Evaluation of Control Frameworks for Battery Energy Storage Systems

Daniel Olshansky

Supervisor: Peter Taylor

The University of Melbourne

School of Mathematics and Statistics

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science in Mathematics

October 2023

Abstract

The increasing affordability of Battery Energy Storage Systems (BESS), coupled with substantial rises in electricity costs, has facilitated their widespread adoption among a growing number of energy consumers.

Effective BESS utilisation revolves around the integration of photovoltaic (PV) generation to meet energy demands while storing surplus energy in batteries. This stored energy can be strategically discharged during peak electricity cost periods, resulting in substantial reductions in energy expenditures. However, the linchpin for the success of these systems lies in the deployment of sophisticated control mechanisms. These mechanisms are responsible for making optimal decisions regarding battery charging and discharging, ultimately minimising operational costs.

In this thesis, we conduct an in-depth exploration of three key BESS control mechanisms: Stochastic Model Predictive Control, Deterministic Model Predictive Control, and Stochastic Dynamic Control. While these frameworks have been comprehensively discussed in BESS control literature, current literature often lacks a numerical analysis that mirrors real-world performance. Additionally, it tends to employ performance metrics that can be challenging for readers to evaluate.

We compare these control mechanisms using models that emulate real-world performance and introduce a new metric for objective, instance independent, performance evaluation.

Through our analysis, we observe that certain promising Stochastic Dynamic Control mechanisms appear to underperform when compared to Model Predictive Control-based approaches, offering limited additional benefits, thus underscoring the need for further research into the realm of BESS control.

Contents

1 Motivation and Background

1.1 Motivation

With continuous advancements and cost reductions in Battery Energy Storage Systems (BESS), coupled with soaring electricity prices, installing these systems has become increasingly financially feasible for energy consumers. BESS is often used in reference to a system that also contains photovoltaic (PV) panels, alternatively known as solar panels. A wide range of consumers stands to benefit from such systems, but in this thesis we focus on medium to large commercial energy consumers. These consumers, such as hospitals, office buildings, supermarkets, and agricultural enterprises, experience higher energy costs, thus providing a stronger incentive for optimising their BESS utilisation. Moreover, while most residential energy consumers pay fixed energy rates that vary by time, our focus centres on consumers subject to volatile Wholesale Spot Market tariffs. These tariffs fluctuate based on real-time supply and demand dynamics within the energy market, as elaborated in Section 1.2.1.

The fundamental concept behind BESS utilisation involves harnessing PV generation to meet the consumer's energy needs. Excess generation is stored in the battery, which can be discharged during peak electricity cost periods to offset the consumer's electricity demand. BESS has the potential to significantly reduce a consumer's energy expenses. However, the effective operation of these systems hinges on the implementation of a control mechanism that determines optimal times for battery charging and discharging to minimise operational costs, as we observe in Figure 1.

Model Predictive Control (MPC) systems are commonly employed in the management of BESS. MPC is an optimal control technique for systems subject to constraints over a receding horizon, as discussed in Section 4. While deterministic MPC for such systems can be relatively straightforwardly implemented using linear programming (LP), as demonstrated by Torres D., *et al.* [31], achieving true optimality necessitates accounting for the uncertainties inherent in these systems [39]. These uncertainties stem from a consumer's future energy loads, as well as their PV generation and wholesale electricity prices. This inherent

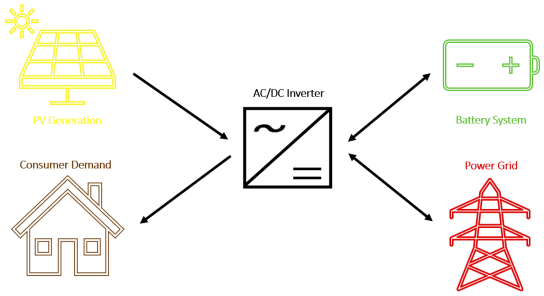


Figure 1: BESS Diagram

uncertainty leads to discrepancies between the forecasts used to control these systems and the actual outcomes. These discrepancies are explored in more detail in Section 3.

Stochastic MPC for BESS has been proposed [27], requiring an analysis of forecasting errors (residuals) that captures the relationships between these errors, especially their correlations (or joint distributions). However, a substantial portion of the scientific literature, including works such as [2], [7], and [14], often overlooks this consideration. Moreover, existing methods for preserving this information frequently limit themselves to residuals following a Gaussian distribution, as seen in [20].

While numerous papers have been published on the topic of stochastic dynamic optimal control for BESS, they often suffer from limitations in the way their results are evaluated. These limitations are frequently due to a lack of comparison between the authors' algorithms and existing state-of-the-art approaches. These issues create a cloud of uncertainty regarding the performance of different algorithms, which can even lead to sub-optimal decision-making, as demonstrated in [35].

The primary objective of this thesis is to provide a comprehensive and objective empirical and numerical analysis of the methods and techniques utilised in existing scientific literature. Additionally, we aim to establish evaluation techniques that can be used to fairly assess the performance of an algorithm.

1.2 Background

1.2.1 Electricity Pricing

In the wholesale electricity market of Australia's eastern and southern states, known as the national electricity market (NEM), the main trading platform is the Spot Market. This market is overseen by the Australian Energy Market Operator (AEMO) and involves participants such as electricity generators and consumers like retailers and large enterprises.

During each 5-minute interval, participants submit offers detailing the amount of electricity they are willing to sell or buy, along with the corresponding price, to the AEMO. These offers are organised into what is called a bid stack. The bid stack is essentially a list that arranges these offers based on ascending prices. Each entry in the bid stack represents a "tranche" and includes the volume of electricity (measured in megawatt-hours, MWh) a participant is willing to trade at a specific price (measured in dollars per megawatt-hour, $/MWh).
AEMO uses this bid stack information to construct a supply curve, which represents the aggregated supply of electricity from all the generators participating in the market. At the same time, AEMO estimates the demand for electricity during that 5-minute period. The spot price for electricity is determined at the point where the supply curve intersects with the estimated demand curve, as we see in Figure 2. This spot price acts as a reference point for all transactions occurring within that interval. Buyers who offered to pay a price equal

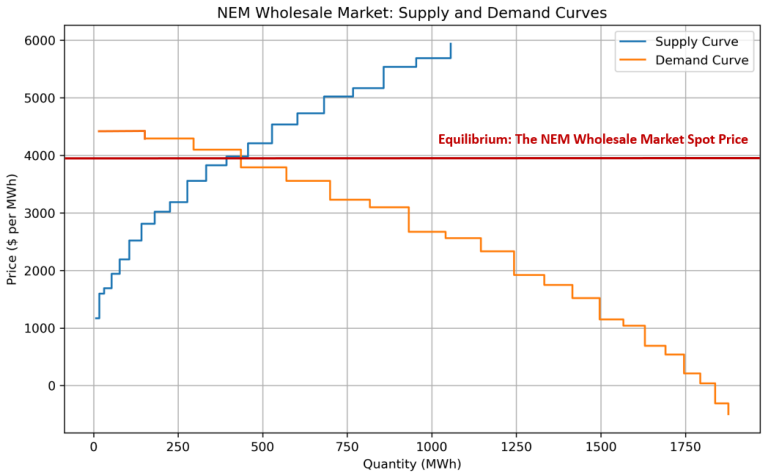


Figure 2: A diagram demonstrating how the NEM spot price is set by comparing the cumulative bids for the demand and supply of electricity.

to or greater than the spot price are successful in their bids and purchase electricity at the

determined spot price. Similarly, generators that offered to sell at a price equal to or lower than the spot price are also successful in their bids and supply electricity at the same spot price, as we show below in Figure 3. The spot price is not fixed; it varies based on the bal-

Figure 3: A diagram demonstrating how an individual buyer's bid stack works.

ance between supply and demand in the market. It can fluctuate between a market ceiling of $15,000/MWh, or 1,500c/kWh (indicating high demand or limited supply) and a market floor of -$1,000/MWh, or -100c/kWh (suggesting excess supply). While this process might seem complex, there's an option for regular consumers to participate indirectly through retailers who offer wholesale rates plus a small fee. This simplifies the process, reducing the bid stack to a single tranche and making it more accessible to a wider range of participants. For the purposes of this thesis, our focus is primarily on average electricity prices over aggregated 30-minute periods for consumers that participate in the market indirectly, which simplifies the forecasting and optimisation.

Moreover, beyond the actual cost of electricity, there are additional charges, such as network charges, that contribute to the overall electricity costs paid by residential and commercial consumers. These charges, composed of demand charges and energy consumption charges, are usually paid to transmission companies responsible for ensuring a reliable electricity network. It's important to note that these charges are fixed rate. While there are other charges in an electricity bill, the aforementioned aspects cover the essential points that we need in

order to grasp the content of this thesis.

2 Deterministic Problem Formulation

In our thesis, we delve into a complex problem, yet at its core lies a simple idea. The problem unfolds over H discrete intervals, with the current interval at $t = 0$. Decisions made at $t = 0$ translate into immediate actions, while the subsequent intervals serve as a framework to account for future costs influenced by the initial decisions. In our problem we set $H = 48$, and we sometimes use the two interchangeably.

For our inputs, we define $T = \{0, 1, \ldots, H - 1\}$ as the discrete set of non-overlapping time intervals, typically 30 minutes each, over which we optimise the schedule. The dynamic inputs, applicable for $t \in T$, encompass electricity spot prices, denoted as p_t (ranging from -100 to 1500, in c/kWh), network charges, labelled as n_t (taking a few predetermined positive values), energy demands, represented by d_t (with $d_t \geq 0$), and solar generation, characterised by g_t (with $g_t \geq 0$). Critically, the electricity spot prices (p_t), energy demands (d_t), and solar generation (g_t) are random processes, unknown ahead of time. For convenience, we define $p_t^c = p_t + n_t$ as the energy price that combines network charges and wholesale prices.

The static inputs include the battery's capacity, denoted as c (in kWh), its power rating (maximum charge/discharge per hour), labelled as $P > 0$ (in kW), and the initial state of charge, designated as B_{-1}. The battery's charging and discharging efficiency, expressed as $e \in (0, 1)$ means that to charge the battery xkWh, $\frac{x}{e}$kWh are needed, and if xkWh of energy is discharged from the battery, then only $x \cdot e$kWh will be available for usage (usually around $e \approx 0.9$).

Similarly, variables are defined for $t \in T$, with all of their quantities in kWh. These include the total energy imported in each interval, represented by $I_t \geq 0$. The total energy charged and discharged in each interval is denoted as C_t and D_t (with $C_t \geq 0$ and $D_t \geq 0$), respectively, while the battery's state of charge is described by B_t (with $B_t > 0$).

Usually, $H = 48$, making our optimisation interval run over the course of one day. We do this because batteries usually go through a least one entire charging and discharging cycle in a day, which means that we don't need to consider a battery saving its charge for an event beyond the next 24 hours. Thus, we can iteratively run solve our problem using a moving 24 hour horizon.

Given the battery's capacity of ckWh, we have a constraint given by $B_t \leq c, \forall t \in T$. Similarly, given our maximum charge and discharge rate of PkW/h, we have $C_t \leq P$ and $D_t \leq P \; \forall t \in T$.

The objective of the program is to minimise total costs, represented as the import in each interval, multiplied by the corresponding electricity cost, given by

$$\min \sum_{t \in T} [I_t \cdot p_t^c]. \tag{1}$$

This, together with the rest of the constraints that we outline below, forms the foundation for the deterministic our problem.

The constraint ensuring demand satisfaction in each interval is expressed as

$$d_t - g_t = I_t - \frac{C_t}{e} + e \cdot D_t, \tag{2}$$

while the constraint governing the battery's state of charge update is

$$B_t = B_{t-1} + C_t - D_t. \tag{3}$$

Clearly, we can see that all the decision variables can be reduced to one decision, how much to charge (or discharge) the battery, given by a_t, with it never being optimal to charge and discharge simultaneously, given the efficiency reduction factor, e.

We can formally prove this. Suppose at some interval $t \in T$, we have $\min(C_t, D_t) = \alpha > 0$. Let $C_t^{'} = C_t - \alpha$, and $D_t^{'} = D_t - \alpha$. Then, given that $C_t - D_t = C_t^{'} - D_t^{'}$, the battery's state of charge, given by Equation 3, will remain unchanged. However, $I_t - \frac{C_t^{'}}{e} + e \cdot D_t^{'} = I_t - \frac{C_t}{e} + e \cdot D_t + \alpha(\frac{1}{e} - e)$. And, given that $\frac{1}{e} - e > 0$, by replacing C_t and D_t with $C_t^{'}$ and $D_t^{'}$, respectively, we can set $I_t^{'} = I_t - \min(\alpha(\frac{1}{e} - e), I_t)$. Clearly, $I_t^{'}, C_t^{'}$, and $D_t^{'}$ still satisfy Equation 2. However, as $I_t \cdot p_t^c > I_t^{'} \cdot p_t^c$, utilising our new variable values improves our objective, given by Equation 1. Hence, given that our objective in Equation 1 is to minimise our costs, we must always have $\min(C_t, D_t) = 0$, meaning that we never charge and discharge simultaneously.

Using a_t (representing our 'action'), we can replace our constraints by $B_t = B_{t-1} + a_t$ and $d_t - g_t = I_t - \frac{[a_t]^+}{e} + e \cdot [a_t]^-$.

Hence, given that our both constraints and objective function are linear, then our entire problem is linear. We know that linear problems can be solved very efficiently with the help of linear programming.

Linear programming (LP), as detailed in [25], constitutes a fundamental optimisation problem wherein we seek a maximisation or minimisation of a linear objective function while adhering to a set of linear constraints.
The primary objective in an LP problem is to

$$\min \boldsymbol{c}^T \boldsymbol{x}, \tag{4}$$

subject to

$$A\boldsymbol{x} \geq \boldsymbol{b},$$

where

$$\boldsymbol{c} \in \mathbb{R}^n, \boldsymbol{b} \in \mathbb{R}^m, A \in \mathbb{R}^{n \times m}, \boldsymbol{x} \in (\mathbb{R}^+)^n,$$

with the aim of discovering a feasible and optimal solution for $\boldsymbol{x}$.

The linearity inherent in the problem means that every feasible linear program that is bounded below has an optimal solution. Numerous methods are available for solving such problems, with the most renowned being the Simplex algorithm, devised by Dantzig in 1947. The Simplex algorithm efficiently addresses the majority of LP problems. There are also a number of commercial LP solvers that we can use, these include a solver by Gurobi, which is available through a Python package, called gurobipy.

By applying this concept to our problem we can optimally solve our problem. The objective of our problem is given by

$$\min \sum_{t \in T} \boldsymbol{c}_t^T \cdot \boldsymbol{x}_t, \tag{5}$$

where

$$\boldsymbol{c}_t^T = [p_t^c, 0, 0];$$

$$\boldsymbol{x}_t = [I_t, C_t, D_t]^T.$$

Then, the constraints are given by

$$A_i \boldsymbol{x} \geq \boldsymbol{b}_i, \ \forall\, i = 1, \ldots, 3,$$

and $\boldsymbol{x} \geq \boldsymbol{0}$.

$$\sum_{k=1}^{t} ([A_i]_k \boldsymbol{x}_k) \geq [\boldsymbol{b}_i]_t \ \forall t \in T, i = 4, 5,$$

where

$$(A_1)_t = [1, -\frac{1}{e}, e], (A_2)_t = [0, 1, -1], (A_3)_t = [0, -1, -1],$$

$$(A_4)_t = [0, 1, -1], (A_5)_t = [0, -1, 1],$$

$$(\boldsymbol{b}_1)_t = [d_t - g_t], (\boldsymbol{b}_2)_t = [B_t - B_{t-1}], (\boldsymbol{b}_3)_t = [-P], (\boldsymbol{b}_4)_t = [-B_{-1}], (\boldsymbol{b}_5)_t = [B_{-1}] - c],$$

and we also require $\boldsymbol{x} \in (\mathbb{R}^+)^{|T|}$.

Now that we have formulated our linear program, we can solve it through gurobipy. Suppose at some point in time we know the electricity prices p_t^c, our electricity demands d_t, and PV generation g_t for all $t \in T$, with them being shown in the plots displayed in Figure 4. Suppose

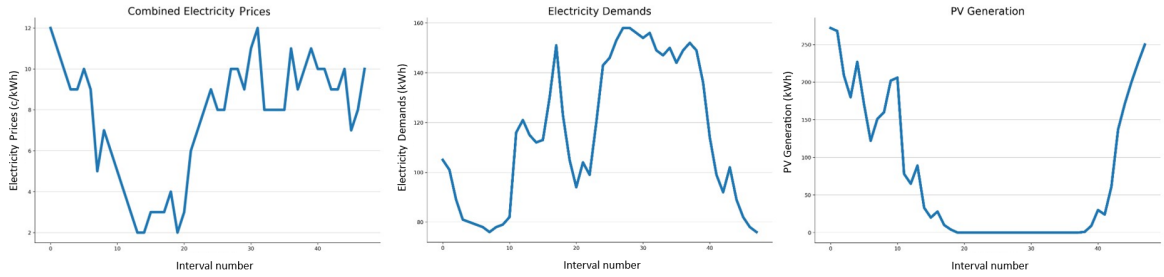


Figure 4: Electricity Prices (left), Demands (centre), and PV generation (right) over a 24-hour period.

that our battery's capacity is 128kWh, with its power being 32kW, its charging and discharging efficiency being (unrealistically) 0.999, and its initial state of charge being 50% (64kWh). Then, by plugging in all of these details into the linear program that we detail in Equation 5 above, the optimal solution and costs are given by Figure 5. Given that the PV generation exceeds the electricity demand within the first few intervals, at the beginning the battery can be both charged or discharged without impacting our costs (which stay at \$0), as it still gets fully charged by the PV generation given the amount of excess PV generation (in reality,

we would try to minimise these charge/discharge cycles to preserve long-term battery life). However, once electricity prices and demands begin to rise, and PV generation decreases, the battery begins charging and discharging more selectively. Given that its capacity isn't high enough to meet the demand, the battery is discharged when the price is at its highest, and is charged when the price is relatively low. While PV generation exceeds energy demand towards the end of this period, given that we are only optimising for 48 intervals the battery does not get charged as PV generation suffices during the final intervals.

Hence, using our above program and the future energy demands, prices and PV generation

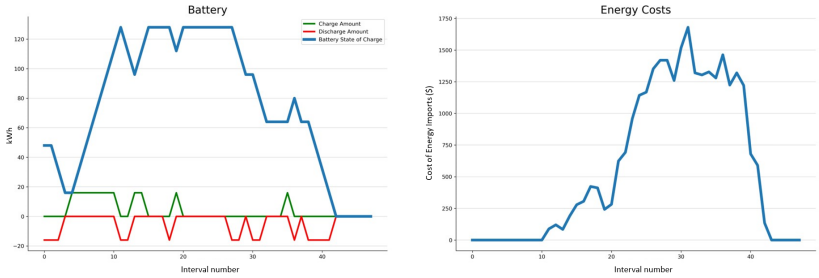


Figure 5: Corresponding optimal battery charge/discharge and SOC (left), and corresponding electricity costs (right) over a 24-hour period.

we can easily optimally schedule our BESS. However, in the reality, we would not know these in advance, as they are given by the outcomes of random events.

