Backward Approximate Dynamic Programming with Hidden Semi-Markov Stochastic Models in Energy Storage Optimization

Joseph L. Durante

Department of Electrical Engineering, Princeton University, Princeton, NJ 08540, jdurante@princeton.edu

Juliana Nascimento

Department of Operations Research, Princeton University, Princeton, NJ 08540, jnascime@princeton.edu

Warren B. Powell

Department of Operations Research, Princeton University, Princeton, NJ 08540, powell@princeton.edu

We consider an energy storage problem involving a wind farm with a forecasted power output, a stochastic load, an energy storage device, and a connection to the larger power grid with stochastic prices. Electricity prices and wind power forecast errors are modeled using a novel hidden semi-Markov model that accurately replicates not just the distribution of the errors, but also crossing times, capturing the amount of time each process stays above or below some benchmark such as the forecast. This is an important property of stochastic processes involved in storage problems. We show that we achieve more robust solutions using this model than when more common stochastic models are considered. The new model introduces some additional complexity to the problem as its information states are partially hidden, forming a partially observable Markov decision process. We derive a near-optimal time-dependent policy using backward approximate dynamic programming, which overcomes the computational hurdles of classical (exact) backward dynamic programming, with higher quality solutions than the more familiar forward approximate dynamic programming methods.

Key words: Backward Approximate Dynamic Programming, Crossing State Hidden Semi-Markov Model, Energy Storage Optimization

History: Version as of October 10, 2017

1. Introduction

Renewable energy sources that exhibit high volatility and intermittency, such as wind or solar, are often paired with energy storage devices to improve the efficiency and reliability of the energy systems that incorporate them. To realize the full potential of the system, we must optimize the control policy to determine the best possible energy allocation decisions in an uncertain environment. Energy storage problems appear in many variations, each with its own unique challenges, characteristics (such as the nature of the stochastics involved), and best approach to optimization. In this paper we consider the problem of satisfying a time-varying load through a combination of energy from a storage device, a highly volatile wind power source, and the larger power grid with a highly volatile and heavy tailed locational marginal electricity price (LMP) at maximum profit.

Our approach to optimizing the control policy for the system involves formulating the problem as a Markov decision process with accurately modeled stochastic processes and finding solutions based on a novel type of approximate dynamic programming (ADP) that we refer to as *backward approximate dynamic programming.*

The paper first focuses on carefully modeling the stochastics involved in the problem, ensuring that any policies resulting from optimizing the system model would also perform well in practice. Wind power generation is modeled with the univariate crossing state hidden semi-Markov model (HSMM) presented in Durante et al. (2017) which is unique in its ability to capture both crossing time distributions and the distribution of errors from forecast. Crossing times, which are contiguous blocks of time for which a stochastic process is above or below some reference series (in this case a forecast) are not well replicated by standard autoregressive models which have been popular in the study of energy storage models (e.g. Löhndorf and Minner 2010, Zhou et al. 2013). Additionally, the distribution of areas above and below the forecast (the surpluses and deficits of energy produced versus expected output) are accurately replicated by the crossing state model.

Characteristics such as crossing time, error, and area distributions are especially important to model in the context of an energy storage problem as they inform choices such as storage device capacity, type, and charge/discharge rate. Furthermore, properly modeling the above behaviors influences the effectiveness of the control policy itself. Consider optimizing the system, whether through policy search, ADP, or another method, using a wind power model whose crossing times are too short compared to the true wind power production crossing time behavior. This will result in control policies that do not account for the possibility that wind will underperform expectations for extended periods of time. Thus, in practice, the resulting policy will not be robust as performance will likely suffer in these scenarios.

We extend the HSMM to model stochastic electricity prices in the real-time market by incorporating the temperature forecast as an explanatory variable and using a daily periodic reference series. This produces very realistic sample paths of electricity prices, which can be particularly difficult to replicate given their heavy-tailed behavior and correlation with temperature.

The energy storage problem is formulated as a discrete time, finite horizon Markov decision process (MDP) in which a control decision must be made at each time step. In smaller, low dimensional problems, an optimal policy can be found using vanilla backward dynamic programming to compute value functions for each possible state. However, when a realistic system model is considered that incorporates more sophisticated stochastic models, the curses of dimensionality that arise in either the decision space, the state space, or the outcome space make performing a full backward pass computationally intractable. Specifically, when using HSMMs for the stochastics, the MDP becomes a partially observable MDP which introduces additional complexity into the problem.

This paper adapts backward ADP techniques for use in energy storage problems to overcome these curses of dimensionality. Unlike classical forward approximate dynamic programming, which estimates value functions while stepping forward in time (sometimes with a backward traversal), backward ADP performs a single backward pass, as done in standard backward dynamic programming, but then fits an approximate model based on a small sample of the states. In this setting, backward ADP produces higher quality solutions than the more familiar forward ADP methods.

Furthermore, backward ADP and crossing state models are highly compatible as the models have a small (and fixed) number of compact post-decision information states for indexing the value function approximations. This allows for a reduction in both computation time and memory required to store value functions. By using backward ADP with the crossing state models, we can create a more realistic energy storage problem model and still find near-optimal control policies.

This paper makes the following contributions: 1) The exogenous processes involved in the co-located wind farm-energy storage device problem are modeled with a new hidden semi-Markov model that accurately replicates both crossing time and error distributions. 2) We propose for the first time the use of backward ADP using regression models, extending prior work by Cai and Judd (2010) and Cheng et al. (2017), to higher dimensional problems, providing a robust complement to more classical forward ADP methods. 3) We extend the basic methodology of backward ADP to handle the hidden state variable of the hidden semi-Markov model, which required developing a more compact state representation, and the design of Bayesian updating logic. 4) We show that the backward ADP methodology produces higher quality results, more consistently, than forward ADP methods for energy storage problems, and further demonstrate that training on the hidden semi-Markov model produces more robust policies than standard stochastic models that have been used in the past.

The paper is organized as follows. Section 2 provides a brief literature review of energy storage problems, common stochastic models used in these problems, and algorithmic strategies related to backward ADP. Section 3 provides a thorough discussion regarding the modeling of the stochastic wind and price processes. Section 4 formally describes the energy storage problem by defining the five elements of the stochastic optimization problem. The backward ADP algorithms used in this paper are presented in Section 5. Numerical results comparing backward ADP to other policy types are reported in Section 6, while results highlighting the impact of model selection on policy effectiveness and robustness are presented in Section 7. The paper is concluded in Section 8.

2. Literature Review

Energy storage optimization is a widely researched topic with many problem variations. We review some of the problems and configurations that have been considered, organized by solution strategy.

Powell (2016) describes four basic strategies for developing control policies for these systems: policy function approximations (PFAs), policies based on cost function approximations (CFAs), direct lookahead policies (DLAs), and policies based on value functions approximations (VFAs). PFAs map a state directly to a feasible action. CFAs maximize a parameterized approximation of a cost function subject to parametrically modified constraints. Both rely on policy search to optimize any parameters involved. DLAs maximize over both current and future actions based on an approximate model of the system; both deterministic and stochastic lookaheads fall in this class. VFAs maximize the one-step contribution of an action plus an approximation of the value of being in a future state. Extra attention is paid to VFA-based solution strategies as backward ADP belongs to this class.

Affine policies, which belong to the PFA class, are often utilized for control in the energy storage domain. In one example, Warrington et al. (2012) utilize affine policies to robustly control power system components such as storage devices and fast-ramping generators (e.g. coal and gas generators) in an intraday scheduling problem. The affine policy maps a state directly to an action in response to wind power forecast errors. In another case, Taylor et al. (2013) consider the use of affine policies for optimizing the control of co-located renewable generation-storage systems in a competitive market. Training an artificial neural network (ANN) to make control decisions based on the system state is also a form of a PFA. This is done in Han et al. (2016) where an ANN is used to allocate energy from a wind farm, storage device, and the power grid to satisfy a time-varying load. Han et al. (2016) explores the use of ANNs to make control decisions for a higher dimensional storage problem as well.

A popular form of CFA is to use a deterministic lookahead (which can accommodate forecasts), with tunable parameters to handle uncertainty. This approach is used for robust power system control in Simao et al. (2017) where reserves are explicitly tuned to meet the variability of renewables. Similarly, in Thalassinakis and Dialynas (2004), a Monte Carlo simulation method tunes the reserve level in addition to other power system settings.

As previously mentioned, deterministic lookahead policies belong to the DLA class and are a commonly used approach for energy system control. In one example, Denholm and Sioshansi (2009) study the benefits of co-locating wind farms and storage devices (compressed air storage is considered) to reduce transmission requirements instead of placing storage devices on the load side of the system. A deterministic lookahead is utilized as the storage operator makes decisions based on a mixed integer program with a forecast for a two-week horizon into the future. Model predictive control (MPC) approaches, which typically consider deterministic models of the future, also belong to the DLA class. See Camacho and Alba (2013) for details on MPC. In the energy storage domain, MPC is often used for the efficient operation of heating and cooling systems for large buildings as in Ma et al. (2012). In a different application, Arnold and Andersson (2011)

utilize MPC to operate a storage hub containing both battery and hot water storage devices to satisfy the loads from aggregated households at minimum cost. This is done in the presence of uncertain renewable sources, electricity prices, and natural gas prices. Stochastic lookaheads are also seen in the literature, such as in Garcia-Gonzalez et al. (2008) where the two-stage stochastic programming method is used to jointly optimize the bids of a wind farm and pumped storage facility in the day-ahead market based on electricity price and wind power scenarios. Two-stage stochastic programming is also used to attempt to tackle much higher dimensional problems in the energy systems realm such as robustly optimizing a large power grid in the day-ahead unit commitment problem (Wang et al. 2013).

VFA-based policies are most often associated with dynamic programming or approximate dynamic programming approaches. An example of approximate dynamic programming in the energy storage domain is seen in Schneider et al. (2015), where an ADP method is used to optimize battery charging accross a network of electic vehical battery swap stations. Another example of a VFA-based approach using forward ADP is stochastic dual decomposition procedure (SDDP), initially described by Pereira and Pinto (1991), where thermal generation is planned in a large power network under uncertain hydro-power production conditions. In this formulation, the stochastics involved are assumed to exhibit stage-wise, or intertemporal, independence. Other multistage stochastic programming algorithms such as the cut sharing algorithm from Infanger and Morton (1996) relax this assumption, allowing for interstage dependency. Another algorithm that does so is the approximate dual dynamic programming (ADDP) method presented in Löhndorf et al. (2013). It is used in Löhndorf et al. (2013) to optimize both bidding and storage decisions in a hydro storage system with a network of reservoirs with uncertainty in both electricity price and environmental conditions. Stochastic prices are modeled using separate linear models for each hour of the day based on state-of-the-world variables such as the electricity demand and wind power production. Meanwhile, the time t wind power mean is determined by a trigonometric regression allowing for trend and seasonal components. The errors (the stochastic component) from this time-varying mean are modeled as a first-order Markov chain.

A similar ADDP approach is employed in Lohndorf and Wozabal (2015) to valuate natural gas storage and futures trading with a high dimensional price process. ADDP's ability to accommodate interstage dependency in the stochastics is leveraged as forward prices are modeled as a multivariate geometric Brownian motion (MGBM) (a first-order Markov model), approximated by a lattice in which state transition probabilities are carefully tuned through lattice quantization learning when discretized for use with ADDP. Discrete approximations of the MGBM model for futures prices are used again in both Lai et al. (2010) and Nadarajah et al. (2015) in which the management of commodity storage (such as natural gas) is considered. In Lai et al. (2010), a novel forward ADP

approach to valuating natural gas storage is used to benchmark commonly used heuristic valuation methods. In Nadarajah et al. (2015), VFAs for states in a commodity storage MDP are found using relaxations of approximate linear programs. While these gas storage problems are related to our battery storage problem, they are concerned with much longer time steps and optimization horizons (months to years) and experience different sources of uncertainty.

We now narrow our focus to the operation of a single energy storage device at finer timer scales. Löhndorf and Minner (2010) combines approximate policy iteration with least squares policy evaluation (a form of forward ADP) to optimize the day-ahead bidding of a renewable supply-energy storage system participating in both the day-ahead and real-time electricity market. In the paper, the price and supply are both modeled as first order autoregressive processes. In this case, the renewable supply is assumed to be large enough to affect the spot price. To account for this in the model, the mean and variance of the price process at each time step is dependent on the current mean and variance of the renewable supply process. Zhou et al. (2013) utilizes exact backward dynamic programming to find an optimal policy for controlling a co-located wind power-storage device system in the presence of stochastic wind and electricity price processes. Wind speed (later transformed to wind power) is modeled as an AR(1) process with a seasonal component, while a carefully calibrated model with mean-reverting, seasonal, and jump components is used for prices. Results show that the exact dynamic programming approach results in considerable improvements over a rolling horizon procedure and other heuristic policies. In Cheng et al. (2017), a battery is co-optimized on different time scales for both energy arbitrage and frequency regulation using backward ADP. The highly correlated frequency regulation and LMP signal are modeled by forming ordered pairs of the prices and using a first order Markov chain to make hourly transitions between these ordered pairs. The LMP then evolves on a five minute time scale conditioned on its basepoint at the beginning of the hour.

