7 Reinforcement Learning Approach for Battery Trading

To address the computational challenges of the stochastic optimal control formulation presented in Section **??**, we develop a reinforcement learning (RL) approach that can learn optimal trading policies directly from market data without requiring explicit probabilistic models of price dynamics. This section details the RL framework, including the choice of algorithm, environment design, and the integration of expert guidance through Rolling Intrinsic strategies.

7.1 Proximal Policy Optimization Algorithm

We employ Proximal Policy Optimization (PPO) [**?**] as our primary RL algorithm due to its proven stability and sample efficiency in continuous control tasks. PPO is a policy gradient method that optimizes a clipped surrogate objective function to prevent destructively large policy updates, making it particularly suitable for the battery trading domain where stable learning is crucial.

The PPO algorithm optimizes the following clipped objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{7.1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the advantage estimate, and ϵ is the clipping parameter (typically 0.2).

We configure PPO with a `MultiInputPolicy` architecture to handle the dictionary-based observation space efficiently, allowing the neural network to process different types of market and battery state information through specialized input layers.

7.2 Environment Design

7.2.1 Episode Structure

Our trading environment models each trading day as a complete episode, reflecting the natural cycle of electricity markets where positions must be settled within the day. An episode consists of N trading timesteps, typically corresponding to 96 quarter-hourly periods, during which the agent can place trades for any future delivery period within that same day. Currently test and train set are the same, just to see if I can make it work at all.

7.2.2 Action Space

The action space is designed as a continuous Box space with the following characteristics:

$$\mathcal{A} = \text{Box}(\text{low} = -C_{\text{norm}}, \text{high} = C_{\text{norm}}, \text{shape} = (N_{\text{periods}},)) \tag{7.2}$$

where:

- C_{norm} represents the normalized battery capacity in MW/MWh
- N_{periods} is the number of delivery periods (typically 96)
- Each action component $a_t^{(i)} \in [-1, 1]$ represents the trade size for delivery period i. $a_t^{(i)} > 0$ is the trade size for delivery period i at time t. Factor of two here since the agent can trade up to twice the battery capacity (with retrading).
- Positive actions indicate selling/discharging, negative actions indicate buying/charging

To ensure physical feasibility, we implement dynamic action rescaling based on current commitments and battery constraints:

$$a_{\text{scaled}}^{(i)} = \begin{cases} a_t^{(i)} \cdot (C_{\text{norm}} - c_t^{(i)}) & \text{if } a_t^{(i)} > 0 \\ a_t^{(i)} \cdot (C_{\text{norm}} + c_t^{(i)}) & \text{if } a_t^{(i)} \leq 0 \end{cases} \tag{7.3}$$

where $c_t^{(i)}$ represents the current commitment ‘for period i.

7.2.3 Observation Space

The observation space captures all relevant information for trading decisions through a dictionary structure:

$$\mathcal{S} = \{\text{soc} : [0, 1], \quad \text{cycles\_used} : [0, N_{\text{max\_daily\_cycles}}] \tag{7.4}$$

Or should I allow it to be greater than $N_{\text{max\_daily\_cycles}}$,

since the actions of the agent can result in exceeding the limit?

$$\text{commitments} : [-C_{\text{norm}}, C_{\text{norm}}]^{N_{\text{periods}}}, \tag{7.5}$$

$$\text{market\_prices} : \mathbb{R}^{N_{\text{periods}}}, \tag{7.6}$$

$$\text{market\_availability} : \{0, 1\}^{N_{\text{periods}}}, \tag{7.7}$$

$$\text{current\_time\_idx} : \{0, 1, \ldots, N - 1\}\} \tag{7.8}$$

where:

- **soc**: Normalized state of charge $\in [0, 1]$.
- **cycles_used**: Cumulative battery cycles utilized. This is the number of times the sum of battery actions comprise a full cycle (once fully charge and once fully discharge).
- **commitments**: Current power commitments for each delivery period.
- **market_prices**: Power prices for all delivery periods.
- **market_availability**: Binary mask indicating active markