the house, $m_k \in \{0,1\}$ is the AC's on/off switch value, $\theta_k^{\text{o}} \in \mathcal{R}$ is the time-varying outdoor temperature, $a = \exp(-\Delta t/\Lambda C)$, and $\Delta t \in \mathcal{R}$ is the time-step. The remaining parameters are sampled from uniform distributions with ranges from [11], where $\theta^{\text{set}} \in \mathcal{R}$ is the temperature set-point, $\theta^{\text{db}} \in \mathcal{R}$ is the temperature dead-band, $\Lambda \in \mathcal{R}$ is the thermal resistance, $C \in \mathcal{R}$ is the thermal capacitance, $\eta \in \mathcal{R}$ is the coefficient of performance, and $P \in \mathcal{R}$ is the energy transfer rate. The aggregate power draw of the set of ACs is $y_k^{\text{Agg}} = \sum_{i=1}^{n^{\text{AC}}} m_k^i P^i (\eta^i)^{-1}$. In this work, we assume the estimation error v_k associated with the demand of the other loads is normally distributed with variance R. Therefore, the noisy aggregate AC demand estimate (i.e., the measurement used within the DFS and MMKF algorithms) is $y_k = y_k^{\text{Agg}} + v_k$.

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

Each algorithm uses a set of dynamic models, developed in [11], [12], that capture the aggregate AC demand. Each model is a linear, time-invariant autonomous system $x_{k+1} = Ax_k$, $y_k = Cx_k$, where the state $x_k \in \mathbb{R}^2$ captures the portion of ACs that are on versus off, $A \in \mathbb{R}^{2\times 2}$ is a transposed Markov transition matrix, and $C \in \mathbb{R}^{1\times 2}$ multiplies the the portion of ACs switched on by a scalar, resulting in the aggregate AC demand $y \in \mathbb{R}$. Both A and C are a function of the outdoor temperature. We use $N^{\text{mdl}} = 3$ aggregate models, where A and C are identified by simulating a set of ACs with the same parameter distributions as those within the plant at outdoor temperatures $\theta^{\text{o}} = 28$, 31, and 34°C. We denote these aggregate models as m^{28}, m^{31}, and m^{34}.

The algorithm implementations are summarized in Table I.

TABLE I

SUMMARY OF ALGORITHMS AND THEIR RMS ESTIMATION ERRORS (KW)

Abbreviation	Details	RMS Error (kW)
MMKF	The standard MMKF algorithm without any heuristics	104.5
MMKF-S	A MMKF with scaling performed according to Section V-B	104.5
MMKF-M	A MMKF using a minimum weight for each model, i.e., using an equation similar to (11) as the weight update	63.4
MMKF-W	A MMKF using an exponential decay weight update and a minimum weight for each model	-
MMKF-E	A MMKF using a sliding window weight update and a minimum weight for each model	61.1
DFS-S	DFS with scaling performed according to Section V-B	104.5
DFS-M	DFS with the standard weight update (11), which includes a minimum weight for each model	61.9
DFS-W	DFS using the sliding window weight update (23), which includes a minimum weight for each model	61.4
DFS-E	DFS using the exponential decay weight update (22), which includes a minimum weight for each model	60.9

We choose the functions within DFS such that the updates resemble the updates of a MMKF. However, for DFS-M, DFS-W, and DFS-E, we do not use the scaling from Section V-B and we tune η^{r} (which does not appear in the MMKF weight update) resulting in different performance between the comparable DFS and MMKF algorithms. By comparing MMKF and MMKF-S we see that the MMKF scaling in Section V-B achieves approximately the same performance in this case, and by comparing MMKF-S and DFS-S we can verify that the algorithms produce identical updates. We set $\lambda = 1e - 5$ in DFS-M, DFS-W, DFS-E, and MMKF-M (which uses a weight update similar to (11)), which allows a single model to dominate the overall estimate if one proves to be the most accurate. We set η^{r} to 0.8 in DFS-M, 0.5 in DFS-W, and 1.2 in DFS-E. We set the window duration to $N^{\ell} = 250$ time-steps in DFS-W and MMKF-W. We set the exponential decay parameter to $\gamma = 0.995$ in DFS-E and MMKF-E. The values of η^{r}, N^{ℓ}, and γ were tuned qualitatively in the given simulation scenario to provide weights that are responsive but not overly erratic, e.g., from measurement noise.