Finally, in the same wind farm-battery storage system configuration considered in this paper, Jiang et al. (2014) compares the effectiveness of several forward ADP methods in optimizing the system. The stochastics (wind energy and electricity price) are modeled in various ways to create different versions of the problem for the same system (load is assumed to be a deterministic sinusoidal function). An IID random error term with different distributions (uniform, pseudo-normal) and variances determines the change in wind power at each time step. Meanwhile, price is modeled in three ways: a sinusoid with a random IID error term, a Markov chain with an IID error term, and a Markov chain with jumps having two error terms at each time step: one with low variance and one with high variance that occurs with low probability. Extensive testing of forward ADP methods based on approximate value iteration and approximate policy iteration produced mixed results unless structure such as convexity or monotonicity could be used.

In contrast to dynamic programming approaches, policies from other classes, such as parametric policies fitted using policy search, can incorporate more complex stochastic models as these are only used for forward simulation. However, in dynamic programming we see it is quite common to utilize a simplified model of the stochastic processes involved in an effort to reduce the dimensionality of the state variable. An extreme case of this is assuming the stochastics exhibit intertemporal, or stage-wise, independence. We see this modeling assumption, for example, in the classic SDDP formulation (Pereira and Pinto 1991).

In other cases, the wind power forecast error, wind speed, or wind power itself, is modeled as a first-order Markov process, whether as an autoregressive process (e.g. Löhndorf and Minner 2010, Zhou et al. 2013) or a first-order Markov chain (e.g. Jiang et al. 2014, Cheng et al. 2017). While the true wind process may be of a higher order, or depend on additional state-of-the-world variables, the information lost as a result of the simplification is often a necessary compromise to allow for the efficient computation of a solution. However, it may be the case that the simplified model is inaccurate with respect to the stochastic base model (or the actual behavior of the process) to the point that solution quality suffers in terms of expectation, robustness, or both.

Similar simplifying assumptions are made for spot electricity prices in dynamic programming approaches. For example, in Sioshansi et al. (2014) perfect knowledge of future prices over the optimization horizon is assumed. A model with intertemporal independence is seen in Xi et al. (2014) where the hourly spot price is determined by a lognormal distribution with parameters that depend on the hour of day. The first-order Markov process assumption is commonly used for prices as well, as is done with the mean reverting models used in Tseng and Barz (2002) or Mokrian and Stephen (2006). This is also the case for the Markov model with jumps that is one of the models considered in Jiang et al. (2014) and the model with mean-reversion, jumps, and deterministic seasonality from Zhou et al. (2013).

The crossing state models from Durante et al. (2017) that are used in this paper capture key characteristics of the processes, such as crossing time and error distributions, that can affect solution quality if not properly modeled. Furthermore, this can be accomplished with a relatively low-dimensional simplification of the information state variable. The result is a unique first-order Markov process that bridges the gap between simpler models which ignore important characteristics of the stochastics and complex models that require complicated information states (e.g. neural networks, models with many explanatory variables). Despite this modeling choice, finding exact solutions for the energy storage problems considered in this paper is still intractable due to the dimensionality of the state, outcome, and decision space.

To overcome these curses of dimensionality, we can turn to one of the many strategies that fall under the umbrella of forward approximate dynamic programming, such as approximate policy

iteration or approximate value iteration. See Bertsekas (1995) or Powell (2011) for an overview of the field. However, these classical ADP methods with several general purpose machine learning methods (linear models, tree regression, support vector regression, Gaussian process regression) were found to work quite poorly in similar energy storage problems, producing policies that achieve only 70-90 percent of optimality (Jiang et al. 2014).

In contrast, backward approximate dynamic programming was found to produce very high quality results in realistic battery storage problems in Cheng et al. (2017), with results ranging from 95 to 98 percent of optimality. Outside of the Cheng et al. (2017) paper, Backward ADP has been relatively unexplored in the energy storage domain. Although, it is closely related to numerical integration methods in dynamic programming that are prevalent in fields such as economics (see Judd 1998, Rust 2008). The main difference is the method of randomly sampling states in the backward pass employed by the backward ADP algorithms, whereas numerical DPs often evaluate points in the state space on a grid before forming value function approximations. In addition, the numerical integration methods are typically applied to low-dimensional problems. Here we look to apply backward ADP to a higher dimensional problem.

Different approximation architectures can be used in backward ADP or numerical DP approaches to fit VFAs to system states. In one example, Senn et al. (2014) uses artificial neural networks to determine VFAs in a backward ADP scheme. In a different case, Cheng et al. (2017) assumes states can be indexed in a matrix at each time period. Then, after carefully and efficiently sampling states, a low rank matrix approximation is formed. Another common approximation technique is the linear VFA (that is, linear in the parameters), or piecewise linear VFA. This is seen in Cai and Judd (2010) for example.

In this paper, backward ADP techniques from Cheng et al. (2017) are adapted to a setting where the stochastic wind and price processes are modeled with HSMMs. We explore both linear and lookup table approximation architectures. However, the backward ADP approach can be used with any parametric or nonparametric statistical learning methods (see, for example, Hastie et al. 2001). The choice of approximation architecture is problem dependent, and should be chosen based on careful experimentation and benchmarking.

3. Modeling the Exogenous Processes

There are three exogenous processes that we have to model in this energy storage problem: the energy produced by wind, the electricity price, and the energy demand. Of these, the demand profile is far more predictable and exhibits far less variation from its expected value at each time t. In the interest of reducing the dimensionality of the outcome space and state space, we assume the load, L_t, follows a deterministic, time-dependent function formed from historical summertime

demand profiles in the Princeton, NJ area. We consider three different demand profiles: one from a hot day, one from an average day, and one from a cool day. These all exhibit daily patterns such as troughs in the morning and peaks in the afternoon, but the total load is larger on hotter days. The instantaneous demand varies between 2 MW to 5 MW depending on the time of day and temperature. The series $\{L_t\}_{t=0}^T$ belongs in the initial state variable S_0 as it is a latent variable in our problem.

3.1. Wind Power Model

Here we provide an overview of the univariate crossing state HSMM presented in Durante et al. (2017), including how to train the model. As mentioned in Section 1, we utilize this model for wind power production as it accurately captures the distribution of times for which wind power is above or below its forecast, otherwise known as the crossing times (examples are illustrated in Figure 1). This behavior is poorly captured by standard time series models in comparison to the crossing state model, as is shown in Figure 2. In the model, we have conditional forecast errors distributions that are dependent on the partially hidden state variable, the crossing state. The "semi-Markov" quality stems from the fact that crossing state transition probabilities are dependent on state duration.

Given a reference series, $\{f_t^E\}_{t=0}^T$, which in the case of wind power production is a power forecast, we define the error $\hat{E}_t = E_t - f_t^E$ where E_t is the instantaneous wind power at time t. We assume a fixed forecast over the optimization horizon (it is a latent variable in the problem) and, therefore, $\{f_t^E\}_{t=0}^T \in S_0$. The wind power model is trained on sets of day-ahead wind power forecasts provided by an external vendor using a proprietary forecasting method and the resulting actual power output series. These are gathered from a single large wind farm in the Great Plains region.

First we define both complete up- and down- crossing times and running up- and down- crossing times. A running up-crossing time of duration d starting at time t is given by:

$$\tau_t^{U,E} = d \text{ if } \begin{cases} \hat{E}_{t-d} \le 0 \\ \hat{E}_{t+d'} > 0 \quad \forall d' \in \{0,1,...,d-1\} \end{cases}.$$

Similarly, a running down-crossing time of duration d at time t is defined as:

$$\tau_t^{D,E} = d \text{ if } \begin{cases} \hat{E}_{t'-d} > 0 \\ \hat{E}_{t'+d'} \le 0 \quad \forall d' \in \{0,1,...,d-1\} \end{cases}.$$

Next, let the set of all indices such that forecast errors cross over from the negative to positive regime be $\mathcal{C}^{U,E} = \{t | \hat{E}_{t-1} \le 0 \wedge \hat{E}_t > 0\}$. Likewise, let the set of all indices such that errors cross over from the positive to negative regime be $\mathcal{C}^{D,E} = \{t | \hat{E}_{t-1} \ge 0 \wedge \hat{E}_t < 0\}$. Complete crossing times are simply running crossing times with $t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E}$. Examples of points in time belonging to $\mathcal{C}^{U,E}$ and $\mathcal{C}^{D,E}$, as well as complete up- and down- crossing times are shown in Figure 1.
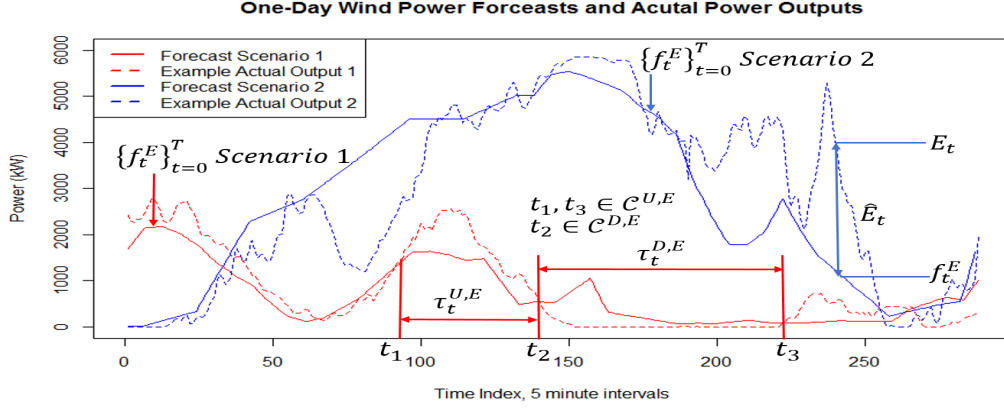
Figure 1   **Two wind forecast scenarios are shown (solid lines), along with corresponding actual wind power outputs (dotted). On the first forecast scenario, examples of points in time belonging to $\mathcal{C}^{U,E}$ and $\mathcal{C}^{D,E}$ and both a complete up- and down- crossing time are shown. On the second forecast path, a single forecast error,**
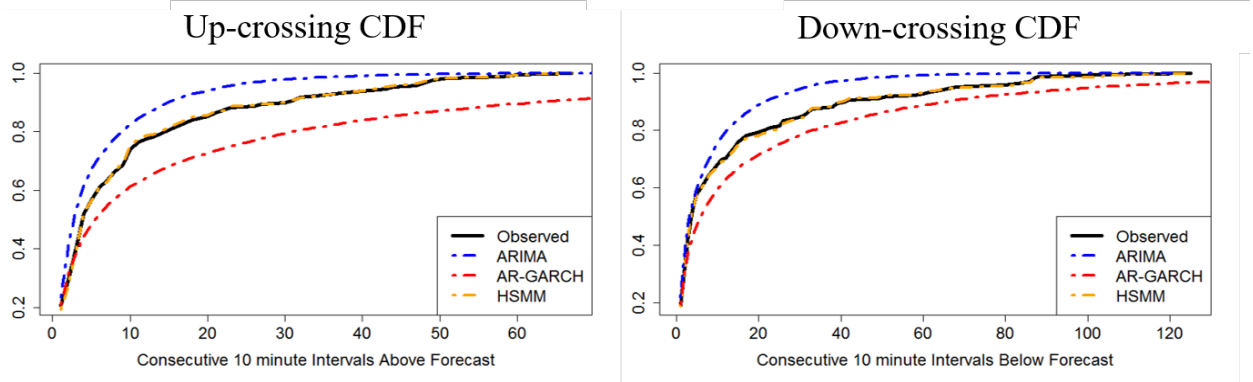$$\hat{E}_t = E_t - f_t^E \text{ is shown.}$$

Figure 2   **Observed versus simulated up- (left) and down-crossing (right) time cumulative distribution functions for wind power forecast errors. The simulated distributions come from two common time series models, ARIMA and AR-GARCH, and the univariate crossing state HSMM presented in Durante et al. (2017). The crossing state model replicates crossing time distributions almost exactly (the CDF plots overlap the observed CDFs), while the time series model produce crossing times that are either consistently too long (AR-GARCH) or too short (ARIMA). This figure was adapted from Durante et al. (2017).**

For both complete up-crossing and down-crossing times, there exists empirical distributions $F^{U,E}$ and $F^{D,E}$ respectively. Complete up-crossing time distributions are quantized by partitioning into m^E bins, splitting at the $q_i = \frac{i}{m^E}$ quantile points for $i = 0, 1, ..., m^E - 1$. A complete up-crossing time, τ_t^U, belongs to crossing time duration bin $B_t^E = b$ if $q_b \leq F^{U,E}(\tau^U) < q_{b+1}$. Complete down-crossing time distributions are similarly quantized.