This realisation means that a large number of the papers that have been published, including [28] and [31], do not evaluate their algorithms' expected real-world performances, as by using deterministic forecasts (and treating them as if they are the future outcomes) they don't incorporate the uncertainty of future outcomes for the constraints and objective function in their respective models.

In [31], a deterministic LP approach is employed to derive an 'optimal' schedule, ensuring the satisfaction of demand and maximisation of profit for BESS used in conjunction with conventional energy sources. The paper provides a detailed formulation, computational complexity analysis, and a numerical example. However, it only presents numerical results when future electricity demands, prices, and PV generation are known in advance, therefore, this paper does not show the effectiveness of the their method in real-life scenarios where these factors are random variables.

Another article, [28], utilises a deterministic model for this problem but incorporates economic and environmental aspects, including CO_2 emissions. It employs a multi-objective optimisation technique, combining different objectives to achieve simultaneous cost reduction and CO_2 emissions reduction. However, the paper's objectives differ from our purely financial focus. Furthermore, similar to [31], they do not consider random variables as inputs for electricity prices, demands, and PV generation, limiting the applicability of their results.

Hence, while a LP with the future electricity demands, prices, and PV generation as inputs would yield us the best possible battery schedule, this is actually unrealistic, as all of those inputs are unknown in advance. Therefore, the first step towards solving the BESS control problem is being able to somehow predict (or forecast) the future outcomes of each of our variables, as we need those as inputs to our LP.

3 Forecasting

While the focus of our thesis is not on forecasting, forecasts are required as inputs to our problem so that we can optimally schedule the battery. As without forecasts, we cannot tell how our system would perform in real-world scenarios where future events are unknown.

In Section 2, we address that in order to optimally schedule our BESS using our LP formulation we need to input the deterministic variables $(p_t^c, d_t, g_t)_{t=0,\dots,H-1}$ into the model. However, this requirement is unrealistic as energy prices, demands, and PV generation at each interval are actually random variables - unknown in advance. Therefore, we cannot refer to these variables as deterministic.

We refer to these random variables using the tuples $\boldsymbol{X}_i = (X_t)_{t=i,\dots,i+H-1} = (X_t^{p^c}, X_t^d, X_t^g)_{t=i,\dots,i+H-1}$, where each element in each tuple for each interval is a random variable. We define X_t over the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with $X_t : \Omega \to S$, where S is the space of all the possible values that the tuple $(X_t^{p^c}, X_t^d, X_t^g)$ can take, as we define in Section 2 (later we refer to S as our state space, which we define in Section 6.1.0.2). The values that these random variables can take are consistent with those that we define in Section 2. Hence, in order to be able to utilise our methodology from Section 2, we need to input point values representing these random variables' future trajectories (or their random paths) into our LP solver. Given that our aim is to minimise the expected costs, we seek to output random sample paths that correspond to the expected future values of the random variables that are critical to the control of our BESS system. While these are random variables, they depend on (or, are correlated with) certain parameters. Suppose that $\boldsymbol{x}_i$ is the vector of parameters that $\boldsymbol{X}_i$ depends on, where i is some interval in time. Then, we need to create a forecasting model, $f(\cdot)$, that utilises the available exogenous information, $\boldsymbol{x}_i$ for some time interval i, in order for its output to be the expected values of X_i,

$$f(\boldsymbol{x}_i) = \{(\hat{p}_t^c, \hat{d}_t, \hat{g}_t) : t \in \{i, \dots, i + H - 1\}\}, \tag{6}$$

representing the expected values of our the random variables that we input into our LP, with $\mathbb{E}[X_i] = f(\boldsymbol{x}_i)$. We can then input these values into our LP in order to find how we should optimally schedule our BESS.

Hence, while we are not aiming to create a state-of-the-art forecaster, but rather a BESS

control mechanism, we need to create a series of these forecasts using which we can test our BESS control mechanism to determine how it would perform in the real world where these variables are unknown and only forecasts exist. While some papers, such as [39], assume that we can simply simulate our forecasts by by taking historical measurements $(p_t^c, d_t, g_t) : t \in \{i, \ldots, i + H - 1\}$ and generate a set of forecasts from those such that the forecasts are

$$(\hat{p}_t^c, \hat{d}_t, \hat{g}_t) : t \in \{i, \ldots, i + H - 1\} = (p_t^c + Z_t^p, d_t + Z_t^d, g_t + Z_t^g) : t \in \{i, \ldots, i + H - 1\}, \tag{7}$$

where $Z_t^p \sim N(0, \sigma_p), Z_t^d \sim N(0, \sigma_d), Z_t^g \sim N(0, \sigma_g), \sigma_p, \sigma_d, \sigma_g \in \mathbb{R}^+$, meaning that we generate forecasts by adding independent Gaussian noise. However, as we show in Section 5.1, such approaches run the risk of inaccurately modelling the forecasts. This, in turn, means that the system would not be well adjusted to utilise the real forecasts that it would eventually have to deal with.

As each of the three elements within each tuple (i.e. $(X_t^{p^c})_{t=i,\ldots,i+H-1}, (X_t^d)_{t=i,\ldots,i+H-1}, (X_t^g)_{t=i,\ldots,i+H-1}$) is a unique random variable, with one corresponding to the electricity prices, one corresponding to the electricity demands, and the final one to PV generation, we need to create a different forecasting model for each of the three series of random variables, before finally combining the models in order to output the complete set of required data in order for us to be able to schedule our model.

Hence, over Sections 3.1, 3.2, and 3.3 we seek to explore how we can create these forecasts so that we can test how our formulation from Section 2 performs using our forecasts, as we do later in Section 4.

3.1 Generating PV Forecasts

PV generation measures the amount of energy that solar panels can produce due to the sun's energy. The amount of PV generation varies according to some stationary factors, such as the size of the solar panels, and their angle, as well a series of deterministic factors, such as the time of the day and the time of the year. However, producing accurate PV generation forecasts is a complex process, largely dependent on stochastic factors such as cloud opacity. While a methodology for achieving accurate PV generation forecasts is demonstrated in [34],

it's essential to note that meteorological point forecasting is not our focus, hence, we do not seek to generate these.

Rather, the historical point forecasts, along with the real measurements that we utilise are from a forecasting software called "Solcast". Solcast's forecasting accuracy is reported to be very high, with its Mean Absolute Percentage Error (MAPE) for the day-ahead hourly-forecast being 5.9% - varying from 2.1% to 9.4%, according to Solcast's website [29]. However, as historical forecasts are not provided by Solcast, in order to collect this data we need to store the live forecasts over an extended period. We achieve this by creating a Python program that collects the live forecasts from Solcast through an API, before deploying this function as a cloud function through Microsoft's Azure, where it is run according to a 30 minute timer (which matches the granularity of the time intervals in our models), saving the captured data in an SQL table which is also hosted by Azure.

Running this function over the course of more than six months for five different sites gives us data to comprehensively test our optimisation frameworks. Our data was collected from December, 2022 until June, 2023, thus ensuring that it captures the solar generation between the summer and winter solstice, allowing the forecasting results to be extrapolated to the whole year. As we observe in Figure 6, the locations of the data measurements are scattered around South Eastern Australia. Hence, we have the historical expected values of the stochastic processes $(X_t^g)_{t=i,\dots,i+H-1}$, along with the actual measurements, $(g_t)_{t=i,\dots,i+H-1}$ for times from December, 2022 until June, 2023. In order to accurately model the expected forecasts of the random processes, the data needs to be processed. First, we fit it into 48 interval blocks of forecasts and actuals, in order to reflect our problem horizon. Secondly, we normalise the data for all the sites to fit a 100kWh site. Formally, this means that for a xkWh site, we multiply all the historical PV generation measurements and forecasts by a factor of $\frac{100}{x}$. Hence, our historical PV generation data, $\mathcal{G}$, consists of the data for each of our 5 sites. Here, $\mathcal{G} = \{(\hat{\mathbf{g}}_t, \mathbf{g}_t) : t \in (12/22 \text{ - } 6/23)\}$, and $\hat{\mathbf{g}}_t, \mathbf{g}_t \in (\mathbb{R}^+)^H$, where $\hat{\mathbf{g}}_t$ is our vector of estimates for X_t^g, while $\mathbf{g}_t$ is the actual values that X_t^g took. As these are overlapping measurements and corresponding forecasts, which we demonstrate in Figure 7, we have $\mathbf{g}_{t,h+s} = \mathbf{g}_{t+s,h}$, $\forall\, 0 \leq h+s \leq H-1$, where H is the forecasting horizon length, and $\hat{\mathbf{g}}_t$ are the expected values for the PV generation for time t to $t+H-1$, made at time t.

Figure 6: Locations of solar generation data collection. https://earth.google.com/web/

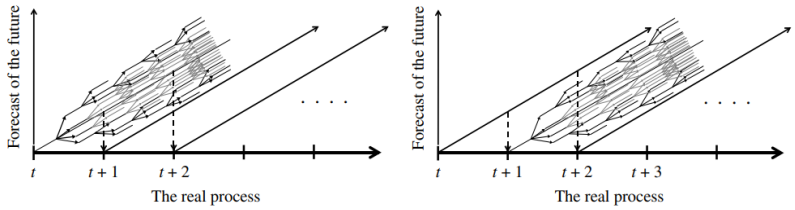


Figure 7: Illustration of a rolling horizon procedure using a stochastic model of the future. [27]

To demonstrate how the PV forecast trajectories compare to the real data when we iteratively plot the historical PV forecasts $\hat{\mathbf{g}}_{t+i}$ over the next $H - i$ intervals, for $i = 0, \ldots, H - 1$ when we are forecasting at interval t, and we compare this to the real data, $\mathbf{g}_t$, which we plot below in Figure 8 for one of our locations. We generate PV generation forecasts over overlapping intervals to demonstrate how our forecasts change over time. We plot every fourth forecasted trajectory as plotting each one leads to too much clutter. The aim in displaying this graph is to show how as we get closer to the forecasted interval, forecasts tend to get more accurate, but also to show how each forecast can be seen as some random sample path for the stochastic process. This helps to illustrate the sequential aspect of our decision making, where as we get new data at each interval our optimal decision will change given the new information that we have. The change in our decision making based on this new information is known as recourse.

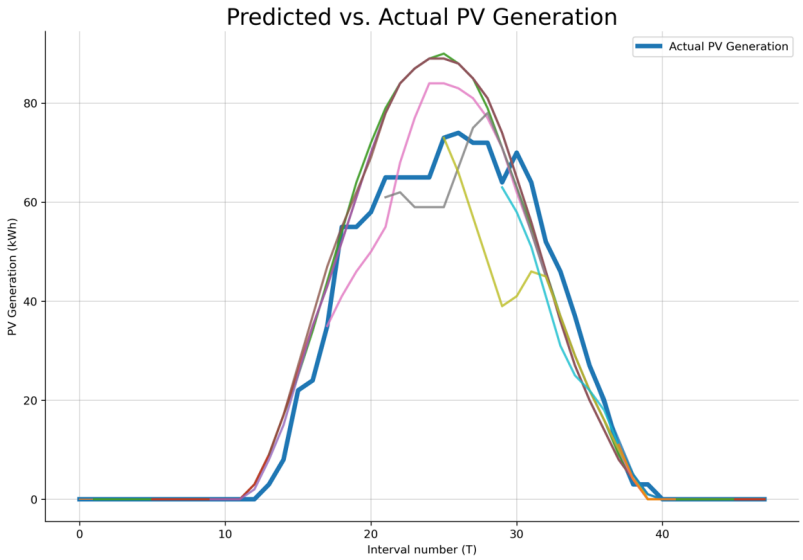


Figure 8: Iteratively Forecasted PV Generation vs. Actual PV Generation for 100kWh Solar Panels
(each unlabelled curve starting at interval i represents the forecast generated, $\hat{\mathbf{g}}_{t+i}$).

Now that we have historical PV generation data and corresponding forecasts, we can use them to test our optimal control framework.

3.2 Generating Demand Forecasts

In this section, we discuss our electricity demand forecasting methodology. For now, the aim of our forecasts is to find the expected values of the stochastic process representing our energy demands, $(X_t^d)_{t=i,\ldots,i+H-1}$. Unlike the PV generation which is sourced from a third-party provider, the demand forecasting is done fully within our thesis, we don't have access to generated historical forecasts.
We have access to a year's worth of demand data for a few commercial enterprises around South Eastern Australia. However, as demand profiles vary widely by company (as seen in Figure 9), the demand needs to be modelled for each site individually.

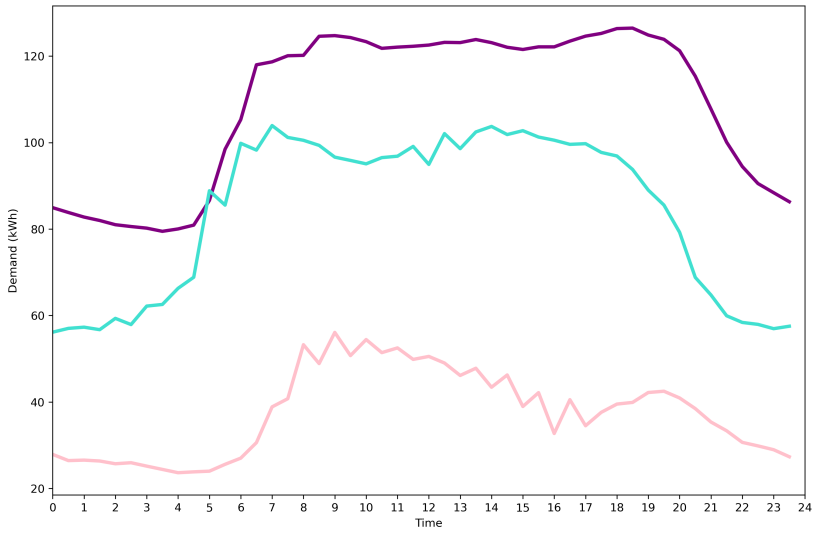


Figure 9: Average hourly demands for the three companies in South Eastern Australia.

Firstly, however, we need to formally define our available data. Our demand data consists of $\{D_i\}_{i=1,2,3} \in \mathcal{H}$ for each of our three companies. Here, $D_i = \{d_{i,t} : t \in \text{year}(2022)\}$, as our demands are from the entirety of year 2022.

While we don't need state-of-the-art forecasts, we need them to at least somewhat correspond to what real-life forecasts would look like, hence, we first evaluate some existing literature that deals with electricity demand forecasting.

In [10], Fan, S., and Hyndman, R.J. propose a 'semi-parametric additive model', with a separate forecaster created for each 30-minute period, with predictions up to 7 days ahead. The logarithm for the predicted demand is given as the sum of functions for three separate variables: calendar effects; temperature effects; and the effects of recent demands. According to [10], the best model for this application contains the following variables:

- Demands around the same time period for the last 2 days;
- The maximum demand in the last 24 hours;
- The minimum demand in the last 24 hours;
- The average demand in the last seven days;
- The current temperature and temperatures from the last half-hour period;

- The current temperature differential;
- The maximum temperature in the last 24 hours;
- The minimum temperature in the last 24 hours;
- The average temperature in the last seven days;
- The day of the week;
- The holiday effect;
- The day of year effect.

However, as their model is used for NEM demand forecasting, a couple of public temperature forecasts suffice for them, whereas for our model, we seek temperature forecasts at the precise locations of the customers whose loads we are forecasting. And, as we do not have access to those, our model cannot contain the temperature variables. Some public forecasts do exist, however, obtaining historical temperature forecasts for specific locations (especially outside major cities) is challenging, with some being behind a paywall. Perhaps including the temperature data can be an extension to our model.

The forecasting method in [10] would likely produce significantly different results in our thesis, since in [10] it is used to predict the total load of the NEM, which features significantly less volatility and unpredictability than most single consumer loads would, as consumers' loads clearly are not perfectly correlated. While we cannot test the exact model given that we do not have the specified temperature data, the modified model does not perform well, with the model that we describe below outperforming it.

Mathematically, the reduced variance of the NEM (relative to the total load) compared to most consumers can be shown through Bienaymé's identity, with there being $n > 0$ consumers' loads given by the random variables L_i, each with a variance σ_i^2 for all consumers $i = 1, \ldots, n$. Each L_i is a random variable representing the energy load of consumer i at a given time. Hence, the average variance over all of the consumers is $\bar{\sigma}^2$ and the correlation between any two consumers, i and j, being $|\rho_{i,j}| \in (0, 1)$, then the total average variance is

given by:

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} L_i\right) = n^{-2}\sum_{i,j=1}^{n}(\rho_{i,j}\sigma_i\sigma_j) < n^{-2}\sum_{i,j=1}^{n}(\sigma_i\sigma_j) \text{ (as } \rho_{i,j} < 1) \tag{8}$$

$$= n^{-1}\sum_{i=1}^{n}\left[\sigma_i \cdot n^{-1}\sum_{j=1}^{n}\sigma_j\right] = n^{-1}\bar{\sigma}\sum_{i=1}^{n}\sigma_i = \bar{\sigma}^2 \tag{9}$$

This means that our forecasting task in this thesis is likely to be more difficult as the data features more volatility. While we do not know the values of $\rho_{i,j}$, as a basic reference, the correlations of the datasets that we have access to have $\rho < 0.5$, which would lead to:

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} L_i\right) < \frac{1}{2}\bar{\sigma}^2. \tag{10}$$

As our forecasting problem features significantly more volatility, this might make the optimal parameters and forecasting model for our problem differ from those presented in [10], as their parameters and model.

However, given that aim is not to create a state-of-the-art forecaster, but rather one that generates realistic forecasts, the exogenous variables that we use predict future demands are similar to those in [10]. They are:

- demands around the same time of the day for the last two days;
- the maximum demand in the last 24 hours;
- the minimum demand in the last 24 hours;
- the average demand in the last 24 hours;
- demand around the same time of the day seven days ago;
- the maximum demand in the last seven days;
- the minimum demand in the last seven days;
- the average demand in the last seven days;

- the holiday and weekend effect;
- the time of the day, represented by cosine and sine functions;
- the day of the year, represented by cosine and sine functions;

As we don't have access to the temperature data for the sites over the periods that the demands were collected, temperature related variables are not used. However, as seasonality does have a significant impact on the electricity demands, as we demonstrate in Figure 10, variables for the time of the year are included.

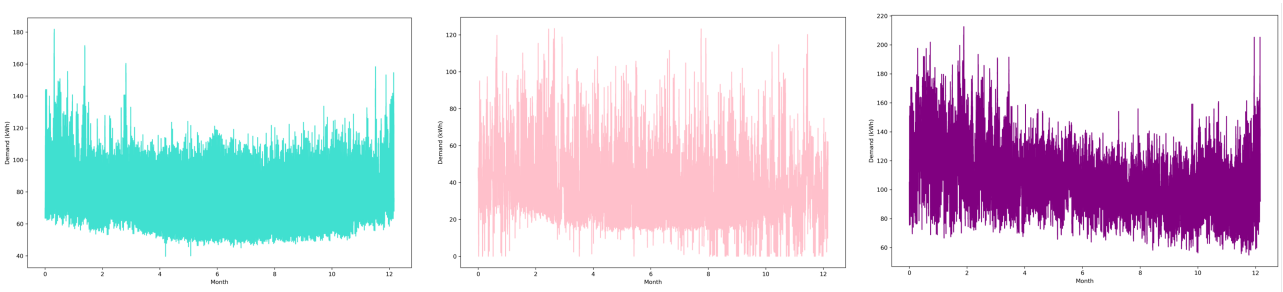


Figure 10: Yearly electricity demands for three companies in South Eastern Australia.

