MAE 573

Optimization Methods for Energy Systems Engineering

Jesse D. Jenkins, Gabriel Mantegna

Fall 2024

Homework 6 (HW6)

Posted: Monday, November 11th, 2024
Due: Monday, November 18th, 2024 before 8pm

10 points in total for this assignment (worth 10% of total course grade)

Assignment Description

In this assignment you are tasked with **developing an algorithm for operating an energy storage device** in a market, in a realistic decision environment where the future is not known. The fact that the future is not known may seem obvious, but perfect foresight is a common assumption in energy models and the implications of this assumption are often not fully appreciated. You will see in this homework that operating storage without perfect foresight is actually quite difficult, which has important implications for both a) storage asset owners who aim to maximize revenue, and b) grid reliability in a decarbonized grid, since it is not straightforward to make sure that storage devices are available when the system is short on capacity. It is entirely possible, but underexplored in the academic literature, that the lack of perfect foresight may make it more difficult to operate a fully decarbonized grid than is commonly assumed.

Your task will be to **develop an algorithm for operating energy storage in this context, and test it using a simulator, which we provide**. You will work in teams of 2 (or 3 if approved by the instructors based on class numbers). Then, the algorithms of all the teams will be tested against one another, on unseen data, and part of your grade will be related to your algorithm's performance in this simulation. The simulator uses data for Austin, TX, for 2023.

In the GitHub repository, there is a file called "Simulator_HW6.ipynb" which contains:

a) some code to read in the relevant data
   - **Please note the code requires a data file called "prediction_scenarios.jld2" to be present in the same directory as the code, which is available on Canvas under Modules > "Data for HW6". You will also have to install the JLD2 package before you can run the code.**
b) a basic linear optimization model which calculates what the storage optimal dispatch and operations would be under perfect foresight, and
c) a simulator for the operations of a 100MW, 400MWh storage device, following the problem components discussed in lecture and in Lab 2.

The simulator as given in the notebook uses a **"buy low, sell high" policy** to make a decision on how much to charge or discharge at each timestep. This policy is a **function** which inputs the state variables, and outputs a decision of how much to charge or discharge at the current timestep. The price in the current timestep is known, but the price in the future is not. (Think of this like trading stocks: you know the price

when you make the decision to buy or sell, but not the future price.) You can see from the output of the notebook that the revenue from this policy is **only 8%** of what the revenue would have been under perfect foresight. This is because the perfect foresight model eeks every last bit of revenue out of all the price variations, and also is prepared to make the most out of price spikes, while the basic policy function only does a very rudimentary job of energy arbitrage based on the buy/sell thresholds.

Your assignment is to develop a policy function that does better than this "buy low sell high" policy, and ideally comes much closer to the revenue under perfect foresight. As described further in the simulator notebook, **when the policy function is called for a given timestep, it has access to a series of 100 forecasts for the next 24 hours**. These forecasts are developed using a regression model on a much longer timeseries, paired with a Markov chain trained on the regression errors for different temperature regimes. The result is a series of forecasts that describes the distribution of possible price scenarios (see the figure below for an example). The scenarios do a relatively good job at capturing the range of possible outcomes, including unforeseen price spikes, but they are not perfect—just like real forecasts. For your policy, you could use either stochastic or robust optimization, but you may modify (or combine) these basic approaches to balance performance under average conditions against the risk of missing out on unforeseen price spikes. However, you can use any policy you like, that uses the inputs provided, and outputs a charge/discharge decision for the current timestep. It is entirely possible that a relatively simple heuristic policy could outperform a more complicated stochastic optimization, and you will be graded only on the performance of your policy, not how complicated it is.

