Multi-Period Compliance Mean Field Game with Deep FBSDE Solver

Orange Ao

October 10, 2024

This is a research report giving big pictures about:

1. *the problem we aim to solve*
2. *the key methods/algorithms we propose*
3. *the main results we get*
4. *comparisons between different methods and consequent results*

Visit my GitHub Repository for more math, algorithm details, and code instructions[1].

[1] Please start an issue at GitHub or reach out to me if you find any problems.

Abstract

This work aims to extend the single-period compliance model in [1] to multi-period, proposing several tricks to improve the numeric stability of the deep solver for FBSDEs with jumps. First by reproducing the aforementioned research by Campbell Steven, et al. (2021), then by considering an additional period to the original model, we make comparisons between long/short-term perspectives when it comes to multi-period production decision-making in renewable electricity certificate markets, as well as between different numeric tricks when it comes to algorithm stability. Meanwhile, some practical takeaways on parameter-tuning are recorded.

1 Problem Overview

Conventional numerical solvers are hard-pressed to solve PA-MFG with market-clearing conditions, which may be faced with the "curse of dimensionality" (Bellman 1957)[2]. Thus in their study [1], Professor Campbell and his fellows proposed an actor-critic approach to optimization, where the agents form a Nash equilibria according to the principal's penalty function and the principal evaluates the resulting equilibria. They apply this approach to a stylized PA problem arising in Renewable Energy Certificate (REC) markets, where agents may *work* overtime (or *rent* capacity), *trade* RECs, and *expand* their long-term capacity to navigate the market at maximum profit. Here we only discuss the agents' problem in the multi-agent-multi-period scenario.

1.1 REC Market Basics

Closely related to carbon cap-and-trade (C&T) markets, REC markets are a type of market-based emissions regulation policies, which are motivating real-world applications of FBSDEs in modeling PA-MFG.

In RES markets, a regulator plays the role of principle, setting a floor on the amount of energy generated from renewable resources (aka. green energy) for each firm (based on a percentage of their total production), and providing certificates for each MWh of green energy generated and delivered to the grid. These certificates can be further traded by individual or companies, i.e. agents, to 1) reduce costs or the greenhouse gas (GHG) emissions impact of their operations; and 2) earn profits from the extra inventories instead of wasting. Since the certificates are traded assets, energy suppliers can trade-off between producing clean electricity themselves, and purchasing the certificates on the market. In all, such policies have played an important role in funding clean energy development, particularly in past years when the cost of green power production was not as competitive with the cost of fossil fuel power.

To ensure compliance, each firm must surrender RECs totaling the floor at the end of a compliance period, with a monetary penalty paid for each lacking certificate. In practice, these systems regulate multiple consecutive and disjoint compliance periods, which are linked together through a mechanism called *banking*, where unused allowances in current period can be carried on to the next period (or multiple future periods). Thus, as an extension to the single-period framework [1], we now consider a 2-period model in this report.[3].

1.2 REC Market Modeling with FBSDEs

Let's consider 2 sub-populations here. Before jumping into the 2-period case, we first reproduce the single-period case following steps in [1]. We denote the period end as T, which can be thought of "the end of the world". Referring to the probabilistic method in [2] (R. Carmona, F. Delarue, 2012), one can show that, for agent i

[2] Bellman, R. E.: Dynamic Programming. Princeton University Press, USA (1957).

[3] At a finite set of joint points, the possible lack of differentiability will not have any significant effects.

in sub-population k, the optimal solution to its problem in a *single* period is exactly the solution to the following coupled FBSDEs:

Now consider the 2-agent-2-period MFG with market-clearing conditions. Let's denote the 2 compliance periods $[0, T_1]$ and $(T_1, T_2]$ as $\mathfrak{T}_1$ and $\mathfrak{T}_2$, respectively. Here we think of T_2 as "the end of the world", after which there are no costs occurs and all agents forfeit any remaining RECs. Similarly, one can prove that the optimal operation for agent i in sub-population k ($\forall\, i \in \mathfrak{N}_k,\ k \in \{1, 2\}$) can be modeled with the following coupled FBSDEs:

$$\begin{cases} dX_t^i = (h^k + g_t^i + \Gamma_t^i + C_t^i)dt + \sigma^k dW_t^k - \min\left(X_{T_1}^i, K\right)\mathbf{1}_{t=T_1}, & X_0^i = \zeta^i \sim \mathcal{N}(v^k, \eta^k) \\ dC_t^i = a_t^i dt, & C_0^k = 0 \\ dV_t^i = Z_t^{V,k} dW_t^i, & V_{T_1}^i = w * \mathbf{1}_{X_{T_1}^i < K} \\ dU_t^i = Z_t^{U,k} dW_t^i, & U_{T_1}^i = 1 * Y_{T_1}^i \mathbf{1}_{X_{T_1}^i > K} \\ dY_t^i = Z_t^{Y,k} dW_t^i, & Y_{T_2}^i = w * \mathbf{1}_{X_{T_2}^i < K} \quad , \end{cases} \tag{1}$$