Our crossing state variable $I_t^{C,E} \equiv (C_t^E, B_t^E)$ is defined as the pair of variables describing whether or not the error is above the forecast, $C_t^E = \mathbf{1}_{\{\hat{E}_t > 0\}}$, and to which crossing time duration bin, B_t^E, the completed crossing time will belong. Note that this means that during online optimization, the

state C_t^E is observable (we know if we are above or below the forecast), but the duration bin B_t^E is not until the sign variable C_t^E switches. However, when building the crossing state-dependent error distributions from training data for the model we can observe the duration bin at each point in time by peeking into the future to find the complete crossing time. Letting $\mathcal{I}^{C,E}$ be the set of all possible crossing states for the process, there exists a distribution from training data of complete crossing times $F_i^{\tau,E}$ for each possible crossing state $i \in \mathcal{I}^{C,E}$. These distributions serve as the sojourn time distributions for the crossing states.

Transitions between crossing states are made utilizing a transition matrix $\mathbb{P}(i'|i)$ for each pair of crossing states $(i',i) \in \mathcal{I}^{C,E} \times \mathcal{I}^{C,E}$ in which self-transitions are not allowed ($\mathbb{P}(i|i) = 0 \ \forall i \in \mathcal{I}^{C,E}$). This matrix is computed from training data by considering only pairs of points in time $(t, t+1)$ such that $t+1 \in (\mathcal{C}^{U,E} \cup \mathcal{C}^{D,E})$ (points where the crossing state makes a transition). For all of these pairs, letting $n(I_{t+1}^{C,E} = i'|I_t^{C,E} = i)$ be the count of the transitions from state i to state i' occurring for each pair of crossing states (i,i') and $n(I_t^{C,E} = i)$ be the number of times $I_t^{C,E} = i$ for each crossing state i, the empirical transition probability from crossing state i to i' is given by:

$$\mathbb{P}(i'|i) = \frac{n(I_{t+1}^{C,E} = i'|I_t^{C,E} = i)}{n(I_t^{C,E} = i)}. \tag{1}$$

The crossing state duration-dependent transition probability is then a function of the running crossing time as follows:

$$\mathbf{P}(I_{t+1}^{C,E} = i'|I_t^{C,E} = i, \tau_t^E) = \begin{cases} 1 - F_i^{\tau,E}(\tau_t^E) & \text{if } i' = i \\ F_i^{\tau,E}(\tau_t^E)\mathbb{P}(i'|i) & \text{for } i' \neq i \end{cases}.$$

Next we describe conditioning the error generation on the crossing state. From training data, there exists empirical conditional error CDFs $F_i^{\hat{E}}$ and corresponding error density functions $\mathbf{P}(\hat{E}_{t+1}|i)$ for $i \in \mathcal{I}^{C,E}$. Error distributions are not identical across crossing states; in fact they are likely to be quite different, such as the case where the error distribution is asymmetric. Furthermore, error distributions are likely to vary across duration bins as well. This behavior is seen in the left plot of Figure 3, which shows error densities for each type of duration bin with $m^E = 3$. Thus, to better capture the behavior of the error process, the error generation process is conditioned on the crossing state $I_t^{C,E}$.

In addition to errors being crossing state-dependent, they are dependent on error history as well. A first order Markov chain is used to model this behavior. Similar to how the crossing time distributions are quantized, each conditional error distribution $F_i^{\hat{E}}$ is partitioned into n^E bins, splitting at the $q_j = \frac{j}{n^E}$ quantile points for $b = 0, 1, ..., n^E - 1$. The error $\hat{E}_t$ belongs to bin $\hat{E}_t^b$ if $q_b \leq F_i^{\hat{E}}(\hat{E}_t) < q_{b+1}$. Then, given $\hat{E}_t \in \hat{E}_t^b$, we form conditional empirical distributions for the error at time $t+1$ giving $\mathbf{P}(\hat{E}_{t+1}|I_t^{C,E}, \hat{E}_t^b)$. The dependence of $\hat{E}_{t+1}$ on E_t^b, the aggregated state of the
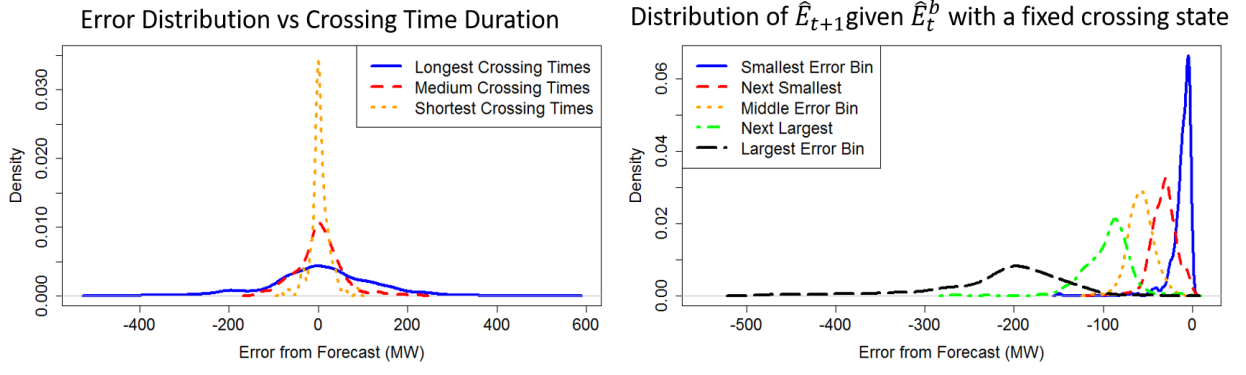
Figure 3   Left: Error distributions conditioned on B_t^E with $m^E = 3$. Both positive and negative errors for equal values of B_t^E are combined to form the distributions. The variance of the errors tends to increase with run length. Right: Example of conditional distributions for $\hat{E}_{t+1}$ given a fixed crossing state, $I_t^{C,E} = (0, 2)$, but varying which error bin, $\hat{E}_t^b$, that $\hat{E}_t$ belongs to. The magnitude of the next error is largely dependent on the magnitude of the current error. This figure is a slightly modified version of a figure from Durante et al. (2017).

current error, is illustrated in the right plot of Figure 3 in which conditional distributions for $\hat{E}_{t+1}$ are plotted for a fixed crossing state, but varying error states $\hat{E}_t^b$.

It is important to realize that the same error $\hat{E}_t$ can fall in different error bins for different crossing states. For example, the error $\hat{E}_t = +2000$ kW may be in bin $\hat{E}_t^b = 5$ for the $I_t^{C,E} = (1, 0)$ crossing state (short up-crossings), but for the $I_t^{C,E} = (1, 2)$ state (longer up-crossings), it may belong to bin $\hat{E}_t^b = 2$. To avoid additional notation, *the variable $\hat{E}_t^b$ will always be paired with a crossing state and refers to the bin that the error $\hat{E}_t$ belongs to for the corresponding crossing state.*

Finally, for each crossing state $i \in \mathcal{I}^{C,E}$, there exists an error density $\mathbf{P}(\hat{E}_{t+1}|i, t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E})$. This is the distribution of the initial error given the process has just transitioned to the new crossing state i.

The information state variable for this process, denoted I_t^E, contains the following variables at each time t: $I_t^E \equiv \left(C_t^E, B_t^E, \tau_t^E, \hat{E}_t^b\right) \equiv \left(I_t^{C,E}, \tau_t^E, \hat{E}_t^b\right)$. If known, these variables fully determine the distribution of the exogenous information $\hat{E}_{t+1}$ as follows:

$$\mathbf{P}(\hat{E}_{t+1}|I_t^{C,E} = i, \tau_t^E, \hat{E}_t^b) = (1 - F_i^{\tau,E}(\tau_t^E))\mathbf{P}(\hat{E}_{t+1}|i, \hat{E}_t^b) +$$
$$F_i^{\tau,E}(\tau_t^E) \sum_{i' \neq i} \mathbb{P}(i'|i)\mathbf{P}(\hat{E}_{t+1}|i', t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E}). \tag{2}$$

3.2.   Compact Information States

Letting $\tau_i^{max,E}$ be the largest complete crossing time for crossing state $i \in \mathcal{I}^{C,E}$, we see that there are $\sum_{i \in \mathcal{I}^{C,E}} n^E \tau_i^{max,E}$ possible information states at each time t. This number can be quite large, especially if crossing times tend to span many time periods. Fitting value functions to system states

with backward ADP will likely be too computationally expensive without a more compact information state representation. For this reason, we introduce a modified compact process information state $\tilde{I}_t^E \equiv \left(C_t^E, B_t^E, \hat{E}_t^b \right)$ which can take on $2 \times m^E \times n^E$ states.

Note that the error distributions $\mathbf{P}(\hat{E}_{t+1}|i, \hat{E}_t^b)$ and $\mathbf{P}(\hat{E}_{t+1}|i, t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E})$ for all $i \in \mathcal{I}^{C,E}$ are unaffected by this change. However, the transition between crossing states must now be modeled with a Markov approximation of the semi-Markov model as no running crossing time count is maintained. Transition probabilities are now given by a time-invariant modified crossing state transition matrix $\tilde{\mathbb{P}}(I_{t+1}^{C,E}|I_t^{C,E})$ which allows for self-transitions. This is estimated from training data using Equation 1; however all time periods t are considered, not only pairs of points where errors switch signs.

Consequently, the distribution of $\hat{E}_{t+1}$, given only the compact information state, is:

$$\mathbf{P}(\hat{E}_{t+1}|I_t^{C,E}=i, \hat{E}_t^b) = \tilde{\mathbb{P}}(i|i)\mathbf{P}(\hat{E}_{t+1}|i, \hat{E}_t^b) + \sum_{i' \neq i} \tilde{\mathbb{P}}(i'|i)\mathbf{P}(\hat{E}_{t+1}|i', t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E}).$$

We will be fitting value functions using these compact information states with backward ADP and the Markov approximation for the process, while using the full information state and the semi-Markov model in forward passes.

3.3. Electricity Price Model

The final exogenous process left to model in this problem is the Locational Marginal Price of electricity. Electricity prices exhibit much different behavior than wind power forecast errors, such as the tendency to spike under certain conditions, a heavy dependence on temperature, and a daily periodic pattern. These characteristics can be seen in Figure 4, in which a two week portion of LMP data from the Princeton, New Jersey area during the summer of 2015 is shown. Also shown is the observed temperature in the area during this period. We extend the crossing state model described
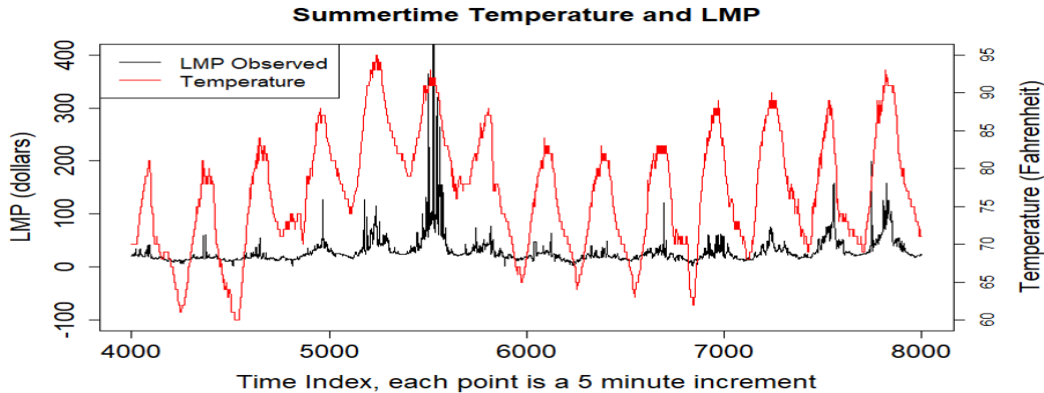

Figure 4 LMP path and temperature data for the Princeton, NJ, area during two weeks in July 2015. LMPs spike with daily peaks and exhibit larger spikes when the peaks are higher.

in Sections 3.1 and 3.2 to model summertime electricity prices by using a periodic reference series and conditioning price distributions on temperature.

As mentioned, there exists a seasonal (daily) component to the LMP path. Before fitting the model, we remove this seasonal component at each time t, $P_t^{random} = P_t - P_t^{seas}$, where P_t is the time t electricity price, so that we can focus on modeling tendencies of the data that cannot be easily captured by a simple periodic function. Using the function $g(t)$ to represent the index for the time of day based on the time increment, the seasonal component is the expected value of the price at the time of day index $g(t)$: $P_t^{seas} = \mathbb{E}[P_{g(t)}]$.