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

B. Results

Figure 2 presents time series for the MMKF-S and DFS-S. Figure 3 presents time series of the MMKF and DFS-M estimates along with the total AC demand; we exclude time series of the other algorithm implementations as they are difficult to distinguish from one another. Figure 4 presents time series of the Kalman filter estimates obtained using each model along with the aggregate AC demand. Table I summarizes the RMS error in kW of the aggregate AC demand estimates for each algorithm implementation, and Fig. 5 presents time series of the weights for various algorithm implementations. Note that in Fig. 5 we exclude weight time series for MMKF-W as results could not be computed due to numerical issues, and we exclude weight time series for MMKF-E as they are similar to those of DFS-E. The numerical issues with MMKF-W arise due to the coefficient

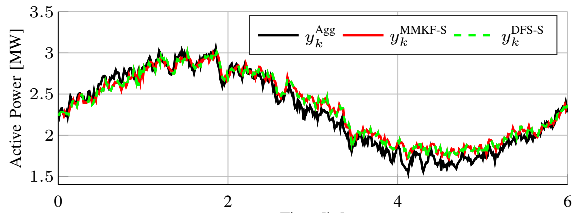


Fig. 2. Time series of the aggregate AC demand, denoted y_k^{Agg}, versus the MMKF-S and DFS-S estimates, denoted $y_k^{\mathrm{MMKF\text{-}S}}$ and $y_k^{\mathrm{DFS\text{-}S}}$, respectively.

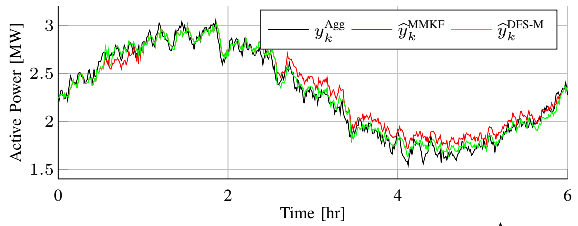


Fig. 3. Time series of the aggregate AC demand, denoted y_k^{Agg}, versus the MMKF and DFS-M estimates, denoted $\widehat{y}_k^{\mathrm{MMKF}}$ and $\widehat{y}_k^{\mathrm{DFS\text{-}M}}$, respectively.

in front of the exponential in (6), which results in values that are approximately zero when computing a windowed weight, i.e., using an update that is similar to (23).

From Fig. 2, we can see that the estimates for MMKF-S and DFS-S are the same. From Table I, we can see that the RMS estimation errors of MMKF, MMKF-S, and DFS-S are the same, which empirically validates the equivalence established via scaling the output equations in Section V-B and demonstrates that, in this case, the MMKF scaling achieves approximately the same result as without scaling.

From Table I, we can see that the MMKF performance can be improved with heuristics. In general, the DFS implementations slightly outperform the comparable MMKF implementations. It is unsurprising that the results are similar due to the similarities in the MMKF and DFS algorithms. Tuning the covariance matrices within the underlying Kalman filters may improve the performance of all implementations, but this improvement would be identical across all implemen-

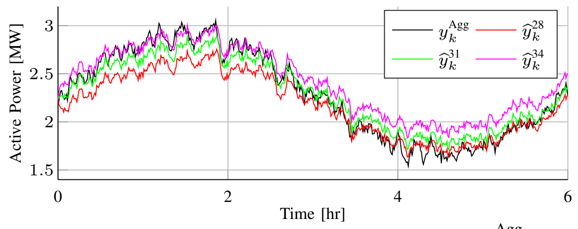
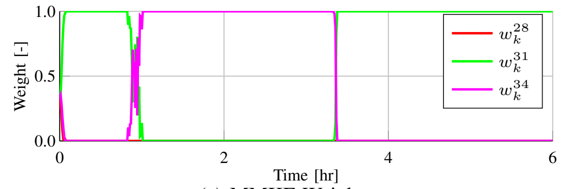
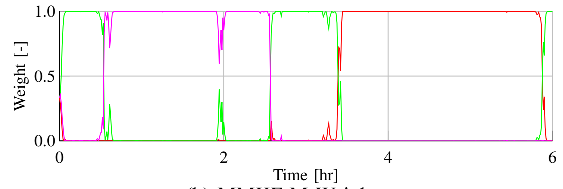


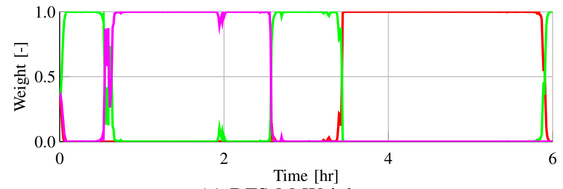
Fig. 4. Time series of the aggregate AC demand, denoted y_k^{Agg}, versus the individual, underlying Kalman filter estimates, denoted $\widehat{y}_k^{28}$, $\widehat{y}_k^{31}$, and $\widehat{y}_k^{34}$ for models m^{28}, m^{31}, and m^{34}, respectively.