where the optimal controls are given by: (2)

$$S_t = \frac{\sum\limits_{k\in\|} \frac{\pi^k}{\gamma^k}\mathbb{E}\left[V_t^i + U_t^i \mid i \in \mathfrak{N}^k; \mathcal{F}_t\right]}{\sum\limits_{k\in\mathcal{K}} (\pi^k/\gamma^k)}\mathbf{1}_{t\in[0,T_1]} + \frac{\sum\limits_{k\in\|} \frac{\pi^k}{\gamma^k}\mathbb{E}\left[Y_t^i \mid i \in \mathfrak{N}^k; \mathcal{F}_t\right]}{\sum\limits_{k\in\mathcal{K}} (\pi^k/\gamma^k)}\mathbf{1}_{t\in(T_1,T_2]} \tag{3}$$

$$g_t^i = \frac{V_t^i + U_t^i}{\zeta^k}\mathbf{1}_{t\in[0,T_1]} + \frac{Y_t^i}{\zeta^k}\mathbf{1}_{t\in(T_1,T_2]} \tag{4}$$

$$\Gamma_t^i = \frac{V_t^i + U_t^i - S_t}{\gamma^k}\mathbf{1}_{t\in[0,T_1]} + \frac{Y_t^i - S_t}{\gamma^k}\mathbf{1}_{t\in(T_1,T_2]} \tag{5}$$

$$a_t^i = \frac{(T_1 - t)(V_t^i + U_t^i) + (T_2 - T_1)Y_t^i}{\beta^k}\mathbf{1}_{t\in[0,T_1]} + \frac{(T_2 - t)Y_t^i}{\beta^k}\mathbf{1}_{t\in(T_1,T_2]} \tag{6}$$

(7)

The parameters for each sub-population are given in the following table:

	π^k	h^k	σ^k	ζ^k	γ^k	v^k	η^k	β^k
k=1	0.25	0.2	0.1	1.75	1.25	0.6	0.1	1.0
k=2	0.75	0.5	0.15	1.25	1.75	0.2	0.1	1.0

Table 1: Sub-Population Parameter Values

The framework above can be extended to more realistic models with more than 2 sub-populations and compliance periods, with a penalty approximated by multi-knot functions.

2 Algorithm And Numeric Tricks

2.1 Algorithms: Joint-Optimization Vs. Separate-Optimization

To solve the said FBSDEs in 1.2, we implement the ***shooting method*** with ***Deep Solvers*** [3], discretizing the SDEs in a fine time grid and parameterizing the co-adjoint processes and initial values with neural nets. Let $\mathfrak{T} = \{t_0, \ \ldots\ , t_m\}$ be a discrete set of points with $t_0 = 0$ and $T_m = T$, where m is the number of time steps. Here the step size $dt = (t_i - t_{i-1})$ is a constant and $dt = T/m$. The smaller the value of h, the closer our discretized paths will be to the continuous-time paths we wish to simulate. Certainly, this will be at the expense of greater

computational effort. While there are plenty of discretization schemes available, the simplest and most common scheme is the *Euler scheme*, which is intuitive and easy to implement. In particular, it satisfies the *practical decision-making process* - make decisions for the next point of time conditioned on the current information.

The aforementioned ***shooting method*** is implemented by the ***stepwise approximation***: starting from the initial conditions and *"shoot"* for the "correct" terminal conditions - the "correctness" of terminal approximations will be evaluated by computing the aggregated average forward loss/error over the whole population against corresponding targets (denoted as $\mathcal{L}$). For instance, for the single-period case, the aggregated average forward MSE after m iterations is computed as:

$$\mathcal{L}(\theta^{(m)}) = \sum_{i\in\mathfrak{N}} (Y_T^i - w\mathbf{1}_{X_T^i<K})^2,$$

and for the 2-period case:

$$\mathcal{L}(\theta^{(m)}) = \sum_{i\in\mathfrak{N}} (V_{T_1}^i - w\mathbf{1}_{X_{T_1}^i<K})^2 + \sum_{i\in\mathfrak{N}} (U_{T_1}^i - Y_{T_1}^i\mathbf{1}_{X_{T_1}^i>K})^2 + \sum_{i\in\mathfrak{N}} (Y_{T_2}^i - w\mathbf{1}_{X_{T_2}^i<K})^2.$$

The algorithm takes major steps as follows:

1. start from the neural nets for initial values (i.e. Y_0^i etc.);
2. compute the process values at every time step;
3. get approximations to terminal conditions and compute $\mathcal{L}$;
4. compute gradients of $\mathcal{L}$ against parameters(weights and biases, denoted as $\theta^{(m)}$) in the neural nets (i.e. Y_0^i and Z_t^k etc.) and take gradient steps to determine the next set of parameters.

More specifically, the above steps can be more explicitly displayed by the pseudocode in the Appendix section.

To benchmark the jointly optimized 2-period Model, we first run the 1-period algorithm for each period, i.e. minimize the agents' costs in either period separately.