Note that we are also missing a critical element to the previous hidden semi-Markov model: the forecast for LMPs. However, a true forecast is not necessary to use the model; only a reference point at each point in time is required, forming a series of reference points. For this application, the sum of the mean of the seasonality-removed prices, $\bar{p} = \mathbb{E}[P_t^{random}]$, and the periodic series P_t^{seas} serves as the periodic reference series: $f_t^P = \bar{p} + P_t^{seas}$. As this reference series is deterministic over the optimization horizon, $\{f_t^P\}_{t=0}^{T} \in S_0$.

Observe from Figure 4 that LMPs often spike during the high points of temperature each day. Additionally, the higher the peak temperature, the higher the spikes and price level in general. To capture this behavior, we incorporate temperature as an explanatory variable in the error generation portion of the model by conditioning the price distribution at time $t + 1$ on the temperature forecast for time $t + 1$. Ideally, this would be a forecast made at time t, but we assume a fixed temperature forecast over the optimization horizon as it is accurate enough to capture the general behavior of temperature in the near future and simplifies the optimization algorithm.

We first isolate two components of the temperature series – the seasonal component and the trend component (identified by applying a length 50 moving average filter to the temperature series). Then, the model is conditioned on both of these variables. As both series take on continuous values, they must be aggregated into bins first. In this application, we use two bins for the seasonal component and three bins for the trend component. This granularity, along with the bin division points, were decided upon through trial and error.

Let h_t^s be the seasonal component of the temperature series at time t. Let $h^{s,max} = \max\limits_t h_t^s$. The explanatory variable y_t^s may take on two values:

$$y_t^s = \begin{cases} 2 \text{ if } h_t^s \geq 0.75 \times h^{s,max} \\ 1 \text{ if } h_t^s < 0.75 \times h^{s,max} \end{cases}.$$

This division is intended to separate periods when the temperature is peaking from all other times of the day.
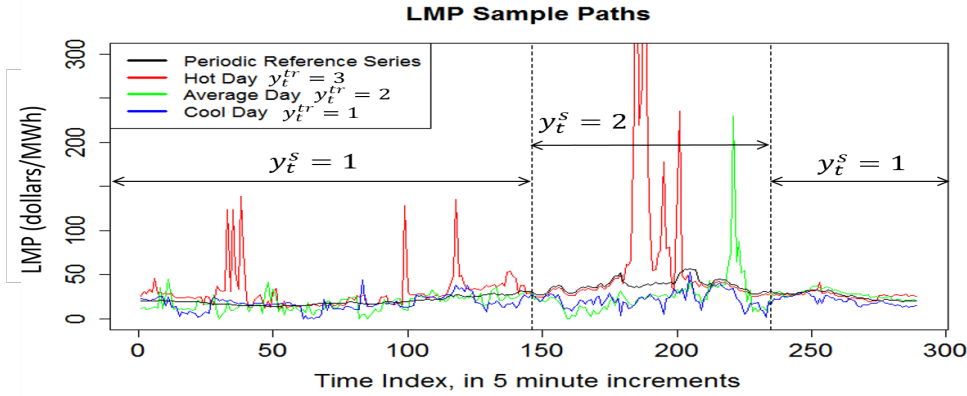
Figure 5 Example LMP sample paths over different days. The effect of conditioning on temperature trend and seasonality separately can be observed.

Let h_t^{tr} be the trend component of the temperature series at time t. Let $h^{tr,max} = \max_t h_t^{tr}$. The explanatory variable y_t^{tr} may take on three values:

$$y_t^{tr} = \begin{cases} 3 \text{ if } h_t^{tr} \geq 0.8 \times h^{tr,max} \\ 2 \text{ if } 0.3 \times h^{tr,max} \leq h_t^{tr} < 0.8 \times h^{tr,max} \\ 1 \text{ if } h_t^{tr} < 0.3 \times h^{tr,max} \end{cases}.$$

This division is intended to separate cool, average, and hot days.

Several LMP sample paths for a one day horizon are shown in Figure 5 where the effect of conditioning on the variables y_t^s and y_t^{tr} can be observed. As $\{y_t^s\}_{t=0}^T$ and $\{y_t^{tr}\}_{t=0}^T$ are both deterministic series (stemming from the fact that we assume a fixed temperature forecast over the optimization horizon), these belong in the initial state S_0.

With this additional conditioning, our error-from-reference density at time $t + 1$ gives $\mathbf{P}(\hat{P}_{t+1}|I_t^P, y_{t+1}^s, y_{t+1}^{tr})$ for each information state I_t^P. Similar to the wind model, we also have a compact information state, $\tilde{I}_t^P = (C_t^P, B_t^P, \hat{P}_t^b)$, utilized when fitting value functions. The set of all compact information states, $\tilde{\mathcal{I}}^P$, has cardinality $2 \times m^P \times n^P$ as in the wind model.

3.4. The Knowledge State and its Bayesian Update

While observing the above stochastic processes in the forward pass, the system operator will know the magnitude and sign of the current error from the reference series for each process as well as both τ_t^E and τ_t^P, the running crossing times. However, the complete crossing time duration bins B_t^E and B_t^P are unknown at time t. Consequently, $\hat{E}_t^b$ and $\hat{P}_t^b$ are unknown as well. Thus, the information states are partially unobservable.

As a VFA-based policy takes the action that maximizes the one step contribution plus the expected value of the downstream state, we must know the probability of reaching each downstream state when taking an action. This requires *knowledge states*, denoted K_t^E and K_t^P for wind and

price respectively, giving the operator's distribution of belief about the unknown variables. The remainder of this section discusses the belief state and Bayesian update for the wind energy process, but note that there are analogous time t beliefs and Bayesian update functions for the price process as well.

Let $K_t^E \equiv \left(\{ \mathbf{P}(I_t^{C,E} = i) \}_{i \in \mathcal{I}^{C,E}}, \tau_t^E, \hat{E}_t \right)$. Given K_t^E, we can derive our belief about the distribution of the error at time $t+1$, $\mathbf{P}(\hat{E}_{t+1} | K_t^E)$. We are able to determine the sign of the error, C_t^E, based on $\hat{E}_t$. Then, for each possible value of B_t^E, and corresponding crossing state $I_t^{C,E} = (C_t^E, B_t^E)$, $\hat{E}_t$ can belong to only one error bin $\hat{E}_t^b$. We then have:

$$\mathbf{P}(\hat{E}_{t+1} | K_t^E) = \sum_{i \in \mathcal{I}^{C,E}} \mathbf{P}(I_t^{C,E} = i) \mathbf{P}(\hat{E}_{t+1} | i, \tau_t^E, \hat{E}_t^b), \tag{3}$$

where $\mathbf{P}(\hat{E}_{t+1} | i, \tau_t^E, \hat{E}_t^b)$ is given by Equation 2.

Subsequently, following the observation of $\hat{E}_{t+1}$, a Bayesian update is performed on the knowledge state at each time step according to the update function $K_{t+1}^E = U^E(K_t^E, \hat{E}_{t+1})$ defined by the following two cases:

- Case 1: $sign(\hat{E}_{t+1}) = sign(\hat{E}_t)$. In this case $\tau_{t+1}^E = \tau_t^E + 1$. This is then used to compute the likelihood that the future complete crossing time belongs to bin B_t^E for each crossing state $i = (C_t^E, B_t^E)$ given the running crossing time τ_{t+1}^E. This likelihood is given by $1 - F_i^{\tau,E}(\tau_{t+1}^E)$. Furthermore, the likelihood of observing error $\hat{E}_{t+1}$ in crossing state i given a crossing state transition has not occurred is $\mathbf{P}(\hat{E}_{t+1} | i, \hat{E}_t^b)$. Thus with prior beliefs, $\mathbf{P}(I_t^{C,E} = i)$ for $i \in \mathcal{I}^{C,E}$, we compute the posterior beliefs as follows:

$$\mathbf{P}(I_{t+1}^{C,E} = i) = \frac{1}{p^{norm}} \left(\mathbf{P}(I_t^{C,E} = i)(1 - F_i^{\tau,E}(\tau_{t+1}^E)) \mathbf{P}(\hat{E}_{t+1} | i, \hat{E}_t^b) \right),$$

  where $p^{norm} = \sum_{i' \in \mathcal{I}^{C,E}} \mathbf{P}(I_t^{C,E} = i')(1 - F_{i'}^{\tau,E}(\tau_{t+1}^E)) \mathbf{P}(\hat{E}_{t+1} | i', \hat{E}_t^b)$.

- Case 2: $sign(\hat{E}_{t+1}) \neq sign(\hat{E}_t)$. In this case $\tau_{t+1}^E = 0$ and we are able to determine the crossing state at time t based on the sign of $\hat{E}_t$ and the completed crossing time τ_t^E; let this be state i^*. Additionally, we know that a crossing state transition has taken place. The likelihood of observing error $\hat{E}_{t+1}$ in crossing state i given a crossing state transition has just occurred is $\mathbf{P}(\hat{E}_{t+1} | i, t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E})$. Thus, for $i \in \mathcal{I}^{C,E}$, posterior beliefs are given by:

$$\mathbf{P}(I_{t+1}^{C,E} = i) = \frac{1}{p^{norm}} \left(\mathbb{P}(i | i^*) \mathbf{P}(\hat{E}_{t+1} | i, t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E}) \right),$$

where $\mathbb{P}(i | i^*)$ is defined in Equation (1) and $p^{norm} = \sum_{i' \in \mathcal{I}^{C,E}} \mathbb{P}(i' | i^*) \mathbf{P}(\hat{E}_{t+1} | i', t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E})$. Given these recursive updating formulas for the knowledge state, we only need to initialize our beliefs and knowledge state at $t = 0$. Given $\hat{E}_0$, we use a discrete uniform distribution for the initial beliefs: $\mathbf{P}(I_0^{C,E} = i) = 1/(m^E)$ for $i \in \mathcal{I}^{C,E}$ such that $C_t^E = sign(\hat{E}_0)$. Setting $\tau_0^E = 0$, this forms K_0^E.
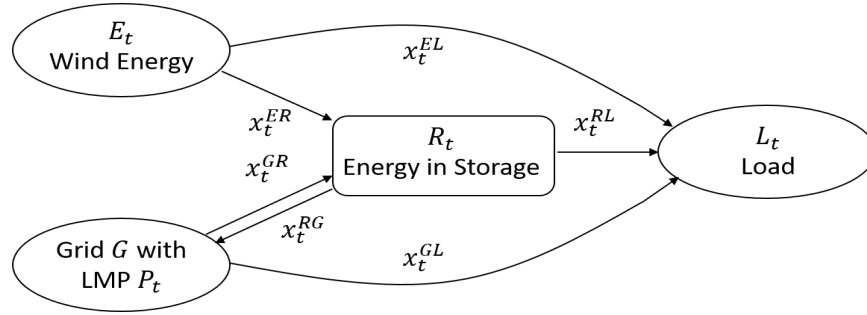
Figure 6 **An energy storage problem with four energy nodes: wind, the larger power grid, storage, and demand/load; three exogenous variables: wind energy production, the price of electricity, and the demand; and six decision variables representing possible energy allocations at each time step.**

4. Mathematical Model of the Energy Storage Problem

We consider an energy storage problem, similar to the model presented in Powell and Meisel (2016), in which a stochastic wind energy supply, an energy storage device, and the power grid, with an associated stochastic electricity price, are used in combination to satisfy a time-varying power demand. The objective is to control the system, whose configuration is illustrated in Figure 6, at minimum cost (or maximum profit). There are four nodes, six decision variables, and three exogenous processes in this energy storage system configuration.

This section provides a complete model of the stochastic optimization problem by defining the state variable, the decision variable, exogenous information, the transition function, and the objective function. The modeling style and notation are adopted from Powell (2016) and Powell and Meisel (2016).

4.1. The State Variable

First, we define the dynamic state variable S_t and the initial state variable S_0. Following the modeling convention from Powell (2016), the initial state contains all data pertinent to the problem, including constants and deterministic variables. Conversely, S_t contains only variables which may change over time.

We are treating forecasts as static in this problem, thus they belong in S_0. In addition to latent variables, S_0 contains initial beliefs about unknown parameters and initial values of wind, price, and storage level. Thus, our initial state is given by:

$$S_0 = \left(\left\{ f_t^E, f_t^P, L_t, y_t^s, y_t^{tr} \right\}_{t=0}^T, R^{max}, \eta, \rho^{ch}, \rho^{dch}, R_0, E_0, P_0, K_0^E, K_0^P \right),$$

where R^{max} is the maximum capacity of the battery, $\eta \in [0,1]$ is the battery round trip efficiency, and ρ^{ch} and ρ^{dch} are maximum battery charge and discharge rates respectively.