As a year is cyclic in nature, this data is represented through cyclic functions: cosine and sine. Similarly, as is shown in Figure 9, demands vary heavily according to the time of the day. Hence variables representing the time of the day are also included. They are also represented through sine and cosine functions.

When it comes to choosing the regression model, we use the regression-tree based gradient-boosting model. Regression trees date back to the 1963 paper published about "Automatic Interaction Detection" by Morgan, J.N. and Sonquist, J.A. [21], and have been at the fore of forecasting since the 1984 release of the renowned textbook "Classification and Regression Trees" by Breiman, L., *et al.* [6]. Regression trees work in the following way:

1. We have all of the past outcomes for our target features (such as electricity demands), given by $\boldsymbol{y}$ (this can correspond to the past measurements, such as D_i). The types of parameters that are used to forecast the values in $\boldsymbol{y}$ are given by types($\boldsymbol{x}$) (these include

parameters such as the time of the day, recent energy demands, etc.), with some of these parameters are discrete (i.e. day of week) and therefore have a discrete set of possible 'labels' that they can take). The values of these parameters corresponding to each measurement in $\boldsymbol{y}$ are given by $\boldsymbol{x}$. We refer to one value in $\boldsymbol{y}$ (i.e. $\boldsymbol{y}_i, i \in \{1, \ldots, |\boldsymbol{y}|\}$) and the corresponding exogenous variables in $\boldsymbol{x}$ (i.e. $\boldsymbol{x}_i$) as an instance.

2. We initialise our tree, T, with only one node, R, corresponding to the root, and we refer to the target features at a node, N, as $\boldsymbol{y}_N$, and to the exogenous variables as $\boldsymbol{x}_N$. We also have a stopping criteria, given by ϵ (we define it below in Equation 11), and, at first we have $\boldsymbol{y} = \boldsymbol{y}_R$, and $\boldsymbol{x} = \boldsymbol{x}_R$.

3. If for all terminal (leaf) nodes N in our tree T,

$$\epsilon > Var(N) = \frac{1}{|\boldsymbol{y}_N|} \sum_{y \in \boldsymbol{y}_N} \left(y - \frac{1}{|\boldsymbol{y}_N|} \sum_{y \in \boldsymbol{y}_N} y \right) \tag{11}$$

, then we stop and our tree is complete.
Else, we find a node $N \in T$ such that $Var(N) \geq \epsilon$ and we follow the splitting procedure:

4. For every exogenous variable type $v \in \text{types}(\boldsymbol{x})$: If v is discrete (such as for the day of the week), then for every label $l \in v$ we split our data into $\boldsymbol{y}_N^l$ and $\boldsymbol{y}_N^{l'}$, to signify the data whose exogenous variable v is labelled l, and the data that isn't labelled l. We then find

$$Var(N_{v,l}) = \frac{1}{|\boldsymbol{y}_N^l|} \sum_{y \in \boldsymbol{y}_N^l} \left(y - \frac{1}{|\boldsymbol{y}_N^l|} \sum_{y \in \boldsymbol{y}_N^l} y \right) + \frac{1}{|\boldsymbol{y}_N^{l'}|} \sum_{y \in \boldsymbol{y}_N^{l'}} \left(y - \frac{1}{|\boldsymbol{y}_N^{l'}|} \sum_{y \in \boldsymbol{y}_N^{l'}} y \right) \tag{12}$$

. Hence, we iterate through each individual label for each variable to determine whether splitting according to that unique label is optimal.

5. If the variable v is continuous (such as the most recent demands), we then sort all its values in ascending order. Then, we iterate over all of its values $v_1, \ldots, v_n$ and for any two adjacent values v_j and $v_j + 1$, we split our data according to whether the value of its variable $v \in \text{types}(\boldsymbol{x})$ is less than or equal to v_j (we label it l), or greater than v_j (we label l'). We then find the variance of every such split according to Equation 12. Of course, this can be very computationally intensive, hence, newer methods do not attempt every such split, but rather first split the variable's values into a number of

bins, thus reducing the number of splits that are attempted (if this is used, then the tree is a histogram-based regression tree).

6. Finally, we split our data according to whether for any given instance its variable v is labelled v_l, where $v, v_l = \arg\min Var(N_{v,v_l})$. Thus, from node N we create two children nodes C_1^N and C_2^N, with the instances in C_1^N having their variable v labelled v_l, and the others having their variable v not labelled v_l (labelled $v_{l'}$).

Then, when we want to predict an electricity demand based on some instance's exogenous features $\boldsymbol{x}_i$, we follow the regression tree T from the root node according to the split that we use (i.e. whether variable v's label is v_l, where v and v_l are stored at each node). We then output the prediction of the resulting terminal node.

Another way of looking at it is that the tree allows us to split our exogenous variable space into a number of non-overlapping sets, corresponding to the leaves of our tree, with a predicted target value associated with each of these sets, as we observe in Figure 11.

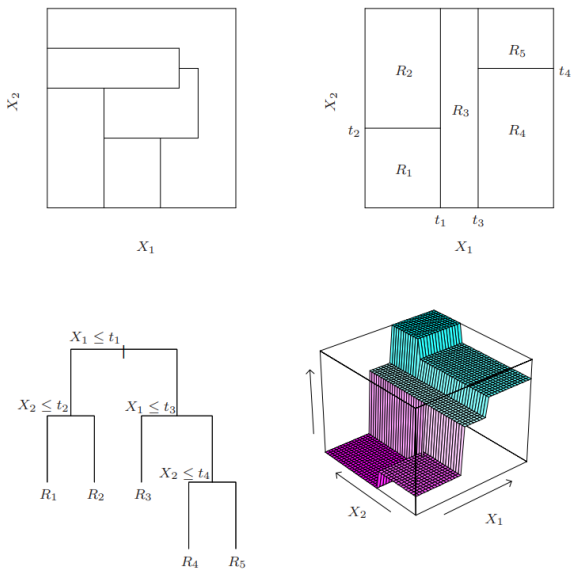


Figure 11: Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel. [12]

As we mention in 3.2, if a variable v has L labels, $v_1, \ldots, v_L$, then we attempt the splits $(\{v_1\}, \{v_2, \ldots, v_L\}), (\{v_2\}, \{v_1, v_3, \ldots, v_L\}), \ldots, (\{v_L\}, \{v_1, \ldots, v_{L-1}\})$. Of course, we see that we only attempt $|L|$ splits at each node, when there is an exponential number of splits to try. To overcome this we can try a series of random splits by choosing more than one variable to be on either side of a split (i.e. $l = (l_1, \ldots, l_j), l' = (l_j + 1, \ldots, l_k)$ for some labels of the variable), and then splitting according to the best of those. Hence, we can build a series of different trees by randomly comparing a series of random splits for each tree, by combining the forecasts of these different trees we can create an ensemble regression tree forecast.

Regression trees can be used together with gradient boosting. As defined by Friedman, J.H. in [11], gradient boosting constructs additive regression models by sequentially fitting a simple parametrised function (in our case a regression tree with a random split) to current "pseudo"-residuals by least squares at each iteration (making it an ensemble forecaster). The pseudo-residuals are the gradient of the loss functional being minimised, with respect to the model values at each training data point evaluated at the current step.

The intuition behind this method is that for each instance i's value that we want to predict (we define our instance as is stated in 3.2), our loss function is the mean squared error and, thus, is given by $L(\boldsymbol{y}_i, f(\boldsymbol{x}_i)) = (\boldsymbol{y}_i - f(\boldsymbol{x}_i))^2$ (we use the average over all instances i), where $f(\cdot)$ is the function that uses the exogenous data $\boldsymbol{x}_i$ (as we define in 3.2) to predict $\boldsymbol{y}_i$. We also know that if we want to minimise $L(\boldsymbol{y}_i, f(\boldsymbol{x}_i))$, then we need to first find $h = \frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}$ (corresponding to the steepest descent direction), before setting $f(\boldsymbol{x}_i) \rightarrow f(\boldsymbol{x}_i) + h \cdot \gamma$, where

$$\gamma = \arg\min_{\gamma} L(\boldsymbol{y}_i, f(\boldsymbol{x_i}) + h \cdot \gamma). \tag{13}$$

Hence, to train the regression-tree gradient boosting model, f for data $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ using the MSE loss function, we do the following:

1. We start by setting $f_0(\boldsymbol{x}_i) = \frac{1}{n} \sum_{i=1,\ldots,n} y_i$. Then, we perform M iterations (where we choose M).

2. In each iteration m from 1 to M, we set $y_i^m = y_i - f_{m-1}(\boldsymbol{x}_i) \forall i = 1, \ldots, n$. We then train a regression tree, $h_m(\cdot)$ according to the method that we detail above (usually, however, using random splits) using the data $(\boldsymbol{x}_i, y_i^m)_{i=1,\ldots,n}$. We then set $f_m(\boldsymbol{x}_i) = f_{m-1}(\boldsymbol{x}_i) +$

$\gamma \cdot h_m(\boldsymbol{x}_i)$. As in Equation 26, $\gamma = \arg\min_\gamma \frac{1}{n} \sum_{i=1,\ldots,n} L(\boldsymbol{y}_i, f_{m-1}(\boldsymbol{x_i}) + h_m(\boldsymbol{x_i}) \cdot \gamma)$.

3. Once we have finished all M iterations, f_M is our final forecasting model.

There are many variants of this algorithm based on different loss functions, as well as different specified parameters, such as for ϵ, or the maximum allowed depth for the trees. As we mention above, the regression-tree gradient boosting model is known to work very well, even for the prediction of consumer energy demands [22, 33], PV generation [32, 36] and electricity spot market prices [4], justifying its usage in our thesis.

We then train a series of $H - 1$ models for each interval ahead of time that we are forecasting for, given that the closer to the current interval that we are forecasting for the more accurate our forecasts are likely to be and, therefore, their optimal hyper-parameters might differ. We build our models using Python's sklearn's ready-to-use "GradientBoostingRegressor" module. We also control variable selection and hyper-parameter optimisation using random cross validation, however, that is beyond the scope of our thesis.

To demonstrate how our demand forecast trajectories compare to the real data when we iteratively forecast the next $H - i$ intervals when we are forecasting at interval $i \in T$, we plot it below in Figure 12 for one of our companies (we plot every fourth forecasted trajectory as plotting each one leads to too much clutter). We generate energy demand forecasts over overlapping intervals to demonstrate how our forecasts change over time. Our graph is structured in the same way as 8, except that now we iteratively plot the historical demand forecasts $\hat{\mathbf{d}}_{t+i}$ over the next $H-i$ intervals, for $i = 0, \ldots, H-1$ when we are forecasting at interval t, and we compare this to the real data, $\mathbf{d}_t$, which we plot below in Figure 12 for one of our locations.
Now, we also have a method for generating demand forecasts, corresponding to the expected values of the stochastic process $(X_t^d)_{t=i,\ldots,i+H-1}$ for every time interval i, which we can then feed into the LP that we define above in order to find the optimal battery schedule at time interval i based on the H expected values of this process that we input into our LP.

3.3 Generating Electricity Price Forecasts

The final forecast that we need is the electricity spot price forecast, equivalent to the expected values of the stochastic process representing our energy prices, $(X_t^p)_{t=i,\ldots,i+H-1}$ (we can convert X^p to X^{p^c} by adding the deterministic network charges). Before we can delve into our electricity spot price forecasting methodology, we first detail how we obtain the spot price

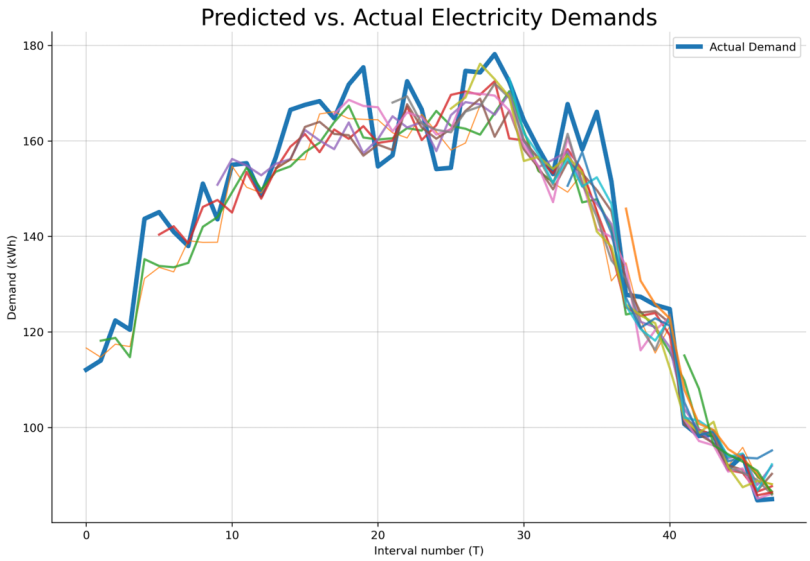


Figure 12: Iteratively Forecasted Demands vs. Actual Demands for Energy Consumer (each unlabelled curve is a forecast of the random process generated at a different time).

data required to generate our forecasts and, subsequently test our model.

All of the data that we use originates from AEMO's archive reports website (http://www.nemweb.com.au/REPORTS/ARCHIVE/), where data from the past year is stored in a series of Excel files, and is free to access. This includes the actual spot prices for every 5 minute period (which we average over 6 periods to convert it to 30-minute periods), as well as what is known as "Pre-dispatch" reports. These reports are released every 30 minutes and contain the forecasts over the course of 16-24 hours (until 4:00 AM) for a series of variables. Among others, the forecasts include the following variables:

- Energy prices (this is only meant to serve as a rough indicator);
- Total NEM demand (electricity demand);
- Available generation (electricity supply);
- Semi-Scheduled (electricity generators with intermittent output, such as wind and solar farms).

Unfortunately, this data is difficult to access, given that to get a year's worth of forecasts we need to download and append data from over 17,000 Excel files. Hence, to overcome this challenge we first create a Python web-scraping application that can automatically download

all of the required folders, unzip them, then access their Excel files (using a series of code patterns for the names), before then filtering the data, cleaning it, and finally saving the combined data for all of the files.

Now that we have all of the required historical data we can look at how we can generate forecasts for it, given that the preliminary AEMO energy price forecasts are extremely inaccurate as they are meant based on the registered supply and demand which is usually adjusted, as we observe in Figure 13 for some given day. As in Figures 8 and 12, at interval $i \in \{0, \ldots, 31\}$ we are plotting AEMO's forecasts of the spot prices for the remaining $31 - i$ intervals remaining within our selected 16-hour window to demonstrate how inaccurate the forecasts are.

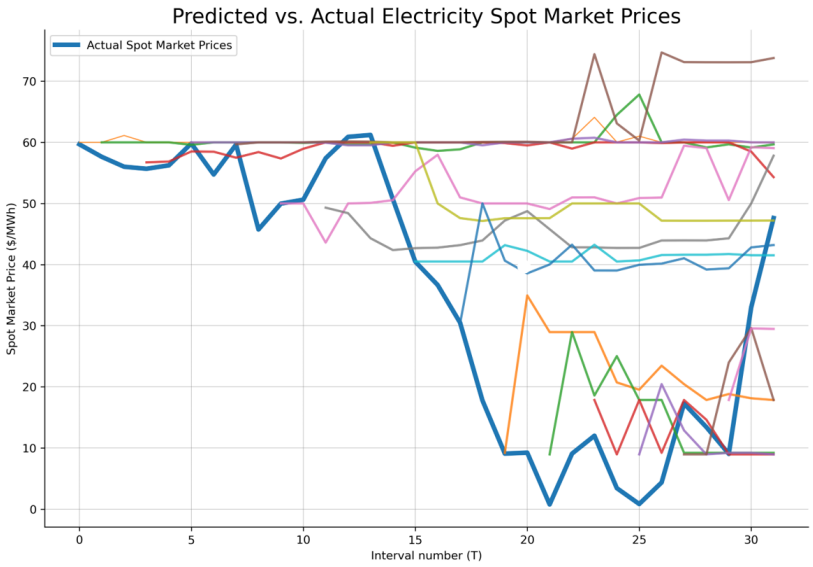


Figure 13: AEMO Iteratively Forecasted NEM Wholesale Spot Market Prices (labelled) vs. Actual Wholesale NEM Spot Market Prices (unlabelled).

While, as mentioned above, forecasting is not the main aim of our thesis, it is still important to understand what some of the current electricity pricing models are in order to create a test our optimisation framework using forecasts that reflect those that it would get in reality.

In [24], a forecasting model is built using two logistic regression models which are given by a sum of functions of the preceding prices. With one giving the centred log-price on day d

and hour h as

$$p_{d,h} = \beta_{h,1} p_{d-1,h} + \beta_{h,2} p_{d-2,h} + \beta_{h,3} p_{d-7,h} + \beta_{h,4} p^{min}_{d-1,h} + \beta_{h,5} z_{d,h} \tag{14}$$

$$+\beta_{h,6} D_{Sat} + \beta_{h,7} D_{Sun} + \beta_{h,8} D_{Mon} + \varepsilon_{d,h}, \tag{15}$$

where the lagged log-prices $p_{d-1,h}$, $p_{d-2,h}$ and $p_{d-7,h}$ account for the autoregressive effects of the previous days, $p^{min}_{d-1,h}$ is the minimum of the previous day's 24 hourly log-prices, $z_{d,h}$ is the logarithm of the zonal load forecast for day d and hour h (exogenous variable), D_{Sat}, D_{Sun} and D_{Mon} are dummy variables that account for the weekly seasonality (Saturday, Sunday, and Monday) and the $\varepsilon_{d,h}$'s are assumed to be independent and identically distributed Gaussian variables.
The second model is the same, except that it replaces $\beta_{h,1}$ with $\left(\sum_{i \in I} \beta_{h,1,i} D_i\right)$ and adds $\beta_{h,8} D_{Mon} p_{d-3,h}$ in order to account for the autoregressive weekly effects, with $I = \{0, Sat, Sun, Mon\}$, with $D_0 = 1$.

While the paper's models seem impressive, a slight modification could significantly improve them, as in their current form they do not include the latest measurements. What this means is that the use information that lags 24-hours behind the latest, thus, perhaps an extension would be to include the variable $\beta_{h,12} p_{d,latest}$, which would capture the latest available data. While this would be unlikely to significantly improve the accuracy for when forecasting 12+ hours ahead of time, it would be very likely to improve accuracy for the first few hours, given that the autocorrelation is higher in the hours leading up to the forecasting window, as we show for NEM Wholesale Spot Market prices in Figure 14.