2.2 Numeric Tricks

The trickiest problem we are facing is the indicator functions in *terminal conditions*. Another problem is that the ordinary Deep FBSDE Solver may learn $V_t^i, U_t^i, Y_t^i \notin [0,1]$ (let's fix $w = 1$ for now), which is meaningless as they represent the *probabilities* of defaulting (i.e. missing the quota). To solve them, we propose 3 numeric tricks and integrate a combined method into the algorithm.

Sigmoid Approximation A natural way to increase continuity and differentiability is by (see Figure 6 in the Appendix section):

$$\mathbf{1}_{0.9>x} \approx \sigma(0.9 - x), \text{ where the sigmoid function } \sigma(u) = \frac{1}{1+e^{-u/\delta}}. \tag{8}$$

In particular, the parameter δ controls the steepness of $\sigma(\cdot)$ and usually is a small positive number - the smaller δ is, the more closely it approximates the step of the indicator function.

Numeric Clamp Instead of using `tensor.clamp` to brutally clamp values within $[0,1]$, we introduce a more numerically stable way, especially for values close to the 2 interval ends (same applies to V_t^i, U_t^i).

$$dY_t^i = Y_t^i(1 - Y_t^i)Z_t dB_t. \tag{9}$$

Nonetheless, there are limitations to the above approaches. For the *sigmoid approximation*, when δ is too small, there is a great potential for numerical overflow - the exponents could be tremendous especially when X_t is far greater than 0.9, such that `torch.exp(u)` is `inf` when $u \geq 7.1$. This will raise errors/warnings[4] in PyTorch. And for *numeric clamp* to work, we must ensure the **initial values** strictly fall in $(0, 1)$. Thus we propose the third approach:

Logit Clamp Transformation To map the range $[0, 1] \to \mathbb{R}$ while avoids working with large exponents, we let: $\tilde{Y} := w * \text{logit}(Y/w)$. Then apply *Itô's formula* (with superscript $[\cdot]^i$ omitted):

$$d\tilde{Y}_t = (w/2 - Y_t)Z_t^2 dt + wZ_t dB_t \ . \tag{10}$$

Correspondingly, we use *BCEWithLogitsLoss* as the loss function, which combines a Sigmoid layer and the *BCELoss* in one single class. This version is more numerically stable than using a plain *Sigmoid* followed by a *BCELoss* as, by combining the operations into one layer, it takes advantage of the log-sum-exp trick for numerical stability.

Worth mentioning, we experimented with multiple combinations of tricks and loss functions, paired with different optimizers and schedulers. Eventually, we chose `Adamax` and `StepLR` due to their relatively better and more stable performance for all cases in general. Specifically, the 4 valid combinations of tricks and loss functions are shown as follows.

	target type	trick	loss type
1	indicator	logit	BCEWithLogitsLoss
2	indicator	clamp	BCELoss
3	indicator	clamp	MSELoss
4	sigmoid	clamp	MSELoss

Table 2: Valid Combos

[4]Examples of RuntimeError and RuntimeWarning on PyTorch Forums.

3 Results

To facilitate the evaluation of algorithm performances, `plot_results` - a user-friendly class - is constructed.[5], which also visualizes agents' behaviors (or interactions) and their market impacts. Specifically, it produces results from the following aspects:

- **Agents' behaviors and market impacts**
  - Learnt optimal control processes
  - Decomposed inventory accumulation processes
  - Inventory levels during 2 compliance periods
  - Terminal inventories ready-to-submit
  - Market-clearing prices
- **Algorithm convergency and learning loss**
  - Average forward losses against a number of epochs trained
  - Learnt terminal conditions vs. targets

And here are some example diagrams.

3.1 Jointly Optimized 2-Agent-2-Period Sample Results

Here we use the sample results by model[6] with terminal target function: $0.25\,\sigma\big(0.9 - X_T^i\big)$, where $\delta = 0.03$ and $T = T_1, T_2$. The loss function is *MSELoss* and the trick used in the ***stepwise approximation*** is *clamp*, correspondingly. (See Table 2.)

The learned controls are shown as the first row of Figure 1 and the accumulative generations by different means as the second row, correspondingly. The green color denotes the sub-population 1 while red the sub-population 2. (The same applies to all the later plots.) The plots in Figure 2 display a rather good convergence of learned terminal values to their *sigmoid targets*.