For our VFA-based policies, we must track our time t beliefs in the knowledge states K_t^E and K_t^P. The contribution function, $C(S_t, x_t) = P_t(L_t - x_t^{GR} - x_t^{GL} + \eta x_t^{RG})$, requires that the current

electricity price be known, thus $P_t = \hat{P}_t + f_t^P$ is incorporated as well. Finally, $E_t = \hat{E}_t + f_t^E$ and R_t are included to determine our constraints on the decision vector x_t. Thus, our dynamic state variable S_t for $t > 0$ is given by:

$$S_t = (R_t, E_t, P_t, K_t^E, K_t^P).$$

Note that in this problem, the post-decision state variable S_t^x, which carries only the information necessary to transition to S_{t+1} *after* a decision has been made (Powell 2011), is given by:

$$S_t^x = (R_t^x, K_t^E, K_t^P).$$

R_t^x is the energy level of the battery following the decision to use or store energy at time t.

4.2. The Decision Variable

The decision variable at each time t is given by:

$$x_t = (x_t^{GL}, x_t^{GR}, x_t^{RG}, x_t^{EL}, x_t^{ER}, x_t^{RL}),$$

where x_t^{AB} indicates energy sent from node A to node B. This is subject to the following constraints:

$$x_t^{EL} + x_t^{ER} \leq E_t, \tag{4}$$

$$x_t^{GL} + x_t^{EL} + \eta x_t^{RL} = L_t, \tag{5}$$

$$x_t^{RG} + x_t^{RL} \leq \min(R_t, \rho^{dch}), \tag{6}$$

$$x_t^{ER} + x_t^{GR} \leq \min(\rho^{ch}, R^{max} - R_t), \tag{7}$$

$$x_t^{GL}, x_t^{GR}, x_t^{RG}, x_t^{EL}, x_t^{ER}, x_t^{RL} \geq 0. \tag{8}$$

All vector decisions x_t that satisfy the above constraints form the set of feasible decisions at time t, $\mathcal{X}_t(S_t)$, or $\mathcal{X}_t$ where the dependence on S_t is implied. However, for the dynamic programming approaches in this paper, we assume that $x_t^{EL} = \min(E_t, L_t)$, and $x_t^{ER} = \min\left(\rho^{ch}, R^{max} - R_t, E_t - x_t^{EL}\right)$ to reduce the dimensionality of the decision space.

Constraint (4) ensures that the amount of renewable energy used at time t does not exceed the amount produced. Constraint (5) requires that the load be met at each time t. Limits on the amount of energy that can be drawn from and sent to the battery at each time step are imposed in constraints (6) and (7) respectively. Finally, a non-negativity constraint is imposed on each element of the decision vector in constraint (8). Note that there is no constraint on the amount of energy that can be purchased from the grid as we assume it is an infinite source of power.

4.3. Exogenous Information

Section 3 was devoted to the stochastic modeling of the exogenous wind and electricity price processes. The exogenous information arriving between t and $t+1$ is given by $W_{t+1} = (\hat{E}_{t+1}, \hat{P}_{t+1})$ where our time t belief about the distribution of $\hat{E}_{t+1}$ is given by $\mathbf{P}(\hat{E}_{t+1}|K_t^E)$ and that of $\hat{P}_{t+1}$ is given by $\mathbf{P}(\hat{P}_{t+1}|K_t^P)$.

4.4. The Transition Function

The following equations describe the system transition function, $S_{t+1} = S^M(S_t, x_t, W_{t+1})$:

$$R_{t+1} = R_t + \eta(x_t^{GR} + x_t^{ER}) - x_t^{RL} - x_t^{RG}, \tag{9}$$

$$E_{t+1} = f_{t+1}^E + \hat{E}_{t+1}, \tag{10}$$

$$P_{t+1} = f_{t+1}^P + \hat{P}_{t+1}, \tag{11}$$

$$K_{t+1}^E = U^E(K_t^E, \hat{E}_{t+1}), \tag{12}$$

$$K_{t+1}^P = U^P(K_t^P, \hat{P}_{t+1}). \tag{13}$$

Note this can be broken into two functions: the pre- to post- decision state transition function, $S_t^x = S^{M,x}(S_t, x_t)$, and the transition function from post-decision state to the next pre-decision state given the arrival of exogenous information W_{t+1}, $S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$. $S_t^x = S^{M,x}(S_t, x_t)$ alters the resource energy level based on the decision x_t as in Equation (9): $R_t^x = R_t + \eta(x_t^{GR} + x_t^{ER}) - x_t^{RL} - x_t^{RG}$. Additionally, P_t and E_t are dropped from pre- to post- decision state, and the remaining variables remain unaltered. $S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$ is then given by Equations (10)-(13) along with $R_{t+1} = R_t^x$.

4.5. The Objective Function

As we aim to operate the system at maximum profit in the real-time electricity market, our finite horizon control problem has the objective function,

$$\max_{\pi \in \Pi} \mathbb{E}^\pi \left[\sum_{t=0}^T C(S_t, X_t^\pi(S_t)) | S_0 \right],$$

where the contribution function is given by $C(S_t, x_t) = P_t(L_t - x_t^{GR} - x_t^{GL} + \eta x_t^{RG})$ and $S_{t+1} = S^M(S_t, X_t^\pi(S_t), W_{t+1})$ is determined by the system transition function. We are maximizing over the set of all possible policies $\pi \in \Pi$. In the contribution function we profit from satisfying the load or selling energy back to the grid, but must pay for any energy that originates from the grid. We assume that we are a price taker and any decisions made do not affect the electricity price process.

5. Backward Approximate Dynamic Programming

Assuming a terminal reward of $V_T^*(S_T)$ for each terminal state S_T, if computationally tractable, an optimal policy can be found using vanilla backward dynamic programming to find value functions, $V_t^*(S_t)$, for each possible system state. Value functions are given by Bellman's equation for finite horizon problems:

$$V_t^*(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}\left[V_{t+1}^*(S_{t+1}) | S_t, x_t \right] \right), \tag{14}$$

and, once these are found, the optimal policy,

$$X_t^*(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}\left[V_{t+1}^*(S_{t+1}) | S_t, x_t \right] \right), \tag{15}$$

maximizes the one-step contribution plus the expected value of the downstream state. In cases where performing a complete backward pass is either impossible or impractical (as is the case in our problem), we can instead rely on approximations of these value functions and use a VFA-based policy:

$$X_t^\pi(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}\left[\bar{V}_{t+1}(S_{t+1}) | S_t, x_t \right] \right), \tag{16}$$

where $\bar{V}_{t+1}(S_{t+1})$ is some approximation of the value of the downstream states.

To remove the expectation from the policy, we can fit value functions instead to the post-decision state variable S_t^x. This is possible for our problem as the transition function $S^M(S_t, x_t, W_{t+1})$ can be broken into two parts: $S_t^x = S^{M,x}(S_t, x_t)$ and $S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$. The resulting post-decision state-based VFA policy is given by:

$$X_t^\pi(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right), \tag{17}$$

where $\bar{V}_t^x(S_t^x)$ serves as an approximation of $\mathbb{E}\left[V_{t+1}^*(S_{t+1}) | S_t, x_t \right]$.

As mentioned previously, forward ADP is a far more common approach to fitting value function approximations for either $\bar{V}_{t+1}(S_{t+1})$ or $\bar{V}_t^x(S_t^x)$, but the classical use of machine learning methods to approximate value functions with forward ADP has been found to work quite poorly in similar energy storage problems (Jiang et al. 2014). This paper instead utilizes a novel ADP technique, backward ADP, as the resulting VFA-based policy achieves better performance.

Additional notation is necessary to describe the backward ADP algorithms presented in this paper. Let $\mathcal{S}_t$ and $\mathcal{S}_t^x$ be the set of all time t pre- and post-decision states respectively, and $\mathcal{S}_t^\alpha$ be a random sample of states in $\mathcal{S}_t$ sampled at rate $\alpha \in (0, 1]$. Also, let $\mathbf{P}(S_{t+1} | S_t^x)$ be the transition probability for each pair (S_t^x, S_{t+1}) with $S_t^x \in \mathcal{S}_t^x$ and $S_{t+1} \in \mathcal{S}_{t+1}$. Finally, let $\mathcal{S}_{t+1}(S_t^x)$ be the set of all time $t+1$ pre-decision states such that $\mathbf{P}(S_t | S_t^x) > 0$ (all the pre-decision states that can be reached from post-decision state S_t^x).

5.1. Backward ADP with Lookup Tables

Backward ADP algorithms resemble textbook backward dynamic programming, except that instead of looping over all states, only a sampled set of states are evaluated, and the results are used to create a value function approximation, replacing the exact lookup table value function used in classical discrete Markov decision processes. This reduces both the CPU time and memory necessary to compute and store value functions.

Let $\mathcal{S}_{t+1}^{\alpha}$ be a small subset of pre-decision states which were sampled at rate α. Their value, $\bar{V}_{t+1}(S_{t+1})$ for $S_{t+1} \in \mathcal{S}_{t+1}^{\alpha}$, is computed as usual by maximizing the one-step contribution plus the value of the resulting post-decision state over feasible decisions at each time step. Sampling the pre-decision states expedites the maximization step.

The expectation step is also streamlined by computing post-decision state values $\bar{V}_t^x(S_t^x)$ as a weighted average (with weights proportional to transition probabilities) of the values from the sampled set of pre-decision states. The approximations $\bar{V}_t^x(S_t^x)$ are stored in a lookup table. This method, described in Algorithm 1, requires careful sampling of the time $t+1$ pre-decision states such that each time t post-decision state can reach at least one sampled next pre-decision state to form the approximation $\bar{V}_t^x(S_t^x)$ for each S_t^x. This is accomplished by looping over each time t post-decision state $S_t^x \in \mathcal{S}_t^x$ and sampling at minimum one time $t+1$ pre-decision state S_{t+1} from $\mathcal{S}_{t+1}(S_t^x)$.

Algorithm 1 Backward ADP with Post-Decision State Values Stored in Lookup Table Form

Choose sample rate α.
Initialize terminal contributions $\bar{V}_T(S_T)$ for each $S_t \in \mathcal{S}_t$.
$\mathcal{S}_T^{\alpha} = \{\}$
for each $S_{T-1}^x \in \mathcal{S}_{T-1}^x$ **do**
    Add $\lceil \alpha | \mathcal{S}_T(S_{T-1}^x) | \rceil$ states $S_T \in \mathcal{S}_T(S_{T-1}^x)$ to $\mathcal{S}_T^{\alpha}$
end for
Initialize $t \leftarrow T-1$.
while $t \geq 0$ **do**
    Expectation Step:
    for each $S_t^x \in \mathcal{S}_t^x$ **do**
        $\mathcal{S}_{t+1}^{sampled} = \mathcal{S}_{t+1}(S_t^x) \cap \mathcal{S}_{t+1}^{\alpha}$
        $p^{norm} = \sum\limits_{S_{t+1} \in \mathcal{S}_{t+1}^{sampled}} \mathbf{P}(S_{t+1}|S_t^x)$
        $\bar{V}_t^x(S_t^x) = \frac{1}{p^{norm}} \sum\limits_{S_{t+1} \in \mathcal{S}_{t+1}^{sampled}} \mathbf{P}(S_{t+1}|S_t^x)\bar{V}_{t+1}(S_{t+1})$
        Write $\bar{V}_t^x(S_t^x)$ to memory.
    end for
    Maximization Step:
    if $t > 0$ **then**
        $\mathcal{S}_t^{\alpha} = \{\}$
        for each $S_{t-1}^x \in \mathcal{S}_{t-1}^x$ **do**
            Add $\lceil \alpha | \mathcal{S}_t(S_{t-1}^x) | \rceil$ states $S_t \in \mathcal{S}_t(S_{t-1}^x)$ to $\mathcal{S}_t^{\alpha}$
        end for
        for $S_t \in \mathcal{S}_t^{\alpha}$ **do**
            $\bar{V}_t(S_t) = \max\limits_{x_t \in \mathcal{X}_t} \{C(S_t, x_t) + \bar{V}_t^x(S_t^x)\}$ where $S_t^x = S^{M,x}(S_t, x_t)$.
        end for
    end if
    $t \leftarrow t-1$
end while

The approach of storing post-decision state values in lookup table form is effective for problems in which the post-decision state space is much more compact than the pre-decision state space, as is the case in our energy storage problem. Of course, the accuracy of the approximate value functions depends on the sampling rate α.