A key limitation of these models, however, is their lack of use of exogenous variables. Given that the electricity price is directly determined by the supply-demand market dynamics, as we detail in Section 1.2.1, it seems important to have some forecast for the available electricity generation (supply) and registered electricity demand, given that it is known in advance. Furthermore, having the time of the day and the time of the year, given that solar generation forms a key part of the total available generation and it varies significantly based on those factors.

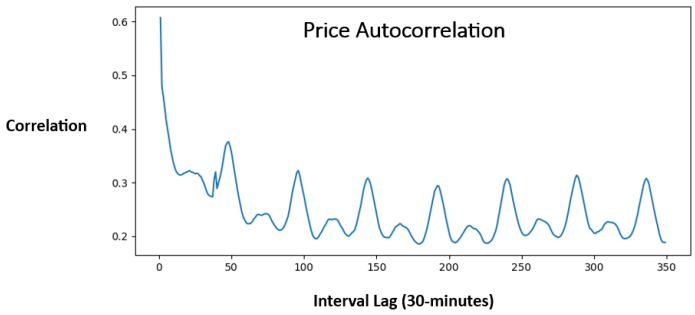


Figure 14: Autocorrelation from Victorian Spot Prices (2022).
Constructed using data from http://www.nemweb.com.au/REPORTS/ARCHIVE/PreDispatchIS_Reports

Hence, for our thesis we choose to build a different model, which we detail here. The regression model framework that we use is the regression-tree gradient boosting model that we define in Section 3.2. Our exogenous variables include the following:

1. Average AEMO pre-dispatch predicted energy price over three intervals around the forecast intervals;
2. AEMO pre-dispatch predicted energy price for the interval being forecasted for ;
3. AEMO pre-dispatch total predicted NEM demand;
4. AEMO pre-dispatch total predicted available generation ;
5. AEMO pre-dispatch predicted semi-scheduled generation;
6. Time of the day, represented by cosine and sine functions;
7. Day of the year, represented by cosine and sine functions;
8. Dummy variables for the day of the week;
9. Public holiday effect;
10. Latest known energy price;
11. Second latest known energy price;
12. Average energy price for the latest day;

13. Actual energy price 24 hours before forecasted interval;

14. Actual average energy price over the latest week;

15. Actual energy price for interval a week before the one being forecasted.

However, given that the pre-dispatch reports are sometimes only available for 16 hours, we do not include variables 1-5 for forecasted intervals that are over 16 hours ahead. Then, as we do for our demand forecasting, we train a series of $H-1$ models for each interval ahead of time that we are forecasting for. We, again, build our models using Python's sklearn's "GradientBoostingRegressor" module. We also control variable selection and hyper-parameter optimisation using random cross validation.

To demonstrate how our energy price forecast trajectories compare to the real data when we iteratively forecast the next $H-i$ intervals when we are forecasting at interval $i \in T$, we plot it in Figure 15 below (again, we plot every fourth forecasted trajectory as plotting each one leads to too much clutter), the same way as we plot Figures 8, 12 and 13.

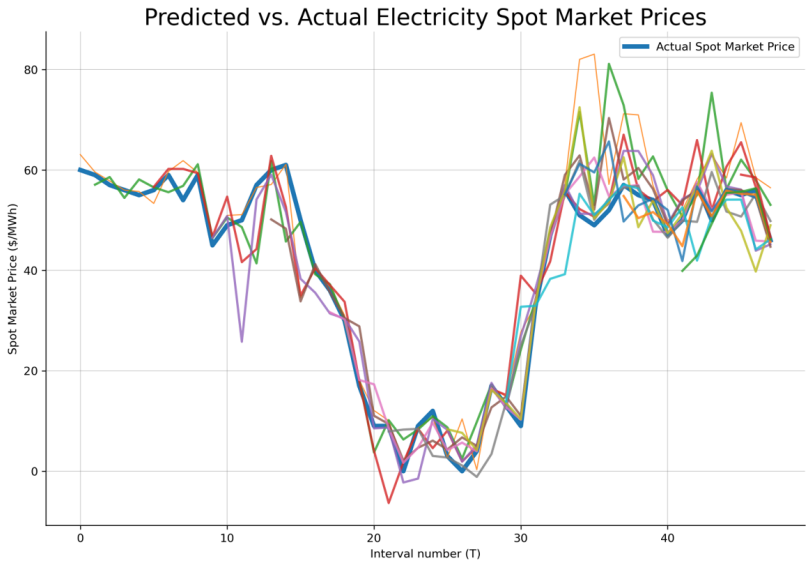


Figure 15: Iteratively Forecasted Prices vs. Actual Prices.

Now, we also have a method for generating price forecasts (i.e. finding the expected values

of the random variables that give our energy prices) which we can then feed into the LP that we define above in order to find the optimal battery schedule. We can now use all three of our forecasting methods together to schedule our BESS through a model predictive control framework, as we detail in Section 4, given that we have a method of finding the expected values of the random processes whose outcomes our decision making depends on.

4 Model Predictive Control

Now that we have a method for generating the point forecasts that our LP needs in order to optimally schedule our battery, we can create our BESS deterministic model predictive control (MPC) framework.

MPC, also known as a look-ahead policy and a receding horizon procedure, utilises forecasts to control the system effectively. As formally described in [3] and [27], MPC is an optimisation-based control strategy that employs a base optimisation procedure (which isn't limited to a LP) to determine control inputs that yield the best-predicted behaviour of the system over a fixed prediction horizon. It operates iteratively in a receding horizon fashion. At each time interval, updated measurements and forecasts are used to predict the future behaviour of the system. Through MPC, we can proactively anticipate future events and changes, as our actions are grounded in future outlooks. Moreover, forecast uncertainties are mitigated through the receding horizon control method, meaning that only the predicted optimal actions corresponding to the current state are executed. For example, using the system forecasts that we input Figure 16 and the corresponding optimal actions given in Figure 17 we only implement the first action, before repeating the process from our new state.

Given the current information known about our system (this is later referred to as the "state", and is defined in Section 6.1.0.2), the controller finds the system inputs (the forecasts), which are the predictions for the PV generation, energy demands, and energy prices for each interval (for now we use the point forecasts)

$$[(\hat{g}_t, \hat{d}_t, \hat{p}^c_t)^t, \ldots, (\hat{g}_{t+H-1}, \hat{d}_{t+H-1}, \hat{p}^c_{t+H-1})^t], \tag{16}$$

and uses those to output the control variables (i.e. actions) that optimise the system's performance over the next H time steps. The control variables that correspond to actions for the current time step are then applied to the system, advancing it to a new state. At the new state, the process is them repeated with updated measurements and fresh forecasts.

The internal model that we utilise is 5, from Section 2. This means that we can plug all our our forecasts given by 16 into our model and find the optimal charge and discharge actions based on these point forecasts by pretending that they are the true measurements (i.e. these input variables are deterministic). The model is solved at each time step, with actions

executed only for the current interval.

Hence, we can iteratively generate forecasts for the next day, as we show in Section 3, to find the optimal actions for our system by solving our base optimisation model (Model 5) by treating these forecasts as the real measurements for the prices, PV generation, and demands, before then executing the outputted optimal action for the first interval before repeating the process once we get our new information at the end of that interval.

As we address in Section 2, a number of papers - including [28] and [31] - test their systems using perfect forecasts, making it difficult to assess how such a system would behave in the real-world when these forecasts are subject to errors.

It remains challenging to assess the real-world performance of each of these deterministic MPC approaches as they seem to explore different problems and often don't test their approaches using forecasts, but rather using hindsight. However, testing the model using real forecasts is critical to evaluating the performance of an optimisation framework, as often a LP inputted with forecasts will come up with a battery schedule that differs to the optimal schedule that an LP with the real measurements would output. In Figure 16 we see how our point forecasts can differ from the real measurements. As our deterministic MPC system assumes that these point forecasts are real measurements, we then see that the projected optimal actions outputted by the LP that uses these forecasts, differs from the true optimal actions, as we see in Figure 17.

Figure 16: Forecasts vs. Actuals for Three Key Variables: Energy Demand, Energy Prices, and PV generation

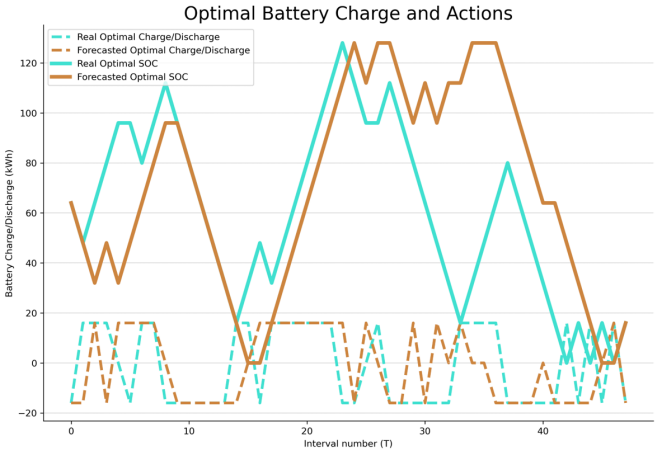


Figure 17: Optimal Battery Charge and Discharge for Forecasts vs. Actuals

We can see that due to the difference between the point forecasts and the real values of our variables, the subsequent optimal battery charging and discharging schedules differ, as we can see in Figure 17, where for the second interval the predicted optimal action is to discharge our battery, whereas the true optimal action is to charge it. While the action taken in Figure 17 in this case is still optimal, since we only carry out the first action before recalculating our forecasts (as is the case with MPC), it could easily lead to errors in other cases, which can lead to an inferior performance, as we see in Section 5.2.

These deviations in the optimal actions are caused by the uncertainty of future events, where the realisations of our random variables will likely differ from our point forecasts. To illustrate the importance of incorporating more than just a point estimate of the random variables that we are trying to model, we should consider the following hypothetical scenario.

Suppose that we want to optimise our BESS over two intervals (i.e. $H = 2$), with the values associated with the random variables for the first interval known, and given by $p_0^c = 10, d_0 = g_0 = 20$. Then, suppose that the initial battery state of charge is $B_{-1} = 5$, with charging and discharging efficiency of $e = 1$, a capacity of $c = 100$, and a power of $P = 20$ (units are as we define them for the given variables in Section 2). Let us also assume that we have the distributions of the random variables corresponding to our prices, demands, and PV generation in the first interval being given by $\mathbb{P}(X_1 = (1000, 20, 20)) = \mathbb{P}(X_1 = (1000, 30, 20)) = 0.5$. Given this data, the inputs that we will feed into our LP will be $((p_0^c, d_0, g_0), (p_1^c, d_1, g_1)) = ((10, 20, 20), (1000, \frac{30+20}{2}, 20))$,

which are the point forecasts, given that they correspond to the expected values of each random variable. Hence, if we use the point forecasts, then the optimal solution that minimises our costs is to use all of our generation in the first interval to offset our energy demands (leading to us not importing anything), and then to use our PV generation and battery to offset our demands in the second interval (resulting in us to importing anything), as we see in Equation 17. This leads to us choosing the actions $a_0 = 0, a_1 = -5$ (battery idle in first interval, battery discharges 5kWh in second interval) with our expected costs (falsely, given that our true variable values differ from our point forecasts) being

$$\mathbb{E}(Cost(a_0 = 0, a_1 = -5)) = p_0^c(d_0 - g_0) + \mathbb{E}(X_1^{p^c})(\mathbb{E}(X_1^d) - \mathbb{E}(X_1^g) - B_1) \tag{17}$$

$$= 10 \cdot (20 - 20) + 1000(25 - 20 - 5) = 0.$$

However, given that we know our distribution, our true expected costs using the above strategy are

$$\mathbb{E}(Cost(a_0 = 0, a_1 = -5)) = p_0^c(d_0 - g_0) + \frac{1000(20 - 20 - 5)^+ + 1000(30 - 20 - 5)}{2}$$

$$= \frac{0 + 5000}{2} = 2500.$$

Had we known this distribution then we could use 5kWh from the generation in the first interval, g_0, to charge our battery, leading to us choosing $a_0 = 5, a_1 = -10$, yielding

$$\mathbb{E}(Cost(a_0 = 5, a_1 = -10)) = p_0^c(d_0 + 5 - g_0) + \frac{1000(20 - 20 - 10)^+ + 1000(30 - 20 - 10)}{2}$$

$$= 10(25 - 20) = 50,$$

which is much better than the expected cost of 2500c above. We explore the mathematical reasoning behind this Equation 21 below.

This leads us to the conclusion that once we consider the inherent uncertainty of future events a more appropriate form for the objective function arises, one that addresses this uncertainty. In each interval, our goal is to minimise the cost of the current interval (whose variables are assumed to be known) in conjunction with the expected costs of future intervals. Similar to our notation in Section 2, we denote $\text{Cost}(s, a) = p^c \cdot [d - g + e \cdot a]^+$ (we do not allow export) as the cost associated with charging the battery by akWh (or discharging

$-a$kWh), given the complete information set s. Here, $s \in S$, where

$$S = \{(d, p^c, g) : d \in \mathcal{D}, p^c \in \mathcal{P}^c, g \in G\}, \tag{18}$$

with $\mathcal{D}$ being our set of possible energy demands, $\mathcal{P}^c$ our possible combined energy prices, and G our possible PV generations.

Thus, S encompasses all possible combinations of the energy prices, demand, and PV generation, as we address in Section 3 (which can be discretised by rounding), it also corresponds to the possible values that X_t can take (again, we address this in Section 3). We refer to S as our state space (referring to all of our possible states, and defined in Section 6.1.0.2), where each state only captures the information that affects the BESS costs for the time interval corresponding to the respective state (rather than capturing historical data, as we see in Section 6.1.0.9).

The objective that we strive to minimise when we utilise the point forecasts as inputs to the LP that we define in Section 2 is equivalent to

$$\min_{a_{t \in T} \in A} \left(\text{Cost}(S_0, a_0) + \sum_{t=1}^{H} \text{Cost}(\mathbb{E}[X_t], a_t) \right), \tag{19}$$

given that our point forecasts for interval t are the expected values of the random variables representing our prices, demands, and PV generation at interval t. However, given that our aim is to minimise the expected costs, our true objective is

$$\min_{a_{t \in T} \in A} \left(\text{Cost}(S_0, a_0) + \sum_{t=1}^{H} \mathbb{E}[\text{Cost}(X_t, a_t)] \right). \tag{20}$$

Hence, the optimal actions for Equation 19 and Equation 20 differ because

$$\mathbb{E}[\text{Cost}(X_t, a_t)] = \sum_{s \in S} [\mathbb{P}(X_t = s) \cdot \text{Cost}(s, a_t)] \neq \text{Cost}\left(\sum_{s \in S} [\mathbb{P}(X_t = s) \cdot s], a_t \right) = \text{Cost}(\mathbb{E}[X_t], a_t). \tag{21}$$

Here, A represents the feasible decision space, typically associated with battery charging or discharging (given that we have constraints for how much we can charge and discharge our

battery).

One of the reasons for this is that the cost function is non-linear, given that exports are not allowed. And, even if exports were allowed the cost function would still be non-linear, given that the price that a consumer receives for exporting energy is lower than the consumer pays for importing energy, with the typical export rate (also known as feed-in tariff) of 6.6c/kWh lower than import rates that range from 20c/kWh to 40c/kWh. And, even if the consumer is a participant in the central dispatch process of the NEM (meaning that they sell their energy at Wholesale Spot Market prices), then $\text{Cost}(s, a) = p(d - g + e \cdot a) + n[d - g + e \cdot a]^+$,with the network consumption charges adding non-linearity to the cost function.

A possible way of overcoming this limitation is to change the base optimisation model that our MPC uses to a model that takes into account the uncertainty of the future - the multi-stage LP does just that. First introduced by Dantzig, G.B., the pioneer of the simplex method, in [9], multi-stage LP models are structured into multiple stages, with only the inputs for the first stage known in advance, and subsequent inputs depending on data that remains unknown. This is designed to reflect real-life scenarios where information unfolds progressively and allows for modelling multi-stage decision problems with recourse, where only the first-stage decisions are initially implemented.

In a standard two-stage stochastic optimisation problem, as described in [9], the decision variables are split into two groups, with the variables $\boldsymbol{x}_d$ for $d \in D_1$ corresponding to the first-stage decisions, and $\boldsymbol{y}_d^t$ for $d \in D_2$, $1 \leq t \leq H$, corresponding to the decisions for the second stage, where $H \in \mathbb{Z}^+$ is the horizon length.

While the first-stage variables need to satisfy the deterministic inequalities $f_c^1(\boldsymbol{x}) \geq \boldsymbol{b}$, for some set linear function $f_c^1 : \mathbb{R}^{D_1} \to \mathbb{R}$, the second-stage constraints are more complex.

A random variable, $\boldsymbol{X}$ (similar to how we define $\boldsymbol{X}_i$ in Section 3), on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, defined as $\boldsymbol{X} : \Omega \to S^{H-1}$, is used to represent the second stage outcomes for some measurable space, S, representing all the possible realisations of our random variable inputs (as defined in Section 3).

The constraints are then given by the inequalities $f_{t,c}^2(\boldsymbol{x}, \boldsymbol{y}_t) \geq g_t(\boldsymbol{\xi})$ for each $\boldsymbol{\xi} \in S, 1 \leq t \leq$

$H-1$, where $f_c^2 : \mathbb{R}^{D_2} \to \mathbb{R}$ and $g_t : \Omega \to \mathbb{R}, 1 \leq t \leq H$ are specified functions, with $f_{t,c}^2$ linear. Thus, the second stage constraints depend on the random outcome of $\boldsymbol{X}$, as well as the decisions made in the first stage, $\boldsymbol{x}$.

The objective for the first stage is deterministic, and is to $\min_{\boldsymbol{x}} f_{t,o}^1(\boldsymbol{x})$, where f_o^1 is also a linear function. The second-stage objective is given to $\min_{\boldsymbol{x},\boldsymbol{y}_t} \sum_{t=1}^{H-1} f_{t,o}^2(\boldsymbol{x}, \boldsymbol{y}_t, \boldsymbol{X})$, which depends on both first-stage and second-stage decisions as well as random variable, $\boldsymbol{X}$. Here, $f_{t,o}^2$ is also a linear function. The variable $\boldsymbol{x}$ represents decision variables are made immediately, and thus, are chosen prior to the outcome of the random event $\boldsymbol{X}$, while the decision variable $\boldsymbol{y}$ is chosen after the outcome of the random event is known. The objective is to minimise the expected value of the sum of the objective functions, hence, it is given by

$$\min_{\boldsymbol{x},\boldsymbol{y}_t^{\boldsymbol{\xi}}} \left\{ f_o^1(\boldsymbol{x}) + \int_{\boldsymbol{\xi} \in S^{H-1}} \left(\sum_{1 \leq t \leq H-1} f_{t,o}^2(\boldsymbol{x}, \boldsymbol{y}_t^{\boldsymbol{\xi}}, \boldsymbol{\xi}) \right) \mathbf{d}\mathbb{P}(\boldsymbol{X} \leq \boldsymbol{\xi}) \right\}, \tag{22}$$

subject to:

$$f_c^2(\boldsymbol{x}, \boldsymbol{y}_t^{\boldsymbol{\xi}}) \geq g(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} \in S, 1 \leq t \leq H-1;$$

$$\boldsymbol{x} \in \mathbb{R}^{D_1}, \boldsymbol{y}_t^{\boldsymbol{\xi}} \in \mathbb{R}^{D_2} \quad \forall\, 1 \leq t \leq H-1, \boldsymbol{\xi} \in S.$$

However, given that this can lead to a huge (or, even infinite) number of constraints, we don't include all $\boldsymbol{\xi} \in S$. Rather, we usually sample $N \in \mathbb{N}$ randomly generated outcomes according to $\boldsymbol{X}$'s distribution, assign a probability of $\frac{1}{N}$ to each, and use that to approximate S. We address some techniques for random sampling in Section 5.1.