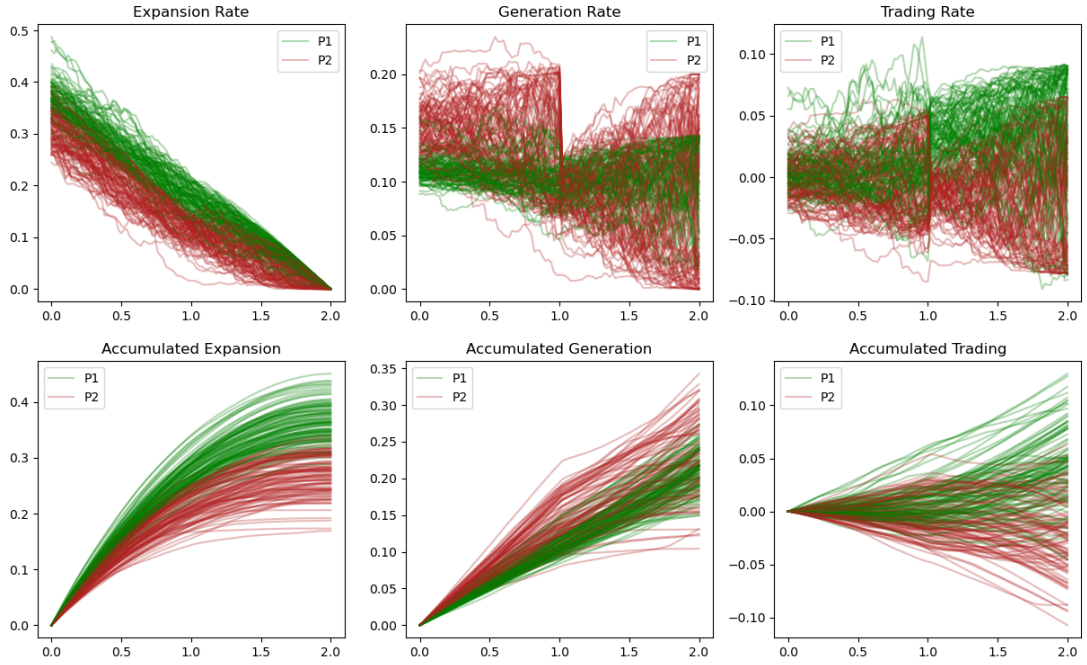

Figure 1: Decomposed Generation of Joint-Optimization

[5]See more instruction details in GitHub README for the Joint-Optimization or Separate-Optimization.
[6]The code and full results can be found in the GitHub Repository.

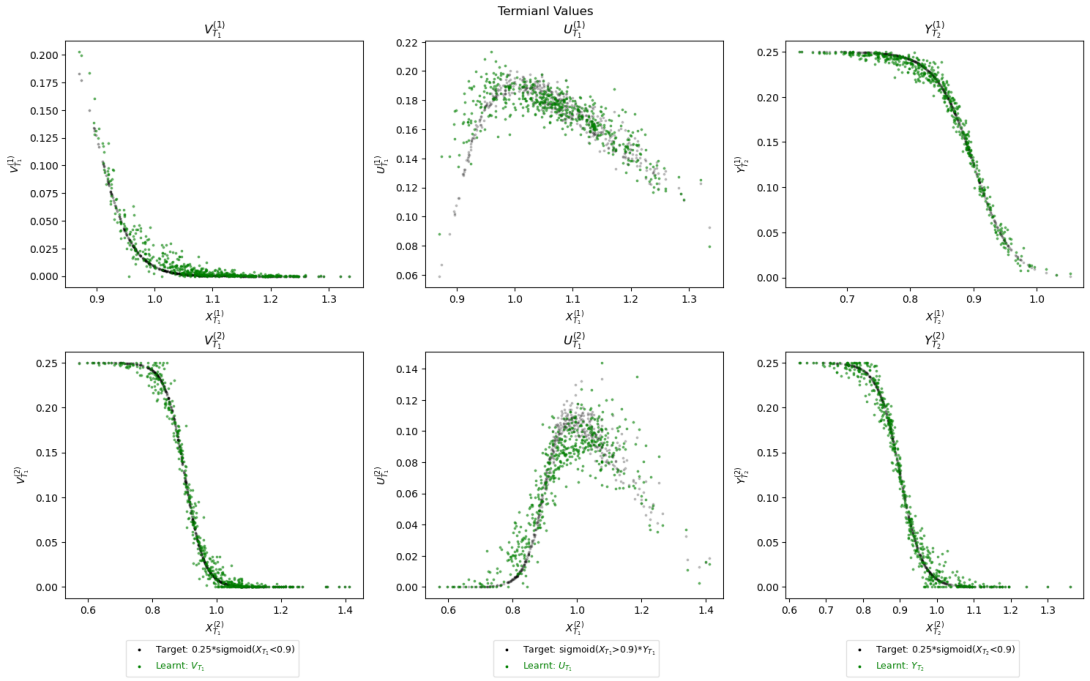

Figure 2: Terminal Values of V, U, Y

3.2 Separately Optimized 2-Agent-2-Period Sample Results

For the sake of comparability, here we use the same numeric trick, target, and loss function, except for minor changes when fine-tuning the specific model parameters[7].