5.2. Backward ADP using Parametric VFAs for Pre-Decision States

An alternative approach, laid out in Algorithm 2, is to again sample pre-decision states, but instead form a parametric approximation $\bar{V}_{t+1}(S_{t+1}|\theta_{t+1})$ given the set of sampled states and their resulting values following maximization over feasible actions. If each $S_{t+1} \in \mathcal{S}_{t+1}^{\alpha}$ has associated value $v(S_{t+1})$, we search for:

$$\theta_{t+1}^{*} = \underset{\theta_{t+1} \in \Theta_{t+1}}{\arg\min} \sum_{S_{t+1} \in \mathcal{S}_{t+1}^{\alpha}} w(S_{t+1}) L\left(v(S_{t+1}), \bar{V}_{t+1}(S_{t+1}|\theta_{t+1})\right),$$

where $w(S_{t+1})$ are weights satisfying $\sum_{S_{t+1} \in \mathcal{S}_{t+1}^{\alpha}} w(S_{t+1}) = 1$, where $L(v(s), \bar{V}_{t+1}(s|\theta_{t+1}))$ is an appropriate loss function such as $L(v, \bar{v}) = (v - \bar{v})^2$ ($\mathcal{L}_2$-norm) or $L(v, \bar{v}) = |v - \bar{v}|$ ($\mathcal{L}_1$-norm), and Θ_{t+1} is the set of possible parameter vectors for the chosen function form $\bar{V}_{t+1}(S_{t+1}|\theta_{t+1})$. The approximation given by $\bar{V}_{t+1}(S_{t+1}|\theta_{t+1}^{*})$ is then used in the expectation step to approximate the values of all time $t+1$ pre-decision states. Additionally, the low-dimensional, best-fit parameter vector θ_{t+1}^{*} is stored in memory at each time step for later use when simulating the policy forward. Note that this alternative is much more memory efficient than a lookup table VFA.

A commonly used architecture for VFAs is a linear model (that is, linear in the parameters), $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f \in \mathcal{F}} \theta_{t,f} \phi_f(S_t)$ where ϕ_f for $f \in \mathcal{F}$ are basis functions of the states that must be

---

Algorithm 2 Backward ADP with Parametric VFAs

Choose sample rate α and VFA form $\bar{V}_t(S_t|\theta_t)$.
Initialize terminal contributions $\bar{V}_T(S_T)$ for each $S_T \in \mathcal{S}_T$.
$\mathcal{S}_T^{\alpha}$ is a random size $\lceil \alpha|\mathcal{S}_T| \rceil$ sample of states $S_T \in \mathcal{S}_T$
Initialize $t \leftarrow T - 1$.
while $t \geq 0$ **do**
    Solve $\theta_{t+1}^{*} = \underset{\theta_{t+1} \in \Theta_{t+1}}{\arg\min} \sum_{S_{t+1} \in \mathcal{S}_{t+1}^{\alpha}} w(S_{t+1}) L\left(v(S_{t+1}), \bar{V}_{t+1}(S_{t+1}|\theta_{t+1})\right)$
    Write θ_{t+1}^{*} to memory.
    Expectation Step:
    for each $S_t^x \in \mathcal{S}_t^x$ **do**
        $\bar{V}_t^x(S_t^x) = \sum_{S_{t+1} \in \mathcal{S}_{t+1}} \mathbf{P}(S_{t+1}|S_t^x) \bar{V}_{t+1}(S_{t+1}|\theta_{t+1}^{*}).$
    end for
    Maximization Step:
    if $t > 0$ **then**
        $\mathcal{S}_t^{\alpha}$ is a random size $\lceil \alpha|\mathcal{S}_t| \rceil$ sample of states $S_t \in \mathcal{S}_t$
        for $S_t \in \mathcal{S}_t^{\alpha}$ **do**
            $v(S_t) = \underset{x_t \in \mathcal{X}_t}{\max} \left\{ C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right\}$ where $S_t^x = S^{M,x}(S_t, x_t).$
        end for
    end if
    $t \leftarrow t - 1$
end while

chosen beforehand and should be appropriate for the application (picking these is not a trivial task). If basis functions are chosen poorly, solution quality will suffer. This paper utilizes basis functions of the state that are relevant to the energy storage problem. These are described in greater detail in Section 6.

Note that, despite only utilizing a linear VFA in this paper, the structure of Algorithm 2 can be generalized to work for many parametric or even nonparametric approximations for $\bar{V}_t(S_t)$. This requires using the appropriate method for fitting the function given the pairs of sampled pre-decision states and their values and storing any parameters or data necessary for simulating forward.

5.3. Backward ADP for the Energy Storage Problem

By using the hidden semi-Markov crossing state model presented in Durante et al. (2017), we create a more realistic energy storage problem, but in the process form a partially observable MDP as the crossing states are partially hidden. Recall that when simulating forward, we have knowledge states K_t^E and K_t^P, giving the operator's distribution of belief about the crossing states given the history of the process up to time t. This information is not available in the backward pass. Instead, value functions are fit to each possible resource state conditioned on each information state in the backward pass. The value of being at resource level R_t at time t given the wind and price processes are in information states I_t^E and I_t^P is given by:

$$\bar{V}_t(R_t, I_t^E, I_t^P) = \underset{x_t \in \mathcal{X}_t}{\arg\max} \left(C(R_t, I_t^E, I_t^P, x_t) + \bar{V}_t^x(R_t^x, I_t^E, I_t^P) \right),$$

where $\bar{V}_t^x(R_t^x, I_t^E, I_t^P) = \mathbb{E}\left[\bar{V}_{t+1}(R_{t+1}, I_{t+1}^E, I_{t+1}^P) | R_t^x, I_t^E, I_t^P \right]$. However, as mentioned in Section 3.2, finding values for all possible system states using the full information state is very computationally expensive.

These computational issues are addressed by utilizing the compact information states introduced in Section 3.2 in combination with backward ADP. We need to introduce additional notation to handle this. Let $\tilde{S}_t = (R_t, E_t, P_t, \tilde{I}_t^E, \tilde{I}_t^P)$ and $\tilde{S}_t^x = (R_t^x, \tilde{I}_t^E, \tilde{I}_t^P)$ be compact time t pre- and post-decision states respectively using the compact information states for the wind and price processes $\tilde{I}_t^E$ and $\tilde{I}_t^P$ from Section 3.2. The sets of all possible compact time t pre- and post-decision states are then represented by $\tilde{\mathcal{S}}_t$ and $\tilde{\mathcal{S}}_t^x$. Using these compact state variables, the post-decision to next pre-decision state transition matrix is defined as follows:

$$\mathbf{P}(\tilde{S}_{t+1}|\tilde{S}_t^x) = \mathbf{1}_{\left\{R_{t+1}=R_t^x\right\}} \mathbf{P}(\tilde{I}_{t+1}^E, E_{t+1}|\tilde{I}_t^E) \mathbf{P}(\tilde{I}_{t+1}^P, P_{t+1}|\tilde{I}_t^P),$$

where, noting $\hat{E}_{t+1} = E_{t+1} - f_{t+1}^E$,

$$\mathbf{P}(\tilde{I}_{t+1}^E = i', E_{t+1} | \tilde{I}_t^E = i) = \mathbf{1}_{\left\{\hat{E}_{t+1} \in \hat{E}_{t+1}^b\right\}} \tilde{\mathbb{P}}(i'|i) \times \begin{cases} \mathbf{P}(\hat{E}_{t+1}|i, \hat{E}_t^b) & \text{if } i' = i \\ \mathbf{P}(\hat{E}_{t+1}|i', t+1 \in \mathcal{C}^{U,E} \cup \mathcal{C}^{D,E}) & \text{otherwise} \end{cases},$$

with matrix $\tilde{\mathbb{P}}(i'|i)$ defined in Section 3.2. $\mathbf{P}(\tilde{I}_{t+1}^P, P_{t+1} | \tilde{I}_t^P)$ is defined similarly.

Replacing $S_t, S_t^x, \mathcal{S}_t$ and $\mathcal{S}_t^x$ with $\tilde{S}_t, \tilde{S}_t^x, \tilde{\mathcal{S}}_t$, and $\tilde{\mathcal{S}}_t^x$ in Algorithms 1 and 2, we find VFAs for the simplified MDP. Despite using these simplified state variables in the backward pass, we utilize the full state variables to make decisions moving forward in time according to the policy given by Equation (16). If Algorithm 1 was used, the expectation $\mathbb{E}[\bar{V}_{t+1}(S_{t+1})|S_t, x_t]$ in this policy is computed based on approximate values for simplified post-decision states: $\bar{V}_t^x(\tilde{S}_t^x)$. Note that $\bar{V}_t^x(\tilde{S}_t^x) = \bar{V}_t^x(R_t^x, \tilde{I}_t^E, \tilde{I}_t^P) = \bar{V}_t^x(R_t^x, (I_t^{C,E}, \hat{E}_t^b), (I_t^{C,P}, \hat{P}_t^b))$ is an approximation of the expected value of the downstream pre-decision state conditioned on the crossing state variables $I_t^{C,E}$ and $I_t^{C,P}$. By applying the law of total expectation using our time t belief states, we have:

$$\mathbb{E}[\bar{V}_{t+1}(S_{t+1})|S_t, x_t] = \sum_{i \in \mathcal{I}^{C,E}} \mathbf{P}(I_t^{C,E} = i) \sum_{j \in \mathcal{I}^{C,P}} \mathbf{P}(I_t^{C,P} = j) \left(\bar{V}_t^x(R_t^x, (i, \hat{E}_t^b), (j, \hat{P}_t^b)) \mathbf{1}_{\left\{R_t^x = S^{M,x}(R_t, x_t)\right\}} \right),$$

where $\mathbf{P}(I_t^{C,E} = i)$ and $\mathbf{P}(I_t^{C,P} = j)$ (contained in K_t^E and K_t^P) give the probability that the stochastic wind and price processes are in each crossing state at time t based on their history. If, instead, parametric VFAs were calculated and stored using Algorithm 2, the policy can be computed by simply replacing $\bar{V}_t^x(R_t^x, (I_t^{C,E}, \hat{E}_t^b), (I_t^{C,P}, \hat{P}_t^b))$ in the above equation with $\mathbb{E}[\bar{V}_{t+1}(S_{t+1}|\theta_{t+1}^*)|R_t^x, (I_t^{C,E}, \hat{E}_t^b), (I_t^{C,P}, \hat{P}_t^b)]$.

6. Numerical Results

In this section we observe the trade-off between computation time and performance for the backward ADP methods at different sampling rates. We show that, in comparison to an exact solution to the simplified MDP, we can reduce computation time significantly via backward ADP without much degradation in solution quality. Additionally, we show that backward ADP outperforms a common forward ADP method, approximate policy iteration, and a parametric policy function approximation in this problem setting.

The above claims are substantiated by the results from several test cases, for which input parameters are varied. We test different combinations of battery sizes, battery charge rates (where we assume $\rho = \rho^{ch} = \rho^{dch}$), energy forecasts, temperature forecasts, and load profiles. Table 1 describes each test case. Also displayed is the C-rate of the battery, where C= $(\rho \times$ number of time periods per hour$)/R^{max}$, which is the maximum discharge rate relative to battery capacity. Common C-rates range from 0.1C (slow charging) to 2C and higher (fast).

Across the test cases, we set $m^E = 3$ in the wind model which determines the number of crossing time duration bins (B_t^E). Additionally, we let $n^E = 3$, fixing the number of error bins $(\hat{E}_t^b)$ for

Table 1 Test cases for the energy storage problem. The temperature forecast (Hot, Avg, Cool) determines the load profile and the series $\{y_t^{tr}\}_{t=0}^T$. The numbers under $\{f_t^E\}_{t=0}^T$ refer to the wind forecasts shown in Figure 1.

Case	1	2	3	4	5	6	7	8	9	10	11	12
Temperature	Avg	Cool	Hot	Avg	Cool	Cool	Hot	Hot	Avg	Avg	Hot	Hot
$\{f_t^E\}_{t=0}^T$	1	1	1	1	2	2	1	1	2	2	2	2
R^{max} (MWh)	5.00	5.00	5.00	5.00	4.00	4.00	3.33	3.33	2.00	4.00	5.00	2.50
ρ (kWh)	83.3	83.3	83.3	416.7	333.3	83.3	416.7	83.3	41.7	83.3	83.3	83.3
C-rate	0.20C	0.20C	0.20C	1.00C	1.00C	0.25C	1.50C	0.30C	0.25C	0.25C	0.20C	0.40C

each crossing state. Wind power (E_t), which can take on continuous values, is discretized to form a discrete MDP. Using a uniform interval of 100 kW, we allow 51 possible values for E_t, evenly spaced from 0 MW to $E^{max} = 5$ MW. Similarly, we let $m^P = 1$ and $n^P = 4$ in the price model and discretize P_t evenly using an interval of \$2/MWh from P^{min} to P^{max} where these represent the lowest and highest prices observed in training data. Here, $P^{min} = -\$312$/MWh and $P^{max} = \$780$/MWh, resulting in 547 possible values P_t may take on. The state of charge in the battery, R_t, is also discretized evenly from 0 MWh to R^{max}. The number of charge states will, in general, depend on the charge rate, R^{max}, and the program time step, though the time step is fixed at five minutes in these problems. The number of battery charge states is between 30 and 60 in the test cases (slower charging, larger batteries need more states). Additionally, η is set to 1.0, and we assume the battery starts out with a 50 percent charge. Finally, each test case has an optimization horizon of $T = 288$ as we make an energy allocation decision every five minutes over the course of one day.