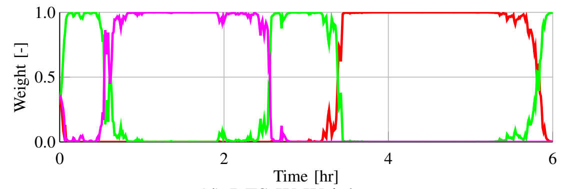
(a) MMKF Weights



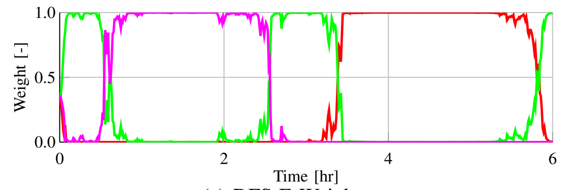
(b) MMKF-M Weights



(c) DFS-M Weights



(d) DFS-W Weights



(e) DFS-E Weights

Fig. 5. Time series of the weights for MMKF, MMKF-M, DFS-M, DFS-W, and DFS-E where w_k^{28}, w_k^{31}, and w_k^{34} denote the weights for m^{28}, m^{31}, and m^{34}, respectively

tations as they all use the same underlying estimates. The improvement in DFS is due to the parameter η^{r}, which allows the algorithm to generate more dynamic weights by setting η^{r} to larger values. This is not possible in a MMKF.

Comparing the various time series of the weights to the accuracy of the underlying Kalman filter estimates illuminate the differences in algorithm performance. Specifically, the MMKF weights presented in Fig. 5a show that the MMKF does not ever weight m^{28} heavily, even though the model is accurate over the final two hours of the simulation. This is because m^{28} was inaccurate over the early portion of the simulation, resulting in a low likelihood and a low weight, and it was unable to regain weight once it became accurate. Including a minimum weight into the MMKF overcomes this issue, as can be seen in Fig. 5b, which shows the weight time series for MMKF-M. In contrast, the DFS algorithm weights m^{28} heavily in the final two hours of the simulation, as can be see in Fig. 5c.

Another characteristic of the DFS-M, MMKF, and MMKF-M weights are that they become smoother as the simulation progresses. This is because the weights sum the losses (related to the output estimation errors) as the simulation progresses, and the weights become more stagnant as the losses accrue. Alternatively, the behavior of the DFS-W and DFS-E weights in Fig. 5d and Fig. 5e, respectively, are more consistent throughout the simulation. The MMKF-E weights behave similar to those of DFS-E. This is because the recent losses have larger influence on the weights. As a result, the weights are able to react to the models' recent performance. Incorporating a sliding window or exponential decay into the weight function performs a similar function, which results in similar weights. This can be seen in that DFS-W and DFS-E weights are almost identical and result in very similar RMS error values.

VII. Conclusions

In this paper, we show that DFS can produce updates that are identical to that of a MMKF. We also showed that DFS can be modified to incorporate heuristics that are commonly used within a MMKF. We applied various implementations of DFS and a MMKF to a demand response simulation. This simulation scenario empirically validated the scaling that is used to produce identical updates between DFS and a MMKF. We showed that including a minimum weight threshold improves the performance of DFS and a MMKF. We also showed that including exponential decay or a sliding window in the DFS or MMKF weight update allows more consistent, responsive behavior in the weights. In addition, the minimum weight, exponential decay, and sliding window versions of the DFS and MMKF algorithms effectively estimate the aggregate AC demand within the demand response simulation.

References

[1] D. Simon, "Optimal state estimation," in *Kalman, H Infinity, and Nonlinear Approaches, Wiley & Sons*, 2006.

[2] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[3] 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.

[4] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[5] G. Ledva, Z. Du, L. Balzano, and J. Mathieu, "Disaggregating load by type from distribution system measurements in real-time," in *Energy Markets and Responsive Grids*, I. Hiskens, S. Meyn, T. Samad, and J. Stoustrup, Eds. New York: Springer, (in press).

[6] G. S. Ledva, L. Balzano, and J. L. Mathieu, "Real-time energy disaggregation of a distribution feeder's demand using online learning," *IEEE Transactions on Power Systems*, (In Press).

[7] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal processing magazine*, vol. 27, no. 3, pp. 50–61, 2010.

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

[9] G. Welch and G. Bishop, "An introduction to the Kalman filter," http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf, August 2001, SIGGRAPH 2001 Course 8.

[10] R. Mortensen and K. Haggerty, "Dynamics of heating and cooling loads: models, simulation, and actual utility data," *IEEE Transactions on Power Systems*, vol. 5, no. 1, pp. 243–249, 1990.

[11] 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.

[12] S. Koch, J. Mathieu, and D. Callaway, "Modeling and control of aggregated heterogeneous thermostatically controlled loads for ancillary services," in *Power Systems Computation Conference (PSCC)*, Stockholm, Sweden, Aug. 2011.