As outlined in [9] and in [27], this general approach can be extended to accommodate any number of stages. This can include systems such as BESS, where the random variable $\boldsymbol{X}$ is replaced by a collection of random variables $\{X_t, 1 \leq t \leq H-1\}$ (as we mention at the top of Section 3).

In reality, most multi-stage LP systems are restricted to the two-stage form that we explore, as the problem size grows exponentially with each additional stage ($\times N$ at each stage, hence, N^H total), potentially rendering it too computationally expensive to solve (intractable).

Stochastic MPC, which is a MPC method which utilises a multi-stage LP as the base model,

theoretically provides a more accurate estimate of the true expected cost as it involves sampling X_t a large number of times (as briefly mention above), thereby accounting for a variety of potential scenarios, this idea is further explored in [27].

The theoretically superior performance of the stochastic MPC compared to the deterministic MPC can be formally proven. We see that $\text{Cost}(X_t, a)$ is a random variable (as it is a function of another random variable, X_t). This random variable is defined over the same space as X_t, except that $\text{Cost}(X_t, a) : \Omega \to \mathbb{R}$. We assume that our sampling accurately follows the distribution of X_t, and then we denote the i^{th} sampled information as $\hat{X}_t^i$, out of a total of N samples (we explore techniques for sampling large quantities of these random variables in Section 5.1). Then, according to the law of large numbers,

$$\lim_{N\to\infty} \left(\frac{1}{N} \sum_{i=1}^{N} \text{Cost}(\hat{X}_t^i, a_t) \right) \to \mathbb{E}\left[\text{Cost}(X_t, a_t)\right]. \tag{23}$$

This implies that sampling different scenarios, as is done for multi-stage LPs, should theoretically provide a better estimate of the expected value of the cost function. Consequently, we can make decisions that minimise our expected cost, in contrast to the deterministic MPC's approach which takes the expected value of the random variable itself, potentially leading to different optimal actions.

However, the approach of generating a large number of scenarios becomes infeasible, as we can observe in Figure 18. This is because if we generate N possible scenarios for interval 1, then for each of those N scenarios we need to generate another N scenarios for stage 2, given that we would repeat this procedure at each future decision-making point, as we see in diagram b) of Figure 18. Therefore, we usually only carry out the procedure once, for interval 1, as we see in diagram a) of Figure 18. This means that our optimisation model does not truly reflect our H interval decision making problem with recourse. Mathematically, this results in our objective becoming:

$$\min_{a_{t\in T}\in A} \left(\text{Cost}(S_0, a_0) + \frac{1}{N} \sum_{i=1}^{N} \left(\text{Cost}(\hat{X}_1^i, a_1^i) + \sum_{t=2}^{H} \text{Cost}(\mathbb{E}\left[X_t | \hat{X}_1^i\right], a_t^i) \right) \right). \tag{24}$$

This formulation now serves as our base model for stochastic MPC method.

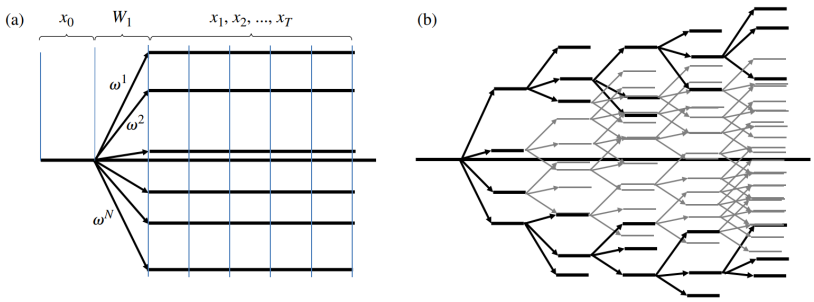


Figure 18: Illustration of (a) a two-stage scenario tree and (b) a multistage scenario tree. [27]

Naturally, this simplification implies that our new formulation may not necessarily guide us to select the optimal action. While the trade-off between optimality and computational tractability is explored in texts such as [27], assessing the extent to which we deviate from the optimal solution, and comparing this deviation with other methodologies, constitutes a critically important problem.

However, before we delve into that we need to first explore how we can randomly sample from the distribution of the random variables representing our energy demands, price, and PV generation in order to generate a series of potential scenarios to more accurately model our problem. We address this in a section separate from Section 3, because now we are not attempting to output point estimates for our random variables, but rather we are trying to randomly generate potential future trajectories (i.e. sample paths) for them.

5 Random Forecast Generation and Stochastic BESS Control

5.1 Generating Random Forecasts

Probabilistic forecasting is field of forecasting where the aim is not simply to create a model $f(\cdot)$ that takes some exogenous data $\boldsymbol{X}_i$ about some instance i in order to output $f(\boldsymbol{X}_i) = \hat{y}_i = \mathbb{E}(\boldsymbol{Y}_i)$, with $\boldsymbol{Y}_i$ being the value being forecasted. Rather, the aim is to return a series of values, such that $f(\boldsymbol{X}_i) = (\hat{y}_i^{p_j})_{j=1,\dots,n}$, where $\mathbb{P}(\boldsymbol{Y}_i \leq \hat{y}_i^{p_j}) = p_j$ (corresponding to the j^{th} quantile), corresponding to a set of potential outcomes for the random variable $\boldsymbol{Y}_i$. These values could then be used as samples of the random variable that we are trying to model in order to help us improve our BESS control, as we show in Section 4. However, we do not attempt to create this type of forecaster in our thesis due to the significant time that it would take to build forecasting models for each quantile being forecasted for. Rather, we attempt to simulate what these types of probabilistic forecasts could look like by randomly generating forecasts.

Before we dive into generating random forecasts, we first need to understand a few key concepts that are fundamental to our goal.

Residual The first of these is a residual. In the context of forecasting this refers to the difference between the forecasts and the actual observations. For example, if we generate a PV generation forecast $\hat{\boldsymbol{g}} = [\hat{g}_1, \dots, \hat{g}_{H-1}]$, with the real measurements being $\boldsymbol{g} = [g_1, \dots, g_{H-1}]$, then the residuals are defined as

$$\boldsymbol{r}^g = [r_1^g, \dots, r_{H-1}^g] = [g_1 - \hat{g}_1, \dots, g_{H-1} - \hat{g}_{H-1}]. \tag{25}$$

Hence, generating a series of random forecasts is equivalent to generating a series of residual sample paths, $(\boldsymbol{r}_1^g, \dots, \boldsymbol{r}_N^g)$, and then creating a series of random forecasts with $\hat{\boldsymbol{g}}_i = \boldsymbol{g} - \boldsymbol{r}_i^g$, $i = 1, \dots, N$. We refer to the random variables representing our residuals as $R_t = (R_t^{p^c}, R_t^d, R_t^g)$ for the forecasts that we generate for $X_t = (X_t^{p^c}, X_t^d, X_t^g)$.

5.1.0.1 Empirical Cumulative Density Function In order to better understand, model and sample from our residuals $(R_t^{p^c}, R_t^d, R_t^g)$ (which can allow us to generate simulated forecasts) it is important to find their distribution. A method of achieving that is through constructing an empirical cumulative density function (ECDF) for them, given that we don't have an explicit formula for their distribution.

(Definition adapted from Kroese, D.P, *et al.* [15]) The ECDF is defined as the function

$$F_N(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{x_i \leq x\} = \frac{|\{i : x_i \leq x\}|}{N}, x \in \mathbb{R}, \tag{26}$$

for given data $x_1, \ldots, x_N$.
The empirical CDF is a non-decreasing step function that jumps up by an amount of $1/N$ at each of the data points $\{x_i\}$ ($\{\mathbf{x}_i\}$ for the multivariate case). Note that F_N is right-continuous and bounded between 0 and 1.

Using this approach, the combined ECDFs (combined for all intervals being forecasted for) for all of our forecasting residuals are shown below in Figure 19. However, we can then extend this approach to create a unique ECDF for each interval.

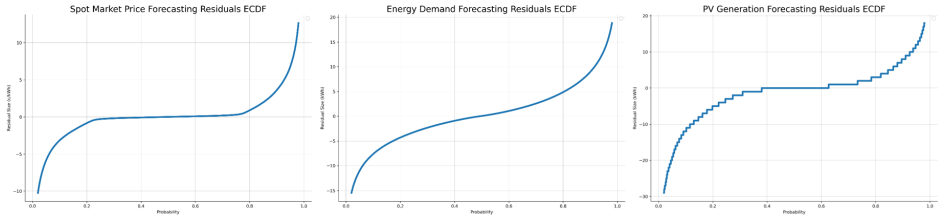


Figure 19: Left: NEM Energy Spot Prices; Centre: Energy Consumer; Right: 100kWh Solar Panel Site.
(Note: The PV Generation ECDF appears less smooth given that the measurements and forecasts are integers.)

5.1.0.2 Inverse Transform Sampling Once we have a method of finding a CDF (or ECDF) for our distribution, we can examine methods of sampling from the CDF, so that we can randomly generate forecasting residuals. Inverse transform sampling is a useful technique for sampling based on the cumulative density function CDF.

(Definition from Kroese, D.P, *et al.* [15]) Let R be a random variable with CDF F_R. Since F_R is a non-decreasing function, the inverse function F_R^{-1} may be defined as

$$F_R^{-1}(y) = \inf\{x : F_R(x) \geq y,\ 0 \leq y \leq 1\}. \tag{27}$$

Let $u \sim U(0,1)$. The CDF of the inverse transform $F_R^{-1}(u)$ is given by

$$\mathbb{P}(F_R^{-1}(u) \leq x) = \mathbb{P}(u \leq F_R(x)) = F_R(x). \tag{28}$$

Thus, to generate a random variable X with CDF F_R, simply generate $u \sim U(0,1)$ and set

$$X = F_R^{-1}(u). \tag{29}$$

We should note that this specific sampling method is only valid for sampling from univariate CDFs. Hence, we need to examine alternative/additional techniques in order to sample our multivariate random vectors. We should also note that pseudo-random generation of u is possible using existing pseudo-random number generation algorithms (some of which are described in [15]).

By repeatedly sampling our random variables R_t we can generate a set of $N \in \mathbb{N}$ values from its distribution $(\boldsymbol{r}_t^1, \ldots, \boldsymbol{r}_t^N)$ from which we can then get $(y_t^1 + r_t^1, \ldots, (y_t^N + r_t^N)$, which is a simulation of what our probabilistic forecasts could look like. This can then allows us to test our BESS control mechanism using probabilistic forecasts. However, it should be noted that through sampling the forecast for each interval independently, we assume that our residuals are independent (i.e. $\mathbb{P}(R_{t+1} = r_{t+1} | R_t = r_t) = \mathbb{P}(R_{t+1} = r_{t+1})$), given that we are required to sample a sample path of residuals (i.e. $(\boldsymbol{r}_t^j, \ldots, \boldsymbol{r}_{t+H-1}^j)$ for sample path number j). However, as we see in Figures 20 and 21, that assumption is false. Some stochastic MPC papers explore the generation of random paths for the forecasts in order to generate a series of scenarios that can be used for the optimal control of their system. The

paper [39] considers key (non-deterministic) inputs necessary for distributed energy systems as forecasts. It utilises the Monte Carlo methods to simulate how data could deviate from the forecasts and subsequently employs a two-stage stochastic LP to solve it, a format similar to what we utilise in Section 4. The results are then compared to those achieved through a deterministic program of the same type (where forecasts are known in advance). It shows that the costs achieved with the optimal deterministic program are only 1.28% better than those achieved with the stochastic program, meaning that the stochastic program achieves almost optimal performance.

A significant limitation, however, is the assumption of independence among forecast residuals for each interval in a forecasted block, which does not accurately represent reality where these residuals are correlated due to common variables. For example, the demand forecasts that we generate using the methodology that we elaborate on in Section 3 yield the correlations that are shown below in Figure 20, between the distributions of the residual errors for residuals within 48-interval forecasting block. Meanwhile, the PV generation forecasts, which are done by a Solcast, are also correlated according to the heatmap in Figure 21.

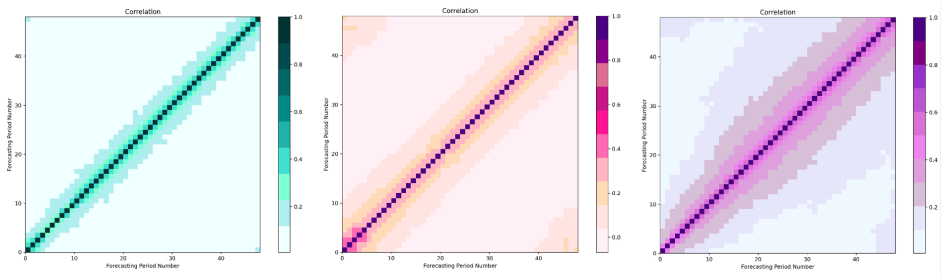


Figure 20: Heatmap of Correlations for inter-interval demand forecasting residuals, using data for the three consumers tested in our thesis.

Hence, the simulation approach in [39] appears to have outputs that significantly differ from true forecasting residuals, potentially leading to inflated test results and suboptimal system control. This discrepancy might explain why their stochastic MPC system appears to perform exceptionally well. However, their methodology could potentially complement a

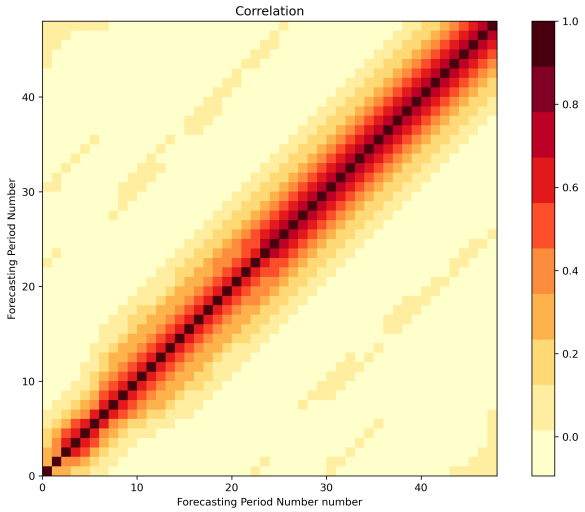


Figure 21: Heatmap of Correlations for inter-interval PV generation forecasting residuals, using combined data for the sites tested in our thesis.

more advanced Monte Carlo simulation approach, particularly one incorporating multivariate probabilistic forecasts. It's important, however, to note that this paper primarily focuses on long-term planning, resulting in a system optimisation context different from that of our thesis.

Another method is the Analog Ensemble method (AnEn) [2, 14]. AnEn transforms deterministic forecasts into probabilistic ones by comparing them with measured values at the same time as the most similar past forecasts. Formally, as defined in [2], if we have a historical point forecast F_t at time t, we aim to generate n potential outcomes that closely approximate actual measurements. To achieve this, we identify the n most 'similar' historical forecasts (we define similarity below), denoted as A_t^k for $k = 1, \ldots, n$. These n distinct outcomes serve as sample outcomes for our stochastic MPC system.
The similarity is determined by the distance metric

$$||F_t, A_t^k|| = \sum_{v=1}^{N_v} \frac{w_i}{\sigma_{f_v}} \sqrt{\sum_{j=-\tilde{t}}^{\tilde{t}} (F_{v,t+j} - A_{v,t+j}^k)^2}, \tag{30}$$

where $F_{v,t}$, and $A_{v,t}^k$, correspond to the current, and historical, value of exogenous variable $v \in \{1, \ldots, N_v\}$, with w_v being its weight (or determined importance), with σ_{f_v} being the standard deviation of past forecasts of the given variable at the respective time period ($[t - \tilde{t}, t + \tilde{t}]$), with $\tilde{t}$ being half the length of the time-window over which the metric is

computed. Thus, a shorter distance indicates a higher similarity for the historical forecast.

Some things to note are that Equation 30 in [2], $F_{v,t+j}$ is instead written as $F_{v,t-j}$, which appears to be a typo, given that its time then would not correspond to $A^k_{v,t+j}$'s time, and the paper also, somewhat arbitrarily, sets $n = 20$.

In [2], it is stated that the premise of this approach is based on the assumption that if similar past forecasts are found, their errors will likely be similar to the errors of the current forecast, which can be inferred from theirs. And, while this method appears impressive, given that we are looking to sample vectors of 48 intervals this approach needs to be expanded as it samples each interval independently. Hence, we slightly adapt this method in order to incorporate entire vectors.

We adapt Equation 30 by changing the time window such that

$$||F_t, A^k_t|| = \sum_{v=1}^{N_v} \frac{w_v}{\sigma_{f_v}} \sqrt{\sum_{j=0}^{47} (F_{v,t+j} - A^k_{v,t+j})^2} \tag{31}$$

and then use $(A^k_t, A^k_{t+1}, \ldots, A^k_{t+47})$ as a potential sample trajectory, with the weight that this sample path takes in Equation 22 of the stochastic MPC given by

$$\mathbb{P}(k) = \frac{||F_t, A^k_t||^{-1}}{\sum_{l=1}^{n} ||F_t, A^l_t||^{-1}}, \tag{32}$$

given that it signifies how 'close' it is to the current interval, relative to the other potential sample paths (we take the inverse because a shorter distance indicates higher similarity).

In the approach that we choose to utilise in our paper, we expand on the AnEn approach that is detailed in [2, 14], as we detail here.

We utilise the forecasts that we generate using the approaches detailed in Section 3 in order to create a dataset of overlapping vectors of the form $([\hat{p}^c_t, \hat{d}_t, \hat{g}_t], [p^c_t, d_t, g_t]), t \in \{i, \ldots, i + H - 1\}$ where i is an interval in time for which we have data. This is similar to the way in which we store our PV generation data in Section 3.1. Except, now we have energy demands and prices data as well.