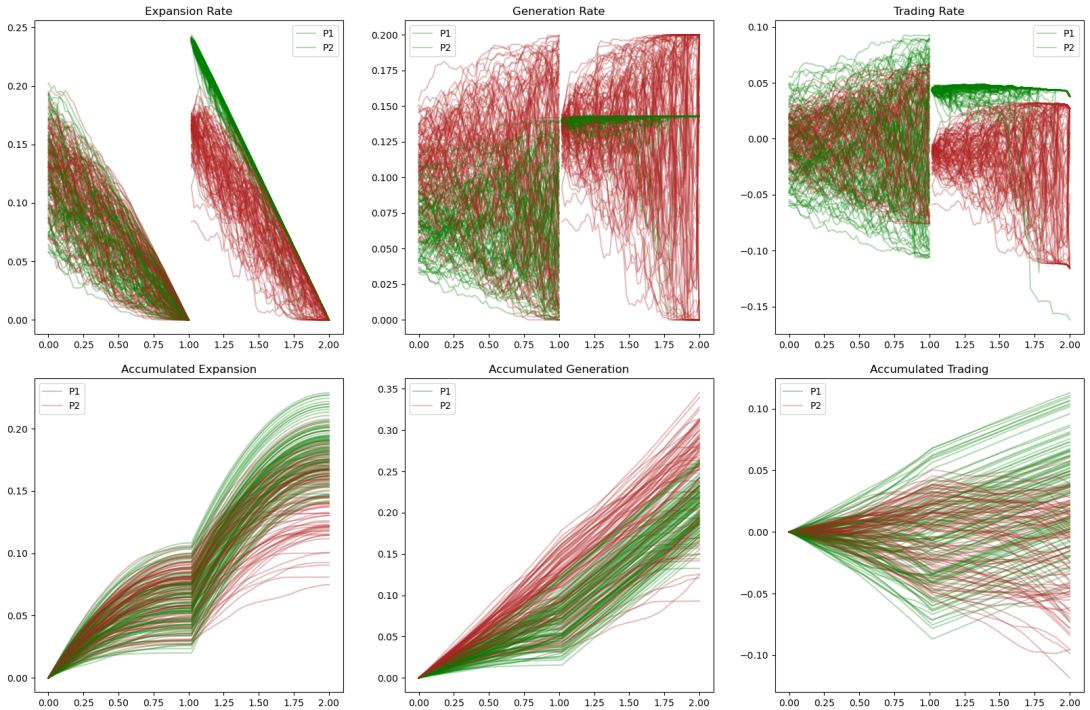

Figure 3: Decomposed Generation of Separate-Optimization

[7]The code and full results can be found in the GitHub Repository.