6.1. Policies Tested

Backward ADP methods are used to find value functions in each test case with various sampling rates α. Type Lookup-α is the approach described in Algorithm 1 which uses a lookup table representation of the value function. Type Lin-α represents the method in Algorithm 2 using linear parametric VFAs and basis functions given by:

$$\phi_1(S_t) = (C_t^E - 0.5)(B_t^E + 1), \ \phi_2(S_t) = E_t, \ \phi_3(S_t) = E_t^2, \ \phi_4(S_t) = (C_t^P - 0.5)(B_t^P + 1), \ \phi_5(S_t) = P_t,$$

$$\phi_6(S_t) = P_t^2, \ \phi_7(S_t) = R_t, \ \phi_8(S_t) = R_t^2, \ \phi_9(S_t) = E_t P_t, \ \phi_{10}(S_t) = E_t R_t, \ \text{and} \ \phi_{11}(S_t) = P_t R_t,$$

where $\phi_1(S_t)$ and $\phi_4(S_t)$ are chosen such that the longer down-crossing states evaluate to lower numbers and the longer up-crossing states evaluate to higher numbers.

The backward ADP policies are also compared to several other policies. One is the optimal policy using the value functions resulting from exact backward dynamic programming. Another is approximate policy iteration (API) utilizing linear regression to fit value functions to post-decision states. This is a form of forward ADP (for a more thorough discussion of API, see Jiang et al. (2014)). The performance of this policy will improve over time as the number of training

iterations increases. We restrict training time to 12 hours, often much longer than the backward ADP algorithms take to run in the test problems, and use the resulting policy at the end of this training period. It is also worth noting that, as with backward ADP with linear VFAs, policy performance is highly dependent on the selection of proper basis functions here as well. The last policy tested is a carefully tuned buy-low, sell-high policy function approximation (PFA) defined by $\theta = (\theta^H, \theta^L)$ with $\theta^H > \theta^L$:

$$X_t^{PFA}(S_t|\theta) = \begin{cases} x_t^{EL} = \min\left\{L_t, E_t\right\} \\ x_t^{RL} = \begin{cases} \min\left\{L_t - x_t^{EL}, \min\left\{R_t, \rho^{dch}\right\}\right\} & \text{if } P_t > \theta^H \\ 0 & \text{if } P_t \leq \theta^H \end{cases} \\ x_t^{GL} = L_t - x_t^{EL} - x_t^{RL} \\ x_t^{ER} = \min\left\{E_t - x_t^{EL}, \rho^{ch}, R^{max} - R_t\right\} \\ x_t^{GR} = \begin{cases} \min\left\{\rho^{ch} - x_t^{ER}, R^{max} - R_t - x_t^{ER}\right\} & \text{if } P_t < \theta^L \\ 0 & \text{if } P_t \geq \theta^L \end{cases} \\ x_t^{RG} = \begin{cases} \min\left\{R_t - x_t^{RL}, \rho^{dch} - x^{RL}\right\} & \text{if } P_t > \theta^H \\ 0 & \text{if } P_t \leq \theta^H \end{cases} \end{cases},$$

where θ is manually tuned via grid search. The θ resulting in the highest average objective function value in each case is chosen.

6.2. Policy Performance Results

The policies described in the previous section are simulated using a set of 100 realistic sample paths of wind energy forecast errors and LMPs, similar to those shown in Figures 1 and 5. Table 2 reports each policy's performance as the cumulative contribution over 100 trials as a percent of the cumulative contribution earned using the exact solution to the discretized MDP.

From Table 2, we see that both backward ADP methods consistently outperform both approximate policy iteration and the policy function approximation. Also observe that the lookup table method performs better than the linear VFA at both sampling rates in most cases. This may be because the linear structure cannot capture the complex relationships between states and their values that exist in this problem, even with a large set of sampled states. The linear VFA method is likely to perform better in a problem where the value functions exhibit more structure, such as a known polynomial form. However, note that the performance of the linear VFA is approximately the same for both the larger and smaller sampling rate α, whereas this is not the case for the lookup table method as performance degrades with decreasing α. This is an encouraging sign for the use of Algorithm 2 and linear VFAs in MDPs with much larger pre-decision state spaces where Algorithm 1 may not work well as it may be necessary to sample at an even smaller rate α.

Table 2 also reports the average CPU time in hours required to compute value functions for each dynamic programming algorithm (in the API case it is the maximum allotted training time).

Table 2 **Mean performance of policies in the various test cases over 100 trials, reported as a percent of the optimal policy. Shown on the far right is the average time in hours needed to compute value functions for each algorithm (or time allotted for policy search), but note that this is highly problem-dependent and can vary greatly between test cases. The optimal policy took an average of 11.3 hours to compute.**

Case	1	2	3	4	5	6	7	8	9	10	11	12	Avg	CPU (hrs)
Lookup-.01	99.1	96.7	98.0	88.8	98.0	100.0	93.2	98.4	99.6	99.4	98.8	98.9	97.4	0.41
Lin-.01	96.2	96.4	96.9	91.8	88.2	95.2	95.4	98.9	98.9	98.2	98.3	99.5	96.2	1.62
Lookup-.10	100.1	99.5	99.3	97.1	99.7	100.2	97.2	99.4	100.0	100.1	99.6	99.8	99.3	0.67
Lin-.10	96.3	96.5	98.1	91.1	88.3	95.2	94.9	98.9	98.9	98.2	99.0	99.2	96.2	2.72
API	82.7	77.9	79.5	57.3	76.2	90.8	50.5	80.4	94.7	90.0	86.4	86.9	79.5	12.0
PFA	93.6	92.3	93.4	71.4	80.6	91.3	72.3	93.4	97.3	95.2	96.0	94.6	89.3	5.14

In general, sampling in the backward ADP algorithms will result in much quicker CPU times (as expected) without sacrificing much performance in comparison to the full MDP solution. This is useful as the backward pass can be performed closer to the time at which it is used to control the system. The accuracy of the input load, wind, and temperature forecasts will be improved as a result of the shorter forecast lead times.

Finally, we note that the storage problems considered in this paper required between .075 GB and .15 GB of memory to store value functions in lookup table form for post-decision states. This is a significant improvement over storing pre-decision state values in lookup table form, however this number can also grow quickly as the dimensionality of the post-decision state increases. Thus, if memory storage becomes an issue, the linear VFA form may be preferable as it only requires that a low- dimensional parameter vector be stored at each time step, requiring something on the order of 10 KB to 100 KB of storage space depending on the dimension of the parameter vector (regardless of the dimensionality of the state space).

7. The Value of Hidden Semi-Markov Models

Finally, in this section we observe that the choice of stochastic model has an effect on the quality of the solution. Specifically, we present results supporting the claim that explicitly modeling crossing time behavior leads to the development of more robust policies for storage systems. We show this by training VFAs on two different models for wind power forecast errors that take into account intertemporal correlation, one that replicates crossing times well and one that does not, and then evaluating the resulting policies on sample paths drawn from history. The models used to train VFAs are:

1. A first-order autoregressive (AR) model: $\hat{E}_t = \gamma^E \hat{E}_{t-1} + \mathcal{N}(0, (s^{E,resid})^2)$ where γ^E is the lag-1 coefficient and $s^{E,resid}$ is the standard deviation of the residuals when fitting this model to training data. This model does not replicate crossing times well (see Figure 2).

2. The crossing state HSMM described in this paper, which replicates crossing times accurately.

The AR model is restricted to a first order model (an AR(1) model) to keep the state variable compact enough to perform backward dynamic programming. This is a fairly common assumption seen in the literature when dynamic programming approaches are considered. For this comparison, the price process is modeled with the crossing state model described in Section 3.3 in both cases.

We train VFAs using the two wind models in three cases from Table 1. Additionally, for each test case we train the wind power model on forecast error data sets from two different Great Plains wind farms from different months of the year (see Section 3.1 for a description of the data). Value functions are then fit to post-decision states in lookup table form using either exact or approximate backward dynamic programming for each model. For the HSMM, we use Algorithm 1 with a sampling rate of $\alpha = 0.10$. The lookup table VFAs for the AR(1) error model are found using exact backward dynamic programming. Thus, the resulting policy is an optimal policy under this modeling assumption. Note that the full solution for the HSMM could be computed in about the same amount of time as the full solution using the AR model. However, the purpose of this test is to show that the benefits of using a better stochastic model can, depending on the application, outweigh the negative effects of approximations in the optimization algorithm that may be necessary to efficiently compute a solution. Thus, the approximate solution is used.

In each case, the policies are then evaluated on two sets of 25 wind power and LMP sample paths: one *Typical* set and one *Worst Case* set. The *Typical* sample paths are a set of 25 historical wind power forecast error and electricity price sample paths chosen at random. Let this set be Ω^{Typ} and each individual sample path be $\omega^{Typ} \in \Omega^{Typ}$. Testing is then repeated on the set of 25 *Worst Case* wind and electricity price sample paths. These are created to produce conditions that could lead to especially high operating costs. The same 25 wind power forecast error sample paths from the *Typical* cases are used, but wind power is altered to drop to 0 kW from 2:00 P.M. to 8:00 P.M., just before and during most of peak usage hours. Thus, the tendency of wind energy to unexpectedly drop out and underperform its forecast for an extended period occurs at the worst possible time. In addition, the LMP path in each case is chosen from one of the five days with the highest average LMP observed in training data. Let this set be Ω^{WC} and each individual sample path be $\omega^{WC} \in \Omega^{WC}$. Let $F^{\pi,Typ}$ and $F^{\pi,WC}$ be the mean additional profit earned by following policy π outside of the baseline profit earned from satisfying the demand (as any feasible policy would receive this portion of the profit) in both *Typical* and *Worst Case* scenarios. Letting $\tilde{C}(S_t, x_t) = P_t(\eta x_t^{RG} - x_t^{GR} - x_t^{GL})$ be the contribution function with the baseline profit subtracted, $F^{\pi,Typ}$ is calculated as follows:

$$F^{\pi,Typ} = \frac{1}{|\Omega^{Typ}|} \sum_{\omega^{Typ} \in \Omega^{Typ}} \sum_{t=0}^{T} \tilde{C}\left(S_t(\omega^{Typ}), X_t^\pi(\omega^{Typ})\right), \tag{18}$$

Table 3   In several test cases, and for two different wind farms, value functions are trained assuming either an AR(1) model for wind power forecast errors or the crossing state HSMM. For each test case, $F^{\pi,Typ}$ and $F^{\pi,WC}$, the mean baseline-shifted profits in both Typical and Worst Case scenarios (calculated using Equation (18)), are reported in dollars when the resulting policies are evaluated on each set of sample paths. Bold indicates better performance in each case.

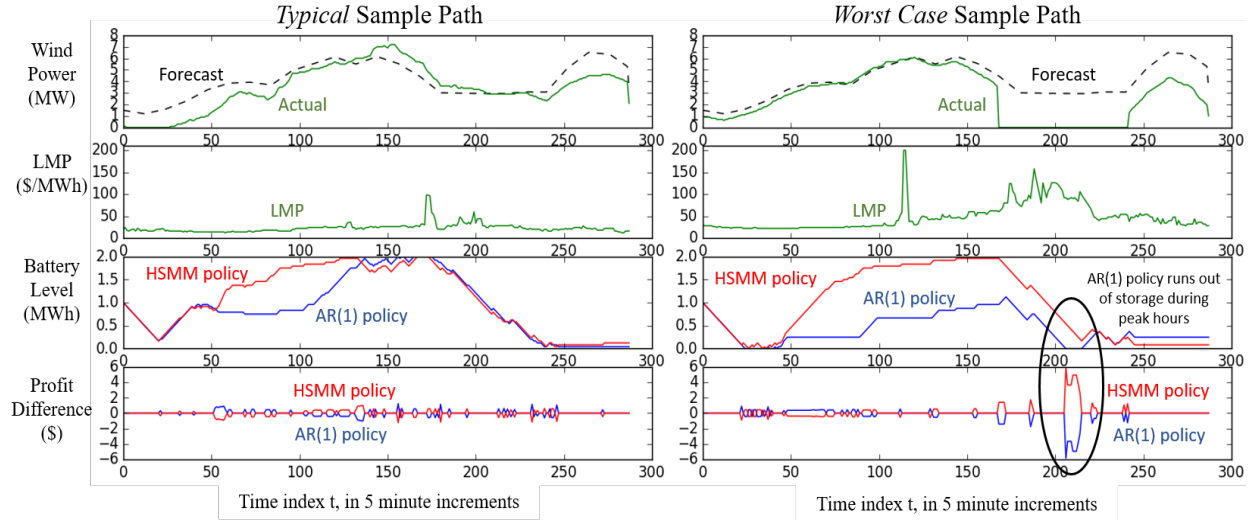| Wind Power Model	Test Case 1				Test Case 5				Test Case 9			
	Farm 1		Farm 2		Farm 1		Farm 2		Farm 1		Farm 2	
	Typ	WC	Typ	WC	Typ	WC	Typ	WC	Typ	WC	Typ	WC
AR(1)	-960	-3036	-399	-2326	55	-1756	**581**	-447	-425	-2807	**-228**	-2184
HSMM	**-959**	**-2987**	**-378**	**-2104**	**79**	**-1516**	342	**-151**	**-420**	**-2750**	-242	**-2059**

Figure 7   For one Typical (left) and one Worst Case (right) sample path in test case 9, the following plots are shown, from top to bottom: 1) the forecasted and actual wind power output, where wind power falls to 0 MW during peak hours in the Worst Case path, 2) the LMP path, which is higher and exhibits more spikes in the Worst Case path, 3) the battery level at each point in time following both policies, where the HSMM-trained policy plans extra storage early in the day and does not run out in the Worst Case scenario as the AR(1)-trained policy does (see the circled region), and 4) the deviation from the mean profit at time t earned by both policies, where profit differences tend to even out in the Typical path, but the robust HSMM-based policy avoids high costs during peak hours in the Worst Case sample path.

where $S_{t+1}(\omega^{Typ}) = S^M\left(S_t(\omega^{Typ}), X_t^\pi(S_t(\omega^{Typ})), W_{t+1}(\omega^{Typ})\right)$ (see Section 4.4 for the definition of the transition function). $F^{\pi,WC}$ is also calculated using Equation (18), but with all "Typ" superscripts replaced with "WC." Table 3 reports $F^{\pi,Typ}$ and $F^{\pi,WC}$ in each test case for both the AR(1)-trained and HSMM-trained policies.