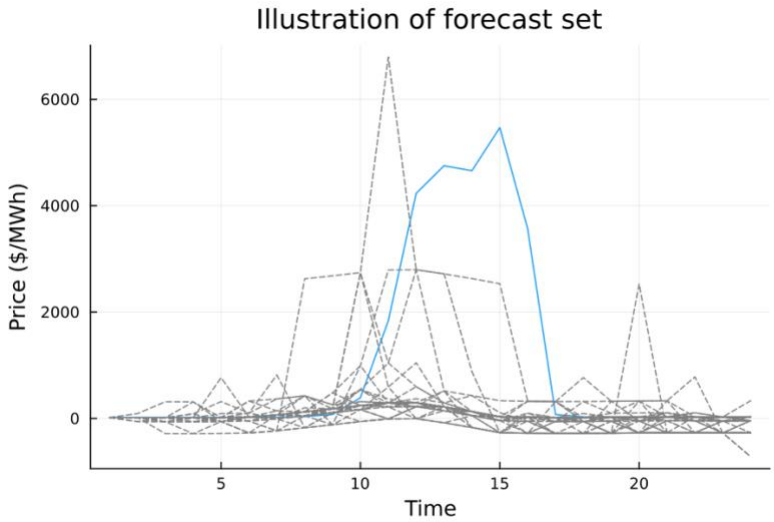

Figure 1: Representative example of a set of price scenarios for one point in time, shown in gray dotted lines, vs the actual price trajectory, shown in blue. As can be seen, the forecast scenarios do a relatively good job of anticipating when there might be price spikes, but they are not perfect.

This assignment should be completed in groups of 2, or 3 if approved by the instructors based on class numbers.

The **deliverables** for this assignment are as follows:

1. A **short write-up** describing: a) the formulation for the policy you developed (please write out the decision variables, constraints etc.), b) any interesting insights you gained from developing your policy (i.e., you can describe any other policies you tried, things that surprised you, etc.), and c) what the performance of your policy was relative to the perfect foresight model. **Please submit**

this as a PDF to Gradescope. Submissions can be one per team, but please include the names of all team members in your submission, as well as your team name.

2. Your version of the notebook where you developed the policy, together with a .jl file that contains your policy function, **submitted together as a zip file to Canvas**. Please name your policy function according to the following format: Policy_YourTeamName(state_variable_.... We will use this function when we compare the policies on unseen data.

The **grading breakdown** for this assignment is:

- 50% short write-up
- 25% performance of your policy in the simulator relative to the perfect foresight model (graded on a curve from 80% to 100%).
    - If it is not feasible to run your policy on the full year with the simulator because it is too computationally intensive to be feasible to run on your laptop, that is okay, and we will run it for you on our supercomputing cluster. More details about this below in the hints section.
- 25% performance of your policy on the unseen data (graded on a curve from 80% to 100%).

There will also be a **10% penalty for submissions not following the submission format described above**, applied to all team members.

Some hints:

- **Write down the formulation** of the policy you are developing before you write the code. Make sure to think about which variables are the "here-and-now" decisions to be made by the policy (aka "first stage" decisions), which will not vary by scenario if you use stochastic optimization, vs the "wait-and-see" decisions which will differ by scenario (aka "second stage" or "recourse" decisions).
- If you are using a stochastic optimization, it may help you to **first develop a policy that uses a deterministic linear optimization** based on the average price forecast for the next 24 hours, and make sure this is working, before implementing a more complicated policy. This will likely help you get familiar with the structure of the code and will also give you a good baseline for performance of a policy that uses optimization.
- **If you need to debug** your policy function, we recommend getting the state variables for just the first time step (or any time step), and then running your policy function code in a cell, not as a function, which will give you more detail about where the bug is occurring. You can also switch to doing all of this in VSCode and use a debugger, if you wish.
- For more complicated policies, the simulator may take a while to run for a whole year. If this is happening we recommend **picking a subset of the year to tune your policy with, then running it once on a full year at the end**. If it turns out to be impractical to run the simulator for a full year at all with your policy, that is okay—we will simulate the policy on the full year for you using our supercomputing cluster.

Good luck!