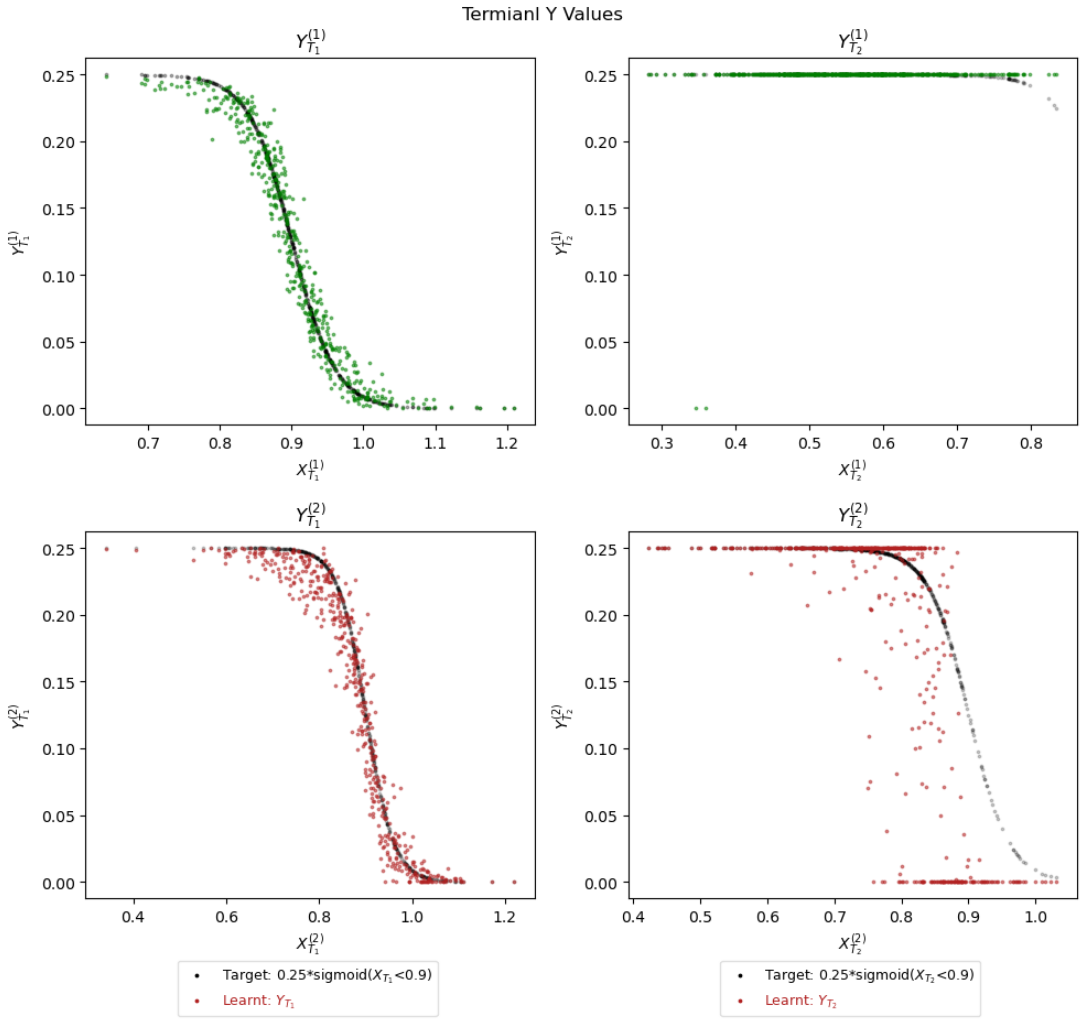

Figure 4: Terminal Values of Y

3.3 Comparisons And Analyses

Joint Vs Separate Optimization Upon comparing the shown results from 2 different perspectives, one can get enlightening implications.

In contrast to the short-sighted agents, those who plan ahead in $\mathfrak{T}_1$ tend to invest more in expansion even at the end of the first period, generating slightly more than the quota required.

Therefore in $\mathfrak{T}_2$, they are saved from starting from scratch. And since most of them have already accumulated a relatively high baseline rate, there's less need to work overtime or purchase inventories than in the first period (shown by the decreased slopes of the accumulated generation plots). However, the agents in the benchmark model have to either 1) try very hard to make up for expansion (exemplified by the green "pop1"), or 2) rely heavily on over-hours and trading (exemplified by the red "pop2").

Their market impacts are reflected in the trading prices: the greater the demand, the higher the prices. This is depicted by Figure 5.

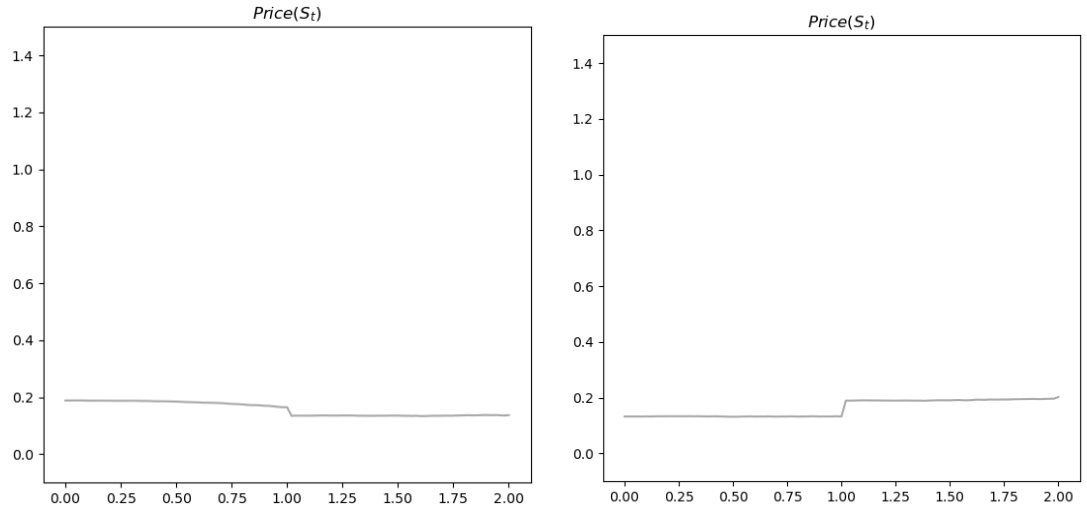

Figure 5: Market Prices: Joint (L) Vs Separate (R) Optimization

Sub-Population 1 Vs 2 Then let's take a closer look at either case, analyzing the differences made by distinctive preferences and initial conditions across sub-populations.

Regardless of perspectives (long/short-term), the explicit advantage in initial inventory level makes the green guys "lazier" in the first period(i.e. less motivated to work extra hours), though they have a hidden shortcoming in baseline capacity. Consequently, in the second compliance period, they are more likely to find themselves hard-pressed to meet the quota and have to purchase from the red guys, which is indicated by the trends and signs (positive for buying and vice versa) of accumulated trading amount.

Model Performances Both algorithms produce descending loss plots and learned terminal conditions that almost overlap with their targets (black dots) given $X^i_{T_1}, X^i_{T_2}$, which suggests converging and stable of model performance as desired. Worth mentioning, since *sigmoid targets* with *MSELoss* have the greatest differentiability, combined with small w (e.g. 0.25) narrowing down the step from 0 to w, models set up as such would produce rather good-looking results. Certainly, there might be other parameter and model settings leading to greater convergence and stability, which are open for experimenting.

4 Conclusions And Takeaways

Conclusions

Enlightening Takeaways From the results and analysis above, one can take away some instructive implications and apply not only to the REC markets but also to one's daily life.