Next, in order to find the residuals of our forecasts, we convert our data above into vectors of the form $(r_t^{p^c}, r_t^d, r_t^g) = (p_t^c - \hat{p}_t^c, d_t - \hat{d}_t, g_t - \hat{g}_t), t \in \{i, \ldots, i+H-1\}$, corresponding to the residuals for each H interval forecasting window and, thus, capturing the structure present within each such set of forecasts (which the [39] does not). As our residuals are still vectors of size H (for each of our three variables), it is very challenging to find their ECDFs (especially since we have a limited amount of data), which we want in order to randomly generate realistic forecasting residual paths. Hence, we seek to convert our H dimensional vectors of residuals into real numbers (1-dimensional) for which we can create ECDFs through the method that we detail in Paragraph 5.1.0.1. We achieve this through summing all of the values in the residual path, thus getting

$$(r_{i_h}^{p^c}, r_{i_h}^d, r_{i_h}^g) = \sum_{t=i_h,\ldots,i_h+H-1} \left(r_t^{p^c}, r_t^d, r_t^g\right). \tag{33}$$

Here h is the hour of the day ($h \in \{0, \ldots, 23\}$, with 0 being 12AM, 23 being 11PM), which we know based on i, despite us not explicitly mentioning it earlier for convenience. Next, we use our approach from Paragraph 5.1.0.1 to create a set of distinct ECDFs $(F_h^{p^c}, F_h^d, F_h^g), h \in \{0, \ldots, 47\}$, where ECDFs $(F_h^{p^c}, F_h^d, F_h^g)$ are made using the datasets $(\{r_{i_h}^{p^c} : \forall i_h \in \mathcal{H}\}, \{r_{i_h}^d : \forall i_h \in \mathcal{H}\}, \{r_{i_h}^g : \forall i_h \in \mathcal{H}\})$, corresponding to the same hourly intervals of each day.

Then we incorporate the AnEn methodology through Equation 31 to create a weighted ECDF that does not simply assign each residual path in our ECDF equal probability of being randomly sampled, but rather attempts to find how similar that path is to our current real measurements and assigns it a probability based on that similarity (the more similar the historical H interval data is, the more likely it is to get chosen). Specifically, we want to modify our ECDF for some given interval $i \in \mathcal{H}$, an interval for which we have real historical measurements given by $(p_t^c, d_t, g_t), t \in \{i, \ldots, i + H - 1\}$ (corresponding to hour h) to assign each entry point within it a weight. Therefore, we look at the real measurements that correspond to each entry in each of the three ECDFs $(F_h^{p^c}, F_h^d, F_h^g)$. Then, for each set of these real measurements $(p_s^c, d_s, g_s), s \in \{j, \ldots, s + H - 1\}$, we find the similarity metric by applying Equation 31. The similarity that we get for each of the three variables for historical

interval j when we are attempting to simulate sample paths for interval i is given by

$$\left(\left(\sum_{t=0,\ldots,47} (p^c_{i+t} - p^c_{j+t})^2 \right)^{-1/2}, \left(\sum_{t=0,\ldots,47} (d_{i+t} - d_{j+t})^2 \right)^{-1/2}, \left(\sum_{t=0,\ldots,47} (g_{i+t} - g_{j+t})^2 \right)^{-1/2} \right). \quad (34)$$

We then normalise the similarities for each entry for each variable by applying Equation 32 (for each entry we divide the similarity weight by the total sum of similarity weights for entries). Now, we have a weighted ECDF for the specific set of measurements for which we want to simulate forecasts.

Then, to simulate a set of forecasts based on the real measurements $(p^c_t, d_t, g_t), t \in \{i, \ldots, i + H - 1\}$ that we can feed into our multi-stage LP for some interval i for each we have real measurements, we find the hour of the day (h) that i corresponds to. Then, we use the inverse transform sampling procedure that we detail in Paragraph 5.1.0.2 to (independently) randomly sample three historical entries, (j_{p^c}, j_d, j_g), from our weighted ECDFs $(F_h^{p^c}, F_h^d, F_h^g)$. Then, we find the corresponding vectors of residuals $\boldsymbol{r}_n^{p^c} = (r_t^{p^c})_{t\in\{j_{p^c},\ldots,j_{p^c}+H-1\}}$, $\boldsymbol{r}_n^d = (r_t^d)_{t\in\{j_d,\ldots,j_d+H-1\}}$, and $\boldsymbol{r}_n^g = (r_t^g)_{t\in\{j_g,\ldots,j_g+H-1\}}$, with the n signifying that this is the n^{th} of $N \in \mathbb{N}$ sets of randomly generated forecasts. We repeat this process N times to create a set of random forecasts which we can then use as inputs to our stochastic MPC, as we show in Section 22. While our the forecasting residuals for the PV generation, energy demands, and energy prices are likely correlated to some extent (e.g. excess PV generation could lead to excess energy supply, leading to decreased prices), we do not attempt to model this correlation. Rather, for simplicity, we sample each ECDF independently.

The reasoning behind our use of this specific approach is threefold. Firstly, it enables us to simply and easily generate random foresting paths. Secondly, we use the sum of the residuals because how much we overestimate or underestimate our variables over our control window translates to how our BESS costs will change. We want to create an ECDF that incorporates the information regarding whether our forecasting vector under-estimates or over-estimates, given that an under-estimation and an over-estimation will likely affect our decision making in vastly different ways, as we show in Equation 17. Thirdly, the sample paths that we generate capture the structure of our forecasting residuals paths, as we are just randomly sampling our historical forecasting residual paths, while also factoring in similarity in order to ensure that our forecasting residual paths are reflective of those that could be generated in the real world for the specific set of H measurements that we are seeking to generate

random forecasting residuals from. Hence, we test our optimisation approach using data that features uncertainty similar to that which our optimisation framework would encounter in the real world.

To demonstrate how our randomly generated forecasts look, for each of our three variables we plot a 48-interval window of real measurements, and the corresponding historical forecasts, we well as our randomly generate sample paths. We plot three randomly generated sample paths to reduce clutter. We see this plot below in Figure 22.

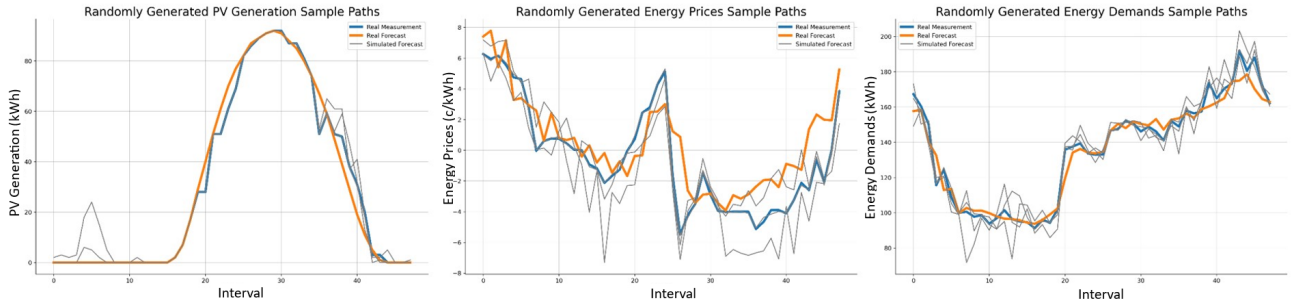


Figure 22: Plots of real measurements and historical forecasts, along with randomly generate forecasts, for PV generation, energy prices, and demands.

5.2 Stochastic MPC BESS Control

Now that we have a method of generating random sample paths for our variables, we can create our stochastic MPC, as we describe it in Section 4. In Figure 23 below, we visualise how our stochastic MPC performs based on the random forecasts that we generate. First, we visualise 3 randomly generated forecasts, based on the real data, that we also plot. We limit the randomly generated sample paths to 3 to reduce clutter. Based on the randomly generated forecasts, we then see in Figure 23 how the optimal battery actions differ based on the different forecasted sample paths. However, as we clearly see, the first action is forced to be the same regardless of the trajectory forecasted. As we explain in Section 4, this helps us ensure that we are choosing the overall best action, taking into account a range of different scenarios, rather than just the one scenario, as is the case in the deterministic MPC. The reason that we force only the first action to be identical is because, as we mention in Section 4, we recalibrate the system every interval (once we get new information).

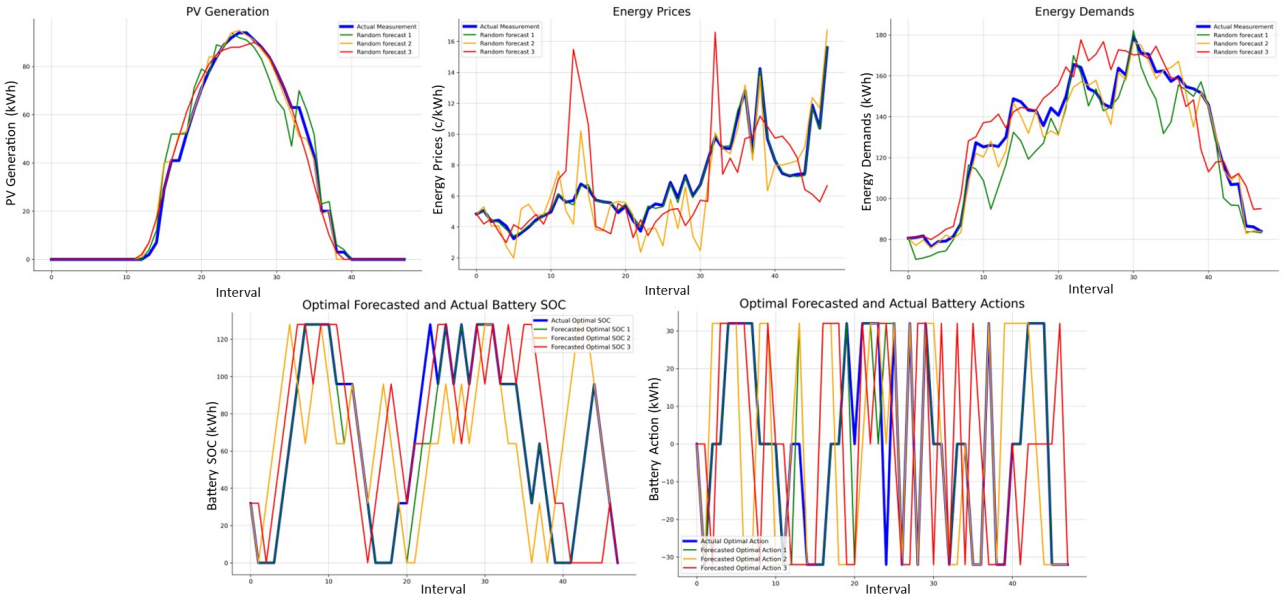


Figure 23: A figure demonstrating the optimal charging and discharging of our battery based on different simulated trajectories.

A natural question to ask now is whether we find that stochastic MPC control outperforms deterministic MPC. Based on our testing, we see that it does, with the stochastic MPC saving over 1% on energy costs when compared to the deterministic MPC. While the percentage gap in this case seems very small, it is important to be able to evaluate it in the context of the data that it is being tested on, as we show in Section 7, where we explore the question of performance evaluation further.

Now that we have examined how the stochastic BESS MPC works, we can move onto our third optimal control framework, which we explore in Section 6 below.

6 Stochastic Dynamic Programming

So far in our thesis we have examined MPC-based methods for BESS control optimisation. While both of the formulations we explored above aim to determine the optimal action through a lookahead model, the model predictive control (MDP) formulation alters this approach by eliminating the need to find forecasts, both regular and conditional. Instead of each state $s \in S$ containing the information needed to minimise costs for the corresponding interval, s now encompasses all the necessary information for us to make the best decision by considering all future expected costs. MDP is the modelling framework that is utilised by stochastic dynamic programming (SDP).

However, to understand how MDPs work we need to dive into the underlying theory.

6.1 MDP's Technical Underpinnings

In this section, we delve into the foundational concepts and fundamental properties of discrete-time Markov chains, elucidating their role as the bedrock of MDPs and dynamic optimal control.
In this section, where we address discrete-time Markov chains, we draw extensively from the mathematical treatises by Norris, J.R. [23], and Kroese, D.P, *et al.* [15].

6.1.0.1 Stochastic Process (Definition adapted from "Handbook of Monte Carlo Methods" by Kroese, D.P, *et al.* [15]) A stochastic process is a collection of random variables $\{X_i, i \in \mathbb{Z}^+\}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. The set S of possible values for X_i is called the state space of the process. Hence, a stochastic process is often thought of as a random variable evolving through time, with X_i representing the state of the process at time i, defined as $X_i : \Omega \to S, \ \forall\, i \in \mathbb{Z}^+$.

6.1.0.2 States and State Space In our problem domain, we treat the state space as a finite discrete set. Applied to the optimal control problem, the state space encompasses all possible system conditions. Each state within the state space captures a unique situation, encapsulating all relevant information for optimal decision-making. In the context of optimal BESS control, this information includes variables such as time of day, current temperature, PV

generation, demand, battery state of charge, electricity prices, forecasts, and other pertinent factors affecting the energy system.

The state variables, which we address in Section 6.1.0.9, collectively define each state, and while this might initially suggest a vast state space, it's common practice to truncate the state space to ensure computational feasibility, a step we also undertake in this thesis.

6.1.0.3 Transitions As we mention in Section 6.1.0.2, the system employed in our problem operates in discrete time, as elaborated in [23].

Applying this concept to our problem, it means transitioning from one state characterised by specific electricity prices, energy demands, and other variables to another state where some of these variables may assume different values. The mechanics of these state transitions are fundamental to what is known as the Markov property.

6.1.0.4 Markov Processes and Markov Property The Markov property, often referred to as the 'memoryless' property, implies that the past is 'forgotten', and only the present state matters.

(Definition adapted from "Handbook of Monte Carlo Methods" by Kroese, D.P, *et al.* [15]) Using our previously defined X_i, we say that the stochastic process X_i is a Markov process if for every $i, j \in \mathbb{Z}^+$, and $A \in S$ it satisfies the Markov property:

$$\mathbb{P}(X_{i+j} \in A | \mathcal{H}_i) = \mathbb{P}(X_{i+j} \in A | X_i), \tag{35}$$

where $\mathcal{H}_i$ is the history of the process up until time i.

The Markov property can be expressed as

$$(X_{i+j} | X_u, u \leq i) \sim (X_{i+j} | X_i) \; \forall \, i, j \in \mathbb{Z}^+, \tag{36}$$

which emphasises that the conditional future distributions of the sample path given the entire sample path history are the same as those given only the present state. In other words, for a Markov process, the conditional distribution of the "future" variable X_{i+j} given the entire

past of the process $\{X_u, u \leq i\}$ is the same as the conditional distribution of X_{i+j} given only the "present" X_i.

This means that any past information, for which conditional probabilities might have otherwise be needed, is uniquely captured by the state and its transition probability.

6.1.0.5 Initial Distribution Our system's initial state, given by X_0, depends on an initial distribution across all states. This is denoted as λ, where $\mathbb{P}(X_0 = s) = \lambda_s, s \in S$. In many cases, the λ distribution is degenerate, meaning that for some $s \in S, \lambda_s = 1$.

6.1.0.6 Transition Probabilities After our initial state, our transition probabilities remain independent of time, meaning that our transition probabilities remain stationary. This yields the crucial relationship

$$\mathbb{P}(X_{i+1} = l | X_i = k) = \mathbb{P}(X_1 = l | X_0 = k), \quad k, l \in S. \tag{37}$$

As elucidated in [23], these probabilities are conveniently represented by a matrix P, where $P_{kl} = \mathbb{P}(X_{i+1} = l | X_i = k) = \mathbb{P}(X_1 = l | X_0 = k)$. This stationarity also simplifies the calculation of probabilities for future states, such as

$$\mathbb{P}(X_{i+j} = l | X_i = k) = P_{kl}^{(j)} = [P^j]_{kl}, \quad k, l \in S,\ i, j \in \mathbb{Z}^+. \tag{38}$$

Translating this concept to our problem domain, transition probabilities depend on various factors, including the likelihood of high prices in the next interval, future demand, and deterministic elements like our battery state of charge (which is solely influenced by our charging or discharging decisions) and the time of day, these factors encompass what is known as state variables, which we discuss below.

6.1.0.7 Markov Chain (Definition from "Handbook of Monte Carlo Methods" Kroese, D.P, *et al.* [15]) A Markov process with a countable index set is called a Markov chain.

Therefore, by combining the the previously defined Markov process, $X_i, i \in \mathbb{Z}^+$, along with our initial distribution and transition probabilities, we obtain Markov(λ, P), which uniquely defines our Markov chain, as is presented in [23].

6.1.0.8 MDP With the foundational concepts of a Markov chain now established, we can transition from theoretical components to practical applications that are directly relevant to our problem. This leads us to MPDs, which expand upon the Markov chain properties we defined earlier and introduce additional concepts essential for constructing an optimal decision-making framework. In this section, we draw insights from article by Powell, W.B. [27].

It's crucial to bear in mind that we must now consider our system within the context of optimal control. This means that we need to utilise the information provided by the uniqueness of each state that our Markovian BESS system can be in (and the states associated costs), along with the transition probabilities, in order to make decisions that minimise our costs.

6.1.0.9 State Variable In Paragraph 6.1.0.6 we have addressed the transition probabilities between states, in this paragraph we address the set of variables that uniquely define each state - the state variables. Given that, as we explain in Paragraph 6.1.0.2, each state in our state space captures a unique situation, the state variable is the function that captures the information representing this unique situation.

More formally, as elaborated in [27], a state variable is a function that minimally represents the entire history that is necessary and sufficient for making optimal decisions. While this does not differ from what is said by Kroese, D.P, *et al.* in [15], the emphasis is now on optimally finding, choosing, and representing these state variables, given that it is usually intractable to include every state variable as it will likely make the state space too large.

In the system that we define in our thesis (based on the system defined in [35]), our state space is defined by

$$S = \{(b, d, p^c, t, g) : b \in B, d \in \mathcal{D}, \in \mathcal{P}^c, t \in T, g \in G\}, \tag{39}$$

where $T = \{1, \ldots, 48\}$ gives the time of day, discretised into 48 30-minute intervals, $\mathcal{P}^c$ is the set of our possible energy prices in c/kWh (stochastic wholesale spot market price, combined with our deterministic network charges), while $\mathcal{D} \subset \mathbb{R}^+$ is the set of our possible energy demands in any 30-minute interval kWh, G is the set of our possible PV generations in kWh for any 30-minute interval, and B is our set of possible states of charge for our battery, with $b \in [0, c] \forall b \in B$. Typically, we may discretise each of these sets into only integers (or

multiples of 0.5, or 5) in order to reduce the size of our state space. However, we should note that there are many possible state space representations for this very problem, and the one that we define here is only one of those.

6.1.0.10 Action Set The core objective of any optimal control problem is to make decisions that maximise desired outcomes. This process involves selecting the most advantageous actions given the specific system conditions. While the determination of the "best action" is discussed later, it is crucial to first define the complete set of actions that can potentially influence the system's state. This is referred to as the action set, denoted as A_s, $s \in S$, as precisely defined in [27], with individual actions represented as $a \in A_s$. Typically, the action set is discrete and may vary depending on the current state, s.

In our context, the action set corresponds to determining how much energy we opt to charge or discharge the battery in any given situation, thereby directly influencing our decisions on buying and selling energy. Specifically, for some state $s = (b, d, p^c, t, g) \in S$, our action set is $A_s = \{a : b \leq a \leq c - b, z \cdot a \in \mathbb{Z}\}$, where z is our discretising factor (i.e. if $z = 1$, our action set is integer).

6.1.0.11 Incorporating Actions into our Transition Probabilities With a well-defined action set in our system, we incorporate actions into the transition probabilities, so that our transition probabilities depend on the action that we take.