Observe from Table 3 that while the AR(1)-trained policy may perform similarly (or perhaps better) in *Typical* scenarios, the crossing state HSMM-trained policy performs better across the test cases in *Worst Case* scenarios. Also note that the profit difference is quite significant in *Worst Case* scenarios. This indicates that training on the HSMM results in more robust policies. Insight as to why the HSMM produces a robust policy can be garnered from Figure 7, in which the AR(1)-trained and HSMM-trained policies are simulated for a single *Typical* and *Worst Case* sample

path. For both cases, each policy's deviation from the average profit at time t, along with the corresponding battery level are plotted. The AR(1) model tends to underestimate the amount of time wind stays below its forecast (see Figure 2) and thus does not emphasize storage for *Worst Case* scenarios. Thus, we see the AR(1)-trained policy tending to sell more energy early in the day, expecting enough wind to be available in later periods. The HSMM-based policy, which replenishes storage levels to a greater extent in preparation for the possibility of extended periods with little to no renewable power, will perform worse early in the day, but profit much more during the peak hours when a *Worst Case* scenario occurs. This is evident in the circled region of Figure 7 which highlights a period when the AR(1)-trained policy runs out of battery storage during peak hours in the *Worst Case* scenario and suffers high operating costs while the HSMM-based policy does not. Meanwhile, in the *Typical* scenario, cumulative profits for both policies tend to even out by the end of the optimization horizon.

Similar testing is performed for the price process model. Three models are used to train value functions, of which the first two are commonly used to model electricity prices in the literature:

1. A first-order Markov chain: $\mathbf{P}(\hat{P}_t | \hat{P}_{t-1}^b)$ where the price errors are binned into discrete states to form conditional distributions of $\hat{P}_t$ given the bin of $\hat{P}_{t-1}$.

2. A model with mean reversion and jump-diffusion (MRJD): $\hat{P}_t = \gamma^P \hat{P}_{t-1} + \mathcal{N}(0, (s^P)^2) + \mathbf{1}_{\{U < p\}} \mathcal{N}(0, (s^J)^2)$, where $U \sim Unif[0,1]$, s^J and p are chosen to capture the standard deviation and frequency of the price spike events, and s^P is the standard deviation of the data without the price spikes.

3. The crossing state model for electricity prices described in this paper.

Note that we are modeling $\hat{P}_t$, the deviation from the time-dependent mean (see Section 3.3), not P_t itself, such that each model has the same deterministic seasonality component. Additionally, the post-decision state space for each model is discretized such that each model has the same number of possible post-decision states. Finally, the same crossing state model is used for the wind power forecast error process in all cases.

The test cases in Table 4 are chosen such that the price models are tested on cool, average, and hot days. Historical price sample paths observed under each temperature condition are used for policy evaluation in the corresponding test case. These are paired with the *Typical* and *Worst Case*

Table 4   The effect of electricity price model choice on policy performance. Similar testing is performed as in Table 3, except a slightly different set of Typical and Worst Case sample paths, described in the text, are used for policy evaluation.

Electricity Price Model	Test Case 1 (Avg Temp)		Test Case 2 (Cool Temp)		Test Case 3 (Hot Temp)	
	Typical	Worst Case	Typical	Worst Case	Typical	Worst Case
Markov	**-757**	-939	-313	-448	-2833	-3457
MRJD	-849	-1029	-365	-500	-2772	-3410
HSMM	**-757**	**-937**	**-306**	**-442**	**-2687**	**-3326**

wind power sample paths described previously to form Ω^{Typ} and Ω^{WC} here. Table 4 reports $F^{\pi,Typ}$ and $F^{\pi,WC}$ for the VFA-based policies trained on each price model in each test case. Observe that the crossing state model for prices produces policies that perform similarly to policies based on more common stochastic models on average and cool temperature days, but are more robust when price spikes are larger and more frequent on hot days (whether wind has *Typical* or *Worst Case* behavior). Conditioning price distributions on a temperature forecast is clearly a key factor in this, as the model more accurately captures when price spikes occur, at what frequency they occur, and at what magnitude under several temperature conditions. The resulting policy based on this model can then plan an appropriate amount of storage based on this information.

8. Conclusions

In the search for robust control policies, whether for portfolio management or energy system control, attention has been mostly given to the use of risk measures. We show that this can also be accomplished in some cases by using a better stochastic model. This is something that has been largely ignored in the literature. Careful stochastic modeling in an effort to reduce risk should be explored when approaching a problem, perhaps in combination with risk measures if desired.

The crossing state models are designed to accurately model behaviors of the stochastics, such as extended down-crossings, that may occur with low-probability, but will lead to worse lower tail performance if not accounted for. This is seen when using a dynamic programming approach to control an energy storage system. Policies developed assuming a standard time series model for wind power (that does a poor job replicating crossing times) suffer in worst case wind power production scenarios, while the policy developed under the crossing state HSMM assumption performs better in these cases. The use of these models is not restricted to dynamic programming however. Not explored in this paper are the possible benefits of utilizing the crossing state models for policy search. The parameters of a CFA or PFA can be tuned based on sample paths generated from the crossing state models to produce a robust policy.

Focusing on VFA-based policies, finding value functions using exact backward dynamic programming with HSMMs for the stochastics is computationally expensive. Backward approximate dynamic programming is used to reduce the CPU time required to compute VFAs in the backward pass. If the MDP has a relatively compact post-decision information state, as this problem does, utilizing Algorithm 1 and lookup table VFAs proves to be not only more effective, but faster as well. However, Algorithm 2 and linear VFAs show promise for the extension of backward ADP to higher dimensional problems as performance does not degrade as quickly as the state sampling rate decreases.

Acknowledgment

References

Arnold M, Andersson G (2011) Model predictive control of energy storage including uncertain forecasts. *Power Systems Computation Conference (PSCC), Stockholm, Sweden*, volume 23, 24–29.

Bertsekas DP (1995) *Dynamic programming and optimal control*, volume 1 (Athena Scientific Belmont, MA).

Cai Y, Judd KL (2010) Stable and efficient computational methods for dynamic programming. *Journal of the European Economic Association* 8(2-3):626–634.

Camacho EF, Alba CB (2013) *Model predictive control* (Springer Science & Business Media).

Cheng B, Asamov T, Powell WB (2017) Low-rank value function approximation for co-optimization of battery storage. *IEEE Transactions on Smart Grid* .

Denholm P, Sioshansi R (2009) The value of compressed air energy storage with wind in transmission-constrained electric power systems. *Energy Policy* 37(8):3149–3158.

Durante J, Patel R, Powell WB (2017) Scenario generation methods that replicate crossing times in spatially distributed stochastic systems. *Submitted to SIAM Journal on Uncertainty Quantification* .

Garcia-Gonzalez J, de la Muela RMR, Santos LM, Gonzalez AM (2008) Stochastic joint optimization of wind generation and pumped-storage units in an electricity market. *IEEE Transactions on Power Systems* 23(2):460–468, ISSN 0885-8950, URL http://dx.doi.org/10.1109/TPWRS.2008.919430.

Han J, et al. (2016) Deep learning approximation for stochastic control problems. *arXiv preprint arXiv:1611.07422* .

Hastie T, Tibshirani R, Friedman J (2001) *The Elements of Statistical Learning*. Springer Series in Statistics (New York, NY, USA: Springer New York Inc.).

Infanger G, Morton DP (1996) Cut sharing for multistage stochastic linear programs with interstage dependency. *Mathematical Programming* 75(2):241–256.

Jiang DR, Pham TV, Powell WB, Salas DF, Scott WR (2014) A comparison of approximate dynamic programming techniques on benchmark energy storage problems: Does anything work? *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium on*, 1–8 (IEEE).

Judd KL (1998) *Numerical methods in economics* (Cambridge, Mass, MIT press).

Lai G, Margot F, Secomandi N (2010) An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation. *Operations research* 58(3):564–582.

Löhndorf N, Minner S (2010) Optimal day-ahead trading and storage of renewable energiesan approximate dynamic programming approach. *Energy Systems* 1(1):61–77.

Lohndorf N, Wozabal D (2015) Optimal gas storage valuation and futures trading under a high-dimensional price process. Technical report, Technical report.

Löhndorf N, Wozabal D, Minner S (2013) Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research* 61(4):810–823.

Ma Y, Kelman A, Daly A, Borrelli F (2012) Predictive control for energy efficient buildings with thermal storage: Modeling, stimulation, and experiments. *IEEE Control Systems* 32(1):44–64.

Mokrian P, Stephen M (2006) A stochastic programming framework for the valuation of electricity storage. *26th USAEE/IAEE North American Conference*, 24–27.

Nadarajah S, Margot F, Secomandi N (2015) Relaxations of approximate linear programs for the real option management of commodity storage. *Management Science* 61(12):3054–3076.

Pereira MV, Pinto LM (1991) Multi-stage stochastic optimization applied to energy planning. *Mathematical programming* 52(1-3):359–375.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (John Wiley and Sons, Inc.), ISBN 9781118029176, URL http://dx.doi.org/10.1002/9781118029176.ch1.

Powell WB (2016) A unified framework for optimization under uncertainty. *Optimization Challenges in Complex, Networked and Risky Systems*, 45–83 (INFORMS).

Powell WB, Meisel S (2016) Tutorial on stochastic optimization in energy part two: An energy storage illustration. *IEEE Transactions on Power Systems* 31(2):1468–1475.

Rust J (2008) *Dynamic programming* (London, UK, Palgrave Macmillan, Ltd).

Schneider F, Thonemann UW, Klabjan D (2015) Optimization of battery charging and purchasing at electric vehicle battery swap stations. *submitted to Transp. Sci* .

Senn M, Link N, Pollak J, Lee JH (2014) Reducing the computational effort of optimal process controllers for continuous state spaces by using incremental learning and post-decision state formulations. *Journal of Process Control* 24(3):133–143.

Simao HP, Powell W, Archer C, Kempton W (2017) The challenge of integrating offshore wind power in the us electric grid. part ii: Simulation of electricity market operations. *Submitted to Renewable Energy* 103:418–431.

Sioshansi R, Madaeni SH, Denholm P (2014) A dynamic programming approach to estimate the capacity value of energy storage. *IEEE Transactions on Power Systems* 29(1):395–403.

Taylor JA, Callaway DS, Poolla K (2013) Competitive energy storage in the presence of renewables. *IEEE Transactions on Power Systems* 28(2):985–996, ISSN 0885-8950, URL http://dx.doi.org/10.1109/TPWRS.2012.2210573.

Thalassinakis EJ, Dialynas EN (2004) A monte-carlo simulation method for setting the underfrequency load shedding relays and selecting the spinning reserve policy in autonomous power systems. *IEEE Transactions on Power Systems* 19(4):2044–2052.

Tseng CL, Barz G (2002) Short-term generation asset valuation: a real options approach. *Operations Research* 50(2):297–310.

Wang Q, Watson JP, Guan Y (2013) Two-stage robust optimization for nk contingency-constrained unit commitment. *IEEE Transactions on Power Systems* 28(3):2366–2375.

Warrington J, Goulart PJ, Mariéthoz S, Morari M (2012) Robust reserve operation in power systems using affine policies. *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 1111–1117 (IEEE).

Xi X, Sioshansi R, Marano V (2014) A stochastic dynamic programming model for co-optimization of distributed energy storage. *Energy Systems* 5(3):475–505.

Zhou Y, Scheller-Wolf A, Secomandi N, Smith S (2013) Managing wind-based electricity generation in the presence of storage and transmission capacity .