- *Always plan ahead and consider for the future.*
- *Always do slightly more than required and maintain a reasonable level of backups.*
- *Don't be blinded by the apparent advantages/achievements, instead care for the growth rate and capacity - that's what you can carry to the future for sure.*
- *When the majority gets lazy or short-sighted, the market gets worse - where any individual will be affected more or less.*

Appendix

FBSDEs For The Single-Period-2-Agent Model

$$\begin{cases} dX_t^i = (h^k + g_t^i + \Gamma_t^i + C_t^i)dt + \sigma^k dW_t^k, & X_0^i \sim \mathcal{N}(v^k, \eta^k) \\ dC_t^i = a_t^i dt, & C_0^i = 0 \\ dY_t^i = Z_t^k dW_t^k, & Y_T^i = w\mathbf{1}_{X_T^i < K}, \end{cases}$$

where: (11)

$$\begin{aligned} Y_t^i &= \mathbb{E}\left[w\mathbf{1}_{X_T^i<K}|\mathcal{F}_t\right] = w\mathbb{P}\left(X_T^i < K \mid \mathcal{F}_t\right) \\ S_t &= \frac{\sum\limits_{k\in\mathcal{K}}\left(\frac{\pi^k}{\gamma^k}\mathbb{E}\left[Y_t^i \mid i \in \mathfrak{N}^k; \mathcal{F}_t\right]\right)}{\sum\limits_{k\in\mathcal{K}}\left(\frac{\pi^k}{\gamma^k}\right)} \\ g_t^k &= \frac{Y_t^k}{\zeta^k} \\ \Gamma_t^k &= \frac{Y_t^k - S_t}{\gamma^k} \end{aligned} \tag{12}$$

Key Notations

The key notations/parameters in the FBSDEs are interpreted as follows:

- $k \in \mathcal{K}$: a sub-population of agents, within which all individuals are assumed to have identical preferences and similar initial conditions/capacities, yet across which are distinct. The sub-population is annotated by superscript $[\cdot]^k$. Here we only discuss $k = 1, 2$.
- $i \in \mathfrak{N}$: an individual agent belonging to the sub-population $\mathfrak{N}^k$, annotated by superscript $[\cdot]^i$.
- $X_t := (X_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: the current inventories in stock. For some key time points:
  - at $t = 0$, there may be some stochastics in the initial inventories, which are assumed to be normally distributed. $X_0^i \sim \mathcal{N}(v^k, \eta^k),\ \forall k \in \mathcal{K},\ \forall i \in \mathfrak{N}^k$.
  - at $t = T_1$, the terminal RECs pre-submission are X_{T_1} carried over from the first period. Shortly after forfeiting $\min\left(K, X_{T_1}^i\right)$, the remaining inventories in stock are $ReLU\left(X_{T_1}^i - K\right)$, which are treated as new initial values for the second period.
  - at $t = T_2$, the terminal RECs pre-submission are $X_{T_2}^i$.
- $I_t := (I_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: the integrated inventory generation. We introduce this process for continuous differentiability at T_1. And X_t has the same initial conditions as I_t. We have:

$$X_t = \begin{cases} I_t\,, & t \in [0, T_1] \\ I_t - \min(I_{T_1}, K)\,, & t \in (T_1, T_2] \end{cases} \quad \text{or} \quad X_t = \begin{cases} I_t\,, & t \in [0, T_1] \\ I_t - I_{T_1} + (I_{T_1} - K)_+\,, & t \in (T_1, T_2]. \end{cases} \tag{13}$$

- K: the quota that agents must meet at the end of each compliance period. Fixed to $K = 0.9$[8].

[8] The choice of knot point is associated with h^k and total time span T_1, T_2. A good target (or quota) should be **"attainable"** - neither too easy

- $P(\cdot)$: the generic penalty function approximated by ***single-knot penalty functions*** [9] :

$$P(x) = w(0.9 - x)_+ \Rightarrow \partial_x P(x) = -w\mathbf{1}_{x<K}.$$

  Further, by tuning the weight w, we can see the relation between the penalty level (controlled by w) and the agents' behaviour, as well as its market impact.

- h: the baseline generation rate at which agents generate inventories with zero marginal cost.

- $C_t := (C_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: incremental REC capacity of agents, i.e. increase of baseline generation rate over time, accumulated by investing in expansion plans - for instance, by installing more solar panels. [10]

- $a_t := (a_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: the control of expansion rate, representing long-term REC capacity added per unit time. Note that it could be made even more realistic by incorporating a *delay* between the decision to expand (a_t) and the increase to the baseline rate h.

- $g_t := (g_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: the control of overtime-generation rate, i.e. extra capacity achieved by working extra hours and/or renting short-term REC generation capacity at an assumed quadratic cost - specifically, over-hour bonus and/or rental fees.

- $\Gamma_t := (\Gamma_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: the control of trading rate, with negative[11] values being the amount sold whereas positive purchased per unit of time.