Instead of the previous notation of P_{kl} representing transitions between states $k \in S$ and $l \in S$, we now define this probability function as $P(l|k, a)$, with $\sum_{s \in S} P(s|k, a) = 1$. This notation signifies the probability of transitioning from state $k \in S$ to state $l \in S$, contingent upon the selection of action $a \in A_k$.

6.1.0.12 Immediate Costs As discussed in [27], the cost (or reward) function for our current interval, denoted as $\text{Cost}(s, a)$ ($r(s, a)$), hinges on our system's current state $s \in S$ and the chosen action $a \in A_s$.

In the context of our energy system, the cost incurred within the current interval is mainly dependent on our current decisions regarding battery charging or discharging (a), as well as factors like energy prices, PV generation, and energy demand specific to the current state (s). The reason that our current interval cost is not only dependent on our current decisions is

due to our battery's state of charge, which limits our action set, A_s (as defined in Paragraph 6.1.0.10), with it depending on our past charging and discharging actions.

It's important to note that minimising $\text{Cost}(s, a)$ by maximising battery discharge may come at the expense of energy reserves for future use. This consideration plays a crucial role in determining the value assigned to states in our system.

6.1.0.13 Discount Factor To quantify how we weigh the importance of future costs incurred by our system, which may result from our actions, we introduce $\gamma \in (0, 1)$. This concept is elaborated upon further in [27]. The parameter, γ, serves as a factor for discounting the system's future costs, with the costs associated with a state i intervals away being discounted by a factor of γ^i.

The rationale behind employing this factor lies in the value reduction over time (i.e. $\$x$ now is worth more than $\$x$ a year from now). As stated in [35], in the context of BESS, this is to emphasise early decisions and costs, in order to emulate the effect of reduced battery efficiency over time. Furthermore, our problem has an infinite horizon - given that we seek to minimise our battery's costs over its lifetime, rather than over a fixed interval - however, we address this issue in Paragraph 6.1.0.15.

6.1.0.14 State Values and the Bellman Equation By amalgamating the costs, actions, and the discount factor, we can define the state value, denoted as $V(s)$, for $s \in S$.

Formally, as defined in [27], $V(s)$ represents the value (or total future cost) of being in state $s \in S$ while selecting the optimal set of actions to minimise the expected system costs.

The state value is determined through the Bellman Equation, extensively discussed in Section 6.1.0.15. In formal terms, the Bellman equation for the value of a given state, s, is

$$V(s) = \min_{a \in A_s} \left(\text{Cost}(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \right). \tag{40}$$

Determining the values for each state necessitates an optimisation approach explored in greater detail in Section 6.1.0.15.

6.1.0.15 Solving MDPs The content of this section is heavily influenced by the seminal work of Bellman, R. [5]. This groundbreaking paper not only introduced the Bellman equation but also presented a method for solving it. Since [5] was published, a number of new methods have been released for solving MDPs, including the Q-Learning method, which we describe in Paragraph 6.1.0.16

Solving a MDP involves repeatedly evaluating a form of Equation 40. We also note that although the state s does capture our time of day (as we show in Paragraph 6.1.0.9), we mention it explicitly for convenience when defining our problem, as t is the modulating state (i.e. we can also mention $P(s'|s,a)$ as $P(s'|s,a,t)$, for time t), hence, we write Equation 40 as

$$V_t(s) = \min_{a \in A_s}(\text{Cost}(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_{t+1}(s')). \tag{41}$$

Given that our notation includes the time of day, t, yet our problem features an infinite horizon, we explicitly mention our transition conditions for states at time $t = H$ for convenience, where

$$V_H(s) = \min_{a \in A_s}(\text{Cost}(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_0(s')). \tag{42}$$

One of the first developed methods for solving MDPs is the Value Iteration Algorithm (VIA), which is a direct implementation of Equation 40, as originally formulated by Bellman, R. in [5]. It works by iteratively updating the value of each state, $s \in S$, to determine the optimal action at that state. Starting with arbitrary values for all states, at each iteration the value of $V_t(s)$ is updated in accordance with the Bellman equation, using the previously calculated values for $V_{t+1}(s^{'})$.
The algorithm terminates when $|\text{change}(V_t(s))| \leq \epsilon$ for all $s \in S$, where $\epsilon > 0$ is a predefined threshold. Smaller values of ϵ increase precision but extend runtime. The algorithm is guaranteed to converge as our number of iterations goes to infinity, with $\text{change}(V_t s) \rightarrow 0$ for all $s \in S$, under certain conditions, as formally proven in [5].

The Policy Iteration Algorithm (PIA) offers another popular approach for solving MDPs. As mentioned in [26], unlike VIA, PIA works by iteratively updating the policy (or action)

for each state by setting

$$\pi_t(s) = \arg\min_{a \in A_s} (\text{Cost}(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V_{t+1}(s')). \tag{43}$$

It begins with the initialisation of arbitrary values for each state, as well as an arbitrary policy vector π. Then, it uses Equation 43 to iteratively update both the optimal value and the corresponding policy. However, at the end of each iteration, if π remains unchanged, the algorithm terminates. This is because the policy often becomes optimal long before the values of each state converge to their correct values.

6.1.0.16 Q-Learning Q-Learning is a method for solving MDPs that was first published by Watkins, C. J. C. H. [38]. A key difference between the Q-Learning approach and the VIA and PIA approaches is that Q-Learning model-free approach.

Model-free approaches learn through experience, that is, by performing an action a at state s and observing the outcome. Meanwhile, model-based approaches makes predictions about the consequences of its actions through building a model that contains all actions, rewards, states, and transitions associated with each. Model-free approaches may sometimes lack the foresight of model-based approaches, however, they are much faster to train and build.

The algorithm for the Q-Learning approach, based on its formulation in [38] is given as the following:

Algorithm 1 Q-Learning Algorithm

Inputs: $(S, A, r, \alpha, \gamma, \epsilon, N)$
Initialise Q-table with zeros: $Q(s, a) = 0 \; \forall s \in S, a \in A$
Initialise the current state, $s \in S$
for iteration = $1 : N$ **do**
    Reset the environment to the initial state, s
    $Done \rightarrow$ **False**
    while Not $Done$ **do**
        Comment: Choose an action using an ϵ-greedy policy
        if Random $U(0, 1) < \epsilon$ **then**
            Randomly select an action, $a \in A$
        else
            Select 'best' action: $a = \arg\max_a Q(s, a)$
        end if
        Execute action a in the state s
        Observe the resulting state (s') and reward (r)
        Update Q-value for the current state-action pair:
        $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
        Set the current state to the resulting state: $s \rightarrow s'$
        Update the exploration rate (e.g. $\epsilon = \alpha \cdot \epsilon, \alpha \in (0, 1)$)
        if s is a final or terminal state **then**
            Set $Done \rightarrow$ **True**
        end if
    end while
end for

There are multiple methods of deciding when to utilise exploration and randomly choose an action, or when to use exploitation and choose the best action.

Since the development of the original Q-Learning algorithm some notable advances have been made. These include Deep Q-Learning, which combines the Q-Learning with another machine-learning algorithm to improve its performance, however, this is beyond the scope of this thesis.

6.1.0.17 Concluding Remarks By consolidating the insights gained from this section, we are now equipped to formulate and solve a MDP for the BESS optimal control problem. Nonetheless, as discussed in Section 6.1.0.2, it is infeasible to incorporate all available information about our system into the state, as doing so would result in an impractically vast state space for computation.

Thus, the challenge in solving such a system lies in finding a delicate balance between the size of the state space, ensuring computability of the optimal policy, and striving for optimal system performance. One approach to address this challenge is explored in Section 6.4.

6.2 Incorporating the MDP framework in BESS control

By utilising the MDP-based approach our objective function changes from the MPC objective given by Equation 20 as

$$\min_{a_{t\in T}\in A}\left(\text{Cost}(S_0, a_0) + \sum_{t=1}^{H}\mathbb{E}\left[\text{Cost}(X_t, a_t)\right]\right) \tag{44}$$

to

$$\min_{a\in A}\left(\text{Cost}(s, a) + \gamma\sum_{s'\in S}P(s'|s, a)\cdot V_{s'}\right). \tag{45}$$

While we can determine the value of each state using the algorithms discussed in Section 6.1.0.15, our challenge lies in deciding how to represent our states and state space.

Once we have our state and state space representations, we can compute the transition probabilities through iterating over our historical data, $x_t, t \in \mathcal{H}$, and for each state $s \in S$, finding the probability of transitioning to state $s' \in S$ through calculating

$$P(s'|s) = \frac{\sum_{t\in H}(\mathbf{1}\{x_t = s\}\cdot\mathbf{1}\{x_{t+1} = s'\})}{\sum_{t\in H}\mathbf{1}\{x_t = s\}}, \tag{46}$$

which looks at how often we have transitioned from state s at time t to state s' at time $t+1$ out of all the times that we have been in state s at time t.
This fits into the concept of transition probabilities that we have discussed in 6.1.0.6, where transition probabilities depend on only our current state.

Notably, our chosen action $a \in A$ does not impact the transition probabilities within the stochastic components of our system, i.e. charging/discharging the battery will not change the electricity prices, demand, or PV generation. This means that we can find the transition probabilities independently of our chosen actions, before expanding our state space by considering the set of feasible actions, A.

While we define S a specific way (as we see in Section 6.1.0.9), it's virtually impossible to accurately capture all the information needed to make the best possible decision, the aim of our MDP formulation is to create a reasonably sized state space that captures as much relevant information as possible for minimising our system's costs.

Once we address how the MDP modelling framework can be applied to the BESS control problem, we can explore some of the existing methods for achieving this.

6.3 MDP BESS Control Literature

Now that we have addressed the foundations of the MDP framework, and how it can be used to model our BESS control problem, we can incorporate it in our problem. We explore the approach presented by van de Ven, P. M., *et al.* [35], as it serves as a valuable starting point for investigating MDP-based models.

The MDP utilised in [35] is defined by the tuple $(S, A, \mu_0, T, r, \gamma, H)$, where the day is discretised into hours, yielding $H = 24$. Then, state-space S is defined as $\Omega = D \times P \times T$, where D represents all possible demands, P encompasses all possible prices, and T denotes the modulating state (time of day) defined as $T = \{i : 1 \leq i \leq H \cap i \in \mathbb{Z}\}$, capturing intervals of the day.
Transition probabilities $T : S \times A \rightarrow \Delta(S)$ are calculated under the assumption that demand and price in a given interval are independent of those in any previous interval, allowing for straightforward empirical probability distribution calculation for each modulating state.
The action space A at interval $1 \leq i \leq H$ is given by the set $\{a_i : -\min(eB_{i-1}, P/2) \leq a_i \leq \min((c - B_{i-1})/e, P/2) \cap 2a_i \in \mathbb{Z}\}$, where e represents efficiency, B_i denotes the battery's state of charge at time i, c is the battery's capacity, and P represents the battery's power. The constraint of $2a_i$ being an integer limits the possible actions to multiples of 0.5kWh, reducing problem complexity.

The reward function $r(a_i, c_i, d_i)$ is defined as $r(a_i, c_i, d_i) = \max(0, d_i + a_i)c_i$, as export is not allowed in this model.
The discount factor γ is set to 0.99, indicating a small perceived loss in opportunity for future intervals, while initial distribution μ_0 is degenerate, with the first state specified.

The model is tested iteratively, with training performed on the preceding month's data and testing on the subsequent month's data (e.g., train on January's data and test on February's data). Computational results can be found in [35].
However, this paper does appear to have some limitations. While challenges such as not permitting solar generation can be overcome (the possibility of adding PV generation is even mentioned within the paper), the primary issue within this paper is to do with the size of the battery used in the numerical examples and model validation, which is approximately 30 times the average hourly household load. Consequently, the savings appear disproportionately high due to the unrealistic battery size rather than the quality of the algorithm. If going by the battery market-value cost of \$1400\kWh, then the cost of such a battery would take over 30 years to recoup (which is more than the standard battery life of 25 years), even without accounting for degradation, as we show in Figure 24. Since the authors only consider savings and not returns on investment, the model erroneously appears to perform exceptionally well, showing a 38% yearly savings rate.

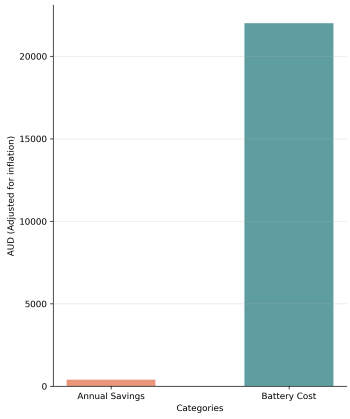
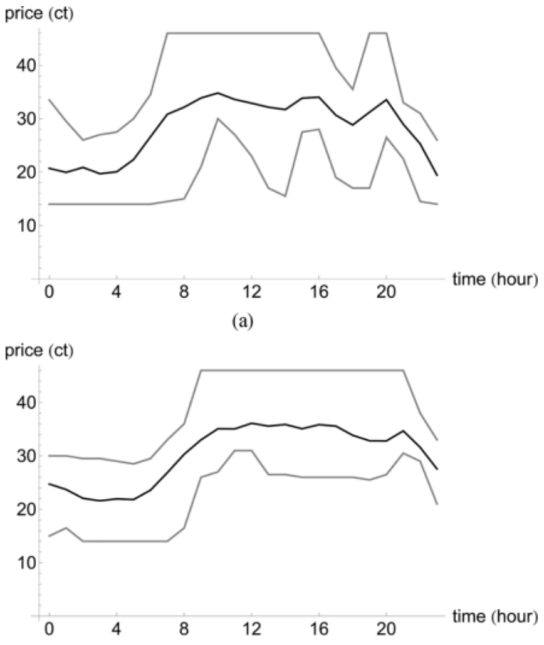


Figure 24: Annual savings with optimisation algorithm in [35], compared to initial investment

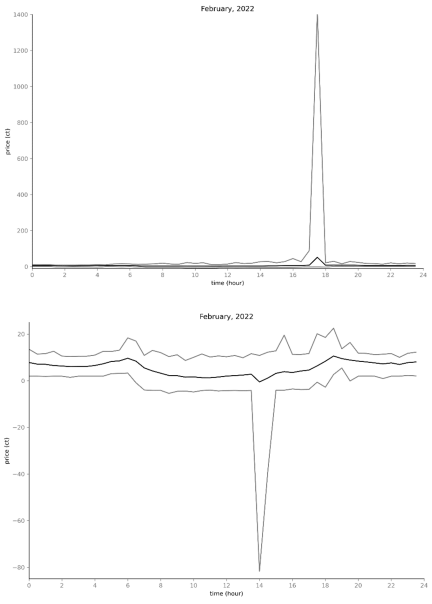
Furthermore, the paper's numerical example features a battery with unrestricted charging

and discharging power, which is highly unrealistic. In practice, accounting for such limitations is a crucial aspect of effective BESS control.

Another concern pertains to the data used in the paper. The household demand data is artificially generated and appears to be more predictable than real-world scenarios, particularly when considering month-to-month variations, which influence price predictions. Additionally, the spot prices employed by the authors exhibit significantly lower volatility compared to what is typically observed, especially in contrast to the NEM, as we demonstrate in Figure 25.



(a) Spot Prices[35]



(b) Victorian 2022 Wholesale Market Spot Prices, from AEMO

Figure 25: Comparison of Spot Prices for January and February (c/kWh)

This results in unrealistically high savings that are simply unattainable for the majority of consumers in Australia, as we show in Figure 26.

In addition, the authors compare the savings of a household with a battery to one without a battery. A more realistic approach would involve evaluating a basic charging heuristic and comparing their results to an optimal schedule, as we do in Section 7. Therefore, while [35] offers a solid starting point for delving into dynamic programming methods for BESS control, the paper does have some limitations when it comes to its applicability to our thesis.

While there is some additional literature on this topic, it appears to be limited. Consequently, we narrow our focus on MDP-based BESS control systems to the approach employed in [35].

6.4 MDP BESS Control Methodology

Given the limited existing MDP BESS control literature, for our thesis we narrow our focus of MDP-based BESS control systems to the approach employed in [35].

As the dataset that is used in [35] is not provided in this paper, we test their proposed algorithm using the datasets referenced in Section 3. Furthermore, we extend the model in [35], by allowing PV generation, as is addressed in their paper.

Hence, our MDP is defined by the tuple $(S, A, \mu_0, T, r, \gamma, H)$, where the day is discretised into half-hours, yielding $H = 48$. Then, state-space S is defined as $\Omega = B \times D \times P \times T \times G$, where D represents all possible demands, P encompasses all possible prices, T denotes the modulating state (time of day) defined as $T = \{i : 1 \leq i \leq H \cap i \in \mathbb{Z}\}$, capturing intervals of the day, G encompasses all possible PV generation amounts, and B is our battery's state of charge. Transition probabilities $T : S \times A \times G \rightarrow \Delta(S)$ are calculated under the assumption that electricity demand, price, and PV generation in a given interval are independent of those in any previous interval, allowing for straightforward empirical probability distribution calculation for each modulating state. Of course, this differs from the real-world situation, but this assumption from [35] is kept in order to maintain tractability, as the program's computational time grows quickly. The action space A at interval $1 \leq i \leq H$ is given by the set $\{a_i : -\min(eB_{i-1}, P/2) \leq a_i \leq \min((c - B_{i-1})/e, P/2) \cap 2a_i \in \mathbb{Z}\}$, where e represents efficiency, B_i denotes the battery's state of charge at time i, c is the battery's capacity, and P represents the battery's power. The constraint of $2a_i$ being an integer limits the possible actions to multiples of 0.5kWh, reducing problem complexity, as is done in [35].

The reward function r is defined as $r(a_i, c_i, d_i) = \max(0, d_i + a_i - g_i)c_i$, as export is not allowed in our model in order to reduce its complexity. Our initial distribution μ_0 is degenerate, with the first state specified, and our tolerance value is $\epsilon = 0.1$.

As is done in [35], we use the VIA algorithm to find our state values for our infinite-horizon MDP. We determine the values of our state by performing the VIA algorithm on a

number of months that precede our testing interval (which is always of length one month). We then test our model by iterating over each interval of our testing period, and undertaking the action that minimises the Bellman Equation for the system inputs, as we show in Section 6.4.

Our final cost is given by the sum of the cost functions, as they are defined in Section 6.3, over all the intervals of our testing period set, *Test*:

$$\text{Cost} = \sum_{i \in Test} \max(0, d_i + a_i) c_i. \tag{47}$$

Furthermore, the discount factor γ is tested for a few different values, as it is unclear what the true loss of opportunity is. And, by testing a few different values we can observe at what value our system performs the best, as well as allowing us to examine the change in runtime for different values of γ, given that a higher value of γ leads to a slower convergence for the VIA algorithm.