- $S_t := (S_t)_{t\in\mathfrak{T}_1\cup\mathfrak{T}_2}$: the equilibrium REC price obtained endogenously through market-clearing condition:

$$\lim_{N\to\inf}\frac{1}{N}\sum_{i\in\mathfrak{N}}\Gamma_t^i = 0$$

- $\zeta,\ \gamma,\ \beta$: scalar cost parameters which are identical for agents within the same sub-population.

- π: the proportion of each sub-population:

$$\pi^k = \frac{|\mathfrak{N}^k|}{\sum\limits_{j\in\mathcal{K}}|\mathfrak{N}^j|}.$$

nor too hard to achieve. Specifically, even if agents do nothing at all, they will have an initial amount plus a baseline generation of inventories - for instance, $0.2 * 1 + 0.6 = 0.8$ for agents in sub-population 1 at the first-period end. Similarly, for sub-population 2, all agents will also have a *"guaranteed"* level of 0.8 for delivery. Thus a target reasonably higher than that, i.e. 0.9, would be regarded **"attainable"**.

[9] See *Report-StepwiseDetail* for more math details.

[10] The incremental capacity over baseline can be carried forward to future periods.

[11] While trading rate may be positive or negative, expansion and overtime-generation rates must be positive.

Algorithms

The stepwise function to approximate the discretized coupled FBSDEs and the main algorithm - forward training loop - is structured correspondingly as:

Algorithm 1: Shooting Method - Stepwise Approximation

Input : Initial conditions and untrained models

Output: Aggregated forward loss ToTLoss

Function `get_forward_loss(`*initial conditions and models*`):`
  Initialization;
  for $j \leftarrow 1$ **to** T_2 **do**
    Update $\mathbb{P}$ (default) for each sub-population:
    $V_j \leftarrow V_{j-1}$ + zv_models [j-1]*dB $_j$, $U_j \leftarrow U_{j-1}$ + zu_models [j-1]*dB $_j$, $Y_j \leftarrow Y_{j-1}$ + zy_models [j-1]*dB $_j$;
    Compute overtime-generation, trading, and expansion rates: g, Γ, and a;
    Update C and X;
    if j=T_1 **then**
      Freeze X, V, and U at T_1;
      Submit inventories: X $\leftarrow$ ReLU(X - K);
    end
  end
  Aggregate the losses: ToTLoss $\leftarrow$ `CalTerminalLoss` (V,U,Y);
  return ToTLoss

Sigmoid Approximation

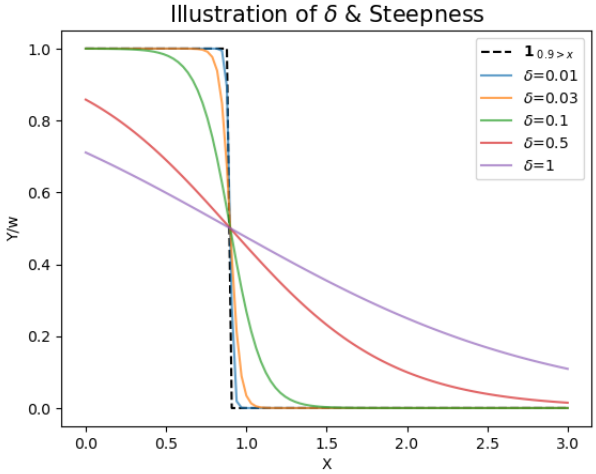

Figure 6: Sigmoid Approximation

Algorithm 2: Training Loop for Forward Loss Approximation

Input : Sample X_0, C_0, and BM Increments dB for each sub-population

Output: Trajectories of X_t, g_t, Γ_t, a_t, C_t, and terminal conditions $V_{T_1}, U_{T_1}, Y_{T_2}$

```
begin
    Initialization;
    for batch ← 0 to MaxBatch do
        SumLoss ← 0;
        for iter ← 0 to OptimSteps do
            Compute forward loss: loss ← get_forward_loss(initial conditions);
            Backpropagate: loss.backward();
            Update parameters: optimizer.zero_grad(), optimizer.step();
            Adjust learning rate: schedular.step();
            SumLoss ← SumLoss + loss;
        end
        AppendToForwardLosses(SumLoss/OptimSteps);
        if not train on the single batch then
            Generate a new batch with updated initial processes;
        end
    end
    Visualize the results: Forward Losses, X_t, g_t, Γ_t, a_t, C_t, and terminal convergence.
end
```

References

[1] S. Campbell, Y. Chen, A. Shrivats, and S. Jaimungal, *Deep learning for principal-agent mean field games*, 2021. arXiv: 2110.01127 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2110.01127.

[2] R. Carmona and F. Delarue, *Probabilistic analysis of mean-field games*, 2012. arXiv: 1210.5780 [math.PR]. [Online]. Available: https://arxiv.org/abs/1210.5780.

[3] J. Han and J. Long, "Convergence of the deep bsde method for coupled fbsdes.," *Probability, Uncertainty and Quantitative Risk*, vol. 5, 2020. [Online]. Available: https://doi.org/10.1186/s41546-020-00047-w.