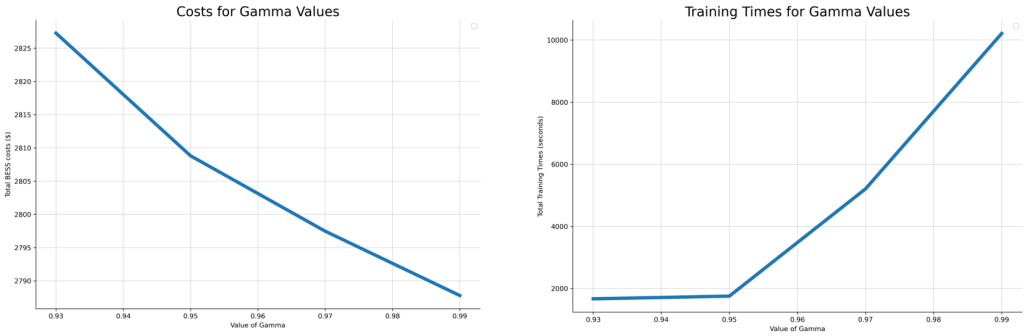


Figure 26: Monthly costs and training times, after a month of training, based on different values of γ

Using the results in Figure 26 above, we see the best percentage savings that we attain here is 11.02%. When we mention savings, it refers to when we compare our MDP's BESS costs to the unoptimised lower bound that we detail in Section 7 below.

It is difficult to understand what to make of this savings percentage, since we compare it neither to the optimal savings possible, nor to the savings from other optimisation algorithms, as is also the case in [35]. This elucidates the need to create a robust and objective evaluation framework, as we attempt to do in Section 7.

7 Performance Evaluation

When we explore different papers that attempt to create BESS control frameworks, as well as when we test our own BESS control frameworks, we notice one key issue: it is often very difficult to quickly and accurately understand the quality of an algorithm's performance.

Comprehensive and objective performance evaluation is essential as it helps prevent the perceived inflation of an algorithm's performance, as observed in [35]. Even for papers that accurately represent their algorithms' performances, readers can often struggle to evaluate them, as the metric typically used is a percentage savings or a dollar amount, not compared to state-of-the-art approaches but rather to unoptimised systems lacking battery storage or PV generation. Furthermore, it does not consider that some instances of the BESS control problem are more straightforward to optimise than others.

Such evaluation approaches are common among the papers we review, including works such as [3], [13], [19], [28], [35], [37], and [39]. While these papers' approaches can be very effective, comparing the results of different papers often requires readers to invest significant time and effort in implementing and manually evaluating their algorithms, as each instance of the BESS control problem is unique. By slowing down the evaluation of existing approaches, this overhead can potentially slow down the development of new and improved optimisation algorithms.

To address these challenges, we propose a different method for evaluating BESS control frameworks. Our method entails finding each algorithm's savings optimality gap. That is, we want to find what an algorithm's savings are, on top of the savings that is inherently generated by the BESS (these are the savings when we compare costs to the energy costs without BESS). An optimality gap is an intuitive way of evaluating an algorithm's performance, and is widely used in scientific literature. To enable an objective evaluation of the savings optimality gap, we establish two key values.

The first is an optimality bound for system performance. By applying the entirety of the actual data to a linear solver, we can determine the maximum possible savings. As deterministic optimal energy storage problems are inherently linear, this approach provides an optimal lower bound on system costs (an upper bound on savings). This value allows us to

evaluate algorithm performance by comparing it to the best possible scenario. We refer to these optimal costs as OC.

The second value represents savings when no optimisation algorithm is used, which means that these are the savings from the BESS itself, independent of any optimisation algorithm used. Thus, it is a lower bound on when an optimisation algorithm can be considered to be optimising the system's performance.

This invariant is calculated using a basic method: during any interval, if PV generation exceeds demand, the excess generation charges the battery to the maximum amount. Conversely, if demand exceeds PV generation, the battery discharges to satisfy demand. Similar to the upper bound, this lower bound allows us to assess how well an algorithm performs compared to an unoptimised system under identical conditions. We refer to these costs as BC, denoting benchmark costs. This is an intuitive way to calculate the savings that are generated by the BESS itself.

While most current papers do not utilise this approach, and only calculate the savings based on a system with no BESS, utilising this approach ensures that an algorithm does not get credit for savings that are inherent to the underlying BESS system, rather than the algorithm itself.

Using the established lower and upper bound values, we can normalise any algorithm's performance, and find its savings optimality gap, using the formula

$$PR = \frac{BC - AC}{BC - OC}, \tag{48}$$

where AC represents algorithm costs, (i.e., the costs achieved using the evaluated algorithm), and PR stands for performance ratio. We derive this formula by dividing the savings of the optimisation algorithm over an unoptimised system by the best possible savings over those of the unoptimised system. This is a modified optimality gap for the savings elicited by the optimisation framework.

7.1 Evaluating Our Results

Using our new evaluation methodology we can now evaluate our different BESS control frameworks in order to better understand how they perform. We plot our results below in Figure 27 in order to better understand the average expected performances of the three algorithms.

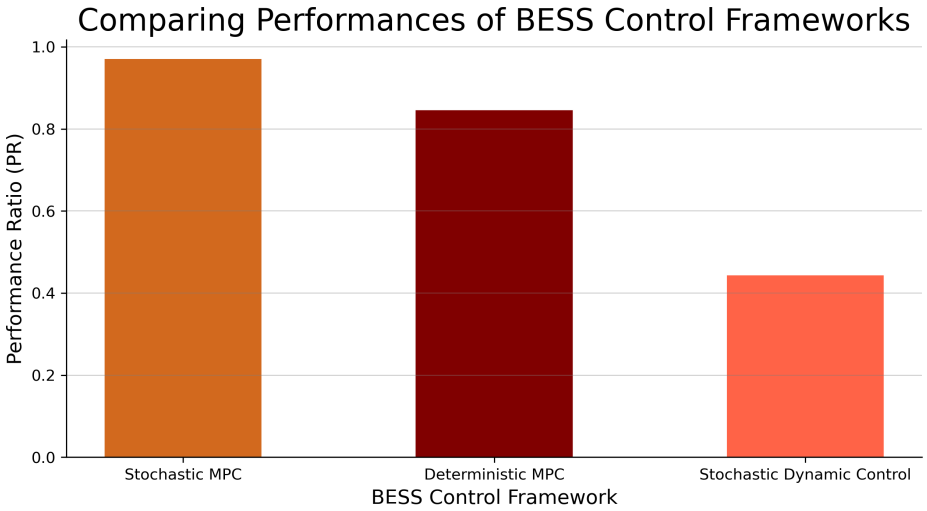


Figure 27: Comparing the PR values of a stochastic MDP (left), deterministic MDP (centre), and stochastic dynamic control (right) for a set of testing data.

We see that the stochastic dynamic control framework is significantly outperformed by the deterministic MPC method, which itself is outperformed by the stochastic MPC method (as expected). The MDP method is significantly outperformed despite it being viewed as a much more sophisticated framework, which further underscores the importance of being able to objectively evaluate the performance of BESS optimisation frameworks, as well as comparing their performances to those of existing frameworks.

The reason for why our SDP framework produces underwhelming results is because of how we choose to define our state space. Clearly our state space does not capture enough information about the state of our BESS in order to make the best decision. Improving our BESS requires us to increase our state space in order to include more relevant information. This information can include more historical data, such as data from the last few intervals, rather than only measurements for the current state. Furthermore, incorporating some forecasts into our state space can also lead to significant improvements.

The key reason for why we choose not to include extra information in our state space is due to the long computation times required to determine the values of the states in our system, likely due to our use of the dated VIA algorithm (described in Section 6.1.0.15). As we mention in Section 6.1.0.16, since the development of the VIA algorithm new methods have been developed, many of which have much faster run-times. Hence, the SDP framework should be reattempted using a method with faster computational times that can subsequently handle a larger state space that captures more critical information required for optimal decision making.

Another reason for the long computation times is due to the use of an MDP with an infinite horizon in [35], which takes longer to compute given that we need to iterate over a larger number of states, especially when the discount factor, γ, is high (as we show in Figure 26 above). It is pretty straightforward to convert this MDP system to a finite horizon MDP with the boundary conditions simply given by the final state of charge of the battery multiplied by the average cost of energy, given that it would take into consideration how much charge we're leaving in the battery for the future - rather than explicitly having an infinite horizon. This would likely significantly speed up computational times, while also allowing us to set $\gamma = 1$, given that it better reflects our system's true value of γ, as our energy does not depreciate in value.

We conclude this section by again reiterating that the reason for why we can observe and objectively judge our BESS control frameworks is due to us having a robust and objective evaluation metric using, which we can easily compare the performances of different methods (even for different instances).

8 Further Reading

In our thesis we concentrate on a simplified BESS setup - one without the capacity for energy export to the grid, exempt from network demand charges, and devoid of the complexities associated with Wholesale Spot Market price bands.

The deliberate omission of these aspects serves to refine the focus of our thesis and prevent unnecessary complications in our models. Nonetheless, it is essential to acknowledge and briefly elucidate these facets, despite being beyond the scope of our work. By doing so, we aim to provide a comprehensive context for the challenges of optimising BESS performance and underscore the potential financial advantages inherent in optimising such a multifaceted system.

9 Conclusion

As stated in the introduction, optimal BESS control can have substantial financial implications. However, there is still significant room left for the improvement of BESS control frameworks. In our thesis we explore the theoretical underpinnings of the real-world BESS control problem, including how to handle the inherent uncertainty of the variables that optimal BESS control depends on.

We explore three key frameworks for optimal BESS control, including deterministic MPC, stochastic MPC, and stochastic dynamic control. We show that while stochastic dynamic control is highly regarded, ultimately the challenges in defining a well-working state space that both holds enough information to make a good decision, while simultaneously being small enough to remain computationally tractable, is challenging. This often leads to the less sophisticated MPC methods outperforming it, as we see in our case. However, using newer MDP-based methods, such as Q-Learning, as well as a finite-horizon MDP, could potentially help to overcome the computational tractability challenge that often proves to be the bottleneck.

On top of exploring BESS control frameworks, in our thesis we also explore investigate methods to best evaluate these frameworks' performances. We do this by generating real forecasts in order to ensure that our control mechanisms would perform well in the real-world, given that the assumptions made about the distributions of forecasts can differ from reality, before randomly generating forecasts using real historical forecasts. We also take a look at ways in which we can evaluate the performance of BESS control frameworks in order to ensure that the performance of the algorithm can be evaluated quickly and efficiently regardless of the specific set of data that it is tested on.

There is still a significant way to go in researching optimal BESS control mechanisms, yet with the great financial incentives that BESS holds, coupled with its key role in assisting the economy in the renewable energy transition, it seems inevitable that more quality BESS control frameworks will be developed.

Bibliography

[1] Link to Github repository with Python code and data:
https://github.com/dan-509/Thesis_Repository

[2] Alessandrini, S., Delle Monache, L., Sperati, S., and Cervone, G. (2015). *An analog ensemble for short-term probabilistic solar power forecast*. Applied Energy, Vol. **157**, pp. 95-110. https://doi.org/10.1016/j.apenergy.2015.08.011.

[3] Arnold, M., and Andersson, G. (2011). *Model predictive control of energy storage including uncertain forecasts*. In Power Systems Computation Conference (PSCC), Stockholm, Sweden, Vol. **23**, pp. 24-29.

[4] Barta, G., Nagy, G. B. G., Kazi, S., and Henk, T. (2015). *Gefcom 2014—probabilistic electricity price forecasting*. In Intelligent Decision Technologies: Proceedings of the 7th KES International Conference on Intelligent Decision Technologies, pp. 67-76. Springer International Publishing.

[5] Bellman, R. (1957). *A Markovian Decision Process*. Journal of Mathematics and Mechanics, Vol. **6**(5), pp. 679–684. http://www.jstor.org/stable/24900506

[6] Breiman, L., Friedman, J.H., Olshen, R.A., and Charles J. Stone (1984). *Classification and Regression Trees (1st ed.)*. Routledge. https://doi.org/10.1201/9781315139470

[7] Burgess, N. (2022). *Correlated Monte Carlo Simulation using Cholesky Decomposition*. Social Science Research Network. https://ssrn.com/abstract=4066115

[8] Delle Monache, L., Eckel, F.A., Rife, D.L., Nagarajan, B., and Searight, K. (2013). *Probabilistic Weather Prediction with an Analog Ensemble*. Monthly Weather Review, Vol. **141**(10), pp. 3498–3516. https://doi.org/10.1175/MWR-D-12-00281.1

[9] Dantzig, G. B. (1955). *Linear Programming under Uncertainty*. Management Science, Vol. **1**(3/4), pp. 197–206. http://www.jstor.org/stable/2627159

[10] Fan, S., and Hyndman, R. J. (2012). *Short-Term Load Forecasting Based on a Semi-Parametric Additive Model*. IEEE Transactions on Power Systems, Vol. **27**(1), pp. 134-141. doi: 10.1109/TPWRS.2011.2162082.

[11] Friedman, J.H. (2001). *Greedy Function Approximation: A Gradient Boosting Machine*. The Annals of Statistics Vol. **29**(5) pp. 1189-1232 https://www.jstor.org/stable/2699986

[12] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. **2**. New York: springer. https://doi.org/10.1007/978-0-387-21606-5

[13] Keerthisinghe, C., Verbič G., and Chapman, A. C. (2014). *Evaluation of a multi-stage stochastic optimisation framework for energy management of residential PV-storage systems*. 2014 Australasian Universities Power Engineering Conference (AUPEC), pp. 1-6. doi: 10.1109/AUPEC.2014.6966552.

[14] Kim, D., and Hur, J. (2018). *Short-term probabilistic forecasting of wind energy resources using the enhanced ensemble method*. Energy, Vol. **157**, pp. 211-226. https://doi.org/10.1016/j.energy.2018.05.157.

[15] Kroese, D. P., Taimre, T., and Botev, Z. I. (2013). *Handbook of Monte Carlo Methods*. John Wiley & Sons.

[16] Land, A. H., and Doig, A. G. (1960). *An Automatic Method of Solving Discrete Programming Problems*. Econometrica, **28**(3), pp. 497–520. https://doi.org/10.2307/1910129

[17] Lele, S., and Richtsmeier, J. T. (1991). *Euclidean distance matrix analysis: A coordinate-free approach for comparing biological shapes using landmark data*. American journal of physical anthropology, Vol. **86**(3), pp. 415-427.

[18] Maimon, O., and Rokach, L. (2006). *Data Mining and Knowledge Discovery Handbook*. Vol. **2**, No. 2005. New York: Springer. https://doi.org/10.1007/b107408

[19] Mavrotas, G., Florios, K., and Vlachou, D. (2010). *Energy planning of a hospital using Mathematical Programming and Monte Carlo simulation for dealing with uncertainty in the economic parameters*. Energy Conversion and Management, Vol. **51**(4), pp. 722-731. https://doi.org/10.1016/j.enconman.2009.10.029

[20] Möller, A., Lenkoski, A. and Thorarinsdottir, T.L. (2013). *Multivariate probabilistic forecasting using ensemble Bayesian model averaging and copulas*. Quarterly Journal of the Royal Meteorological Society, Vol. **139**, pp. 982-991. https://doi.org/10.1002/qj.2009

[21] Morgan, N.J., and Sonquist, J.A. (1963). *Problems in the Analysis of Survey Data, and a Proposal*. Journal of the American Statistical Association, Vol. **58**(302). https://www.jstor.org/stable/2283276

[22] Nie, P., Roccotelli, M., Fanti, M.P., Ming, Z., and Li, Z. (2021). *Prediction of home energy consumption based on gradient boosting regression tree*. Energy Reports, Vol. **7**, pp. 1246-1255. https://doi.org/10.1016/j.egyr.2021.02.006.

[23] Norris, J.R. (1998). *Markov chains*. Cambridge university press, No. 2.

[24] Nowotarski, J., and Weron, R. (2018). *Recent advances in electricity price forecasting: A review of probabilistic forecasting*. Renewable and Sustainable Energy Reviews, Vol. **81**(1), pp. 1548-1568. https://doi.org/10.1016/j.rser.2017.05.234.

[25] Pardalos, P.M. (2001). *Linear Programming*. In: Floudas, C.A., Pardalos, P.M. (eds) Encyclopedia of Optimization. Springer, Boston, MA. https://doi.org/10.1007/0-306-48332-7_261

[26] Pashenkova, E., Rish, I., and Dechter, R. (1996). *Value iteration and policy iteration algorithms for Markov decision problem*. In AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning.

[27] Powell, W.B. (2014). *Clearing the Jungle of Stochastic Optimization*. In INFORMS Tutorials in Operations Research., pp. 109-137. http://dx.doi.org/10.1287/educ.2014.0128

[28] Ren, H., Zhou, W., Nakagami, K., Gao, W., and Wu, Q. (2010). *Multi-objective optimization for the operation of distributed energy systems considering economic and environmental aspects*. Applied Energy, Vol. **87**, pp. 3642-3651. https://doi.org/10.1016/j.apenergy.2010.06.013

[29] Solcast. Live and Forecast Accuracy. solcast.com. https://solcast.com/forecast-accuracy.

[30] Thomas, G. (2007). *Markov decision processes*.

[31] Torres D., Crichigno, J., Padilla, G., and Rivera, R. (2014). *Scheduling coupled photovoltaic, battery and conventional energy sources to maximize profit using linear programming*. Renewable Energy, Vol. **72**, pp. 284-290. https://doi.org/10.1016/j.renene.2014.07.006

[32] Torres-Barrán, A., Alonso, Á., and Dorronsoro, J.R. (2019). *Regression Tree Ensembles for Wind Energy and Solar Radiation Prediction*. Neurocomputing, Vol. **326,327**, pp. 151-160, https://doi.org/10.1016/j.neucom.2017.05.104.

[33] Touzani, S., Granderson, J., and Fernandes, S. (2018). *Gradient boosting machine for modeling the energy consumption of commercial buildings*. Energy and Buildings, Vol. **158**, pp. 1533-1543.

[34] Visser, L., AlSkaif, T., and van Sark, W. (2022). *Operational day-ahead solar power forecasting for aggregated PV systems with a varying spatial distribution*. Renewable Energy, Vol. **183**, pp. 267-282. https://doi.org/10.1016/j.renene.2021.10.102.

[35] van de Ven, P. M., Hegde, N., Massoulie, L., and Salonidis, T. (2013). *Optimal control of end-user energy storage*. IEEE Transactions on Smart Grid Vol. **4**(2), pp. 789–797. doi: 10.1109/TSG.2012.2232943.

[36] Wang, J., Li, P., Ran, R., Che, Y., and Zhou, Y. (2018). *A short-term photovoltaic power prediction model based on the gradient boost decision tree*. Applied Sciences, **8**(5), pp. 689.

[37] Wan, T., O'Neill, D., and Kamath, H. (2015). *Dynamic Control and Optimization of Distributed Energy Resources in a Microgrid*. IEEE Transactions on Smart Grid, Vol. **6**, pp. 2884-2894. doi: 10.1109/TSG.2015.2430286.

[38] Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*.

[39] Zhou, Z., Zhang, J., Liu, P., Li, Z., Georgiadis, M., and Pistikopoulos, E. (2013). *A two-stage stochastic programming model for the optimal design of distributed energy systems*. Applied Energy, Vol. **103**, pp. 135-144. https://doi.org/10.1016/j.apenergy.2012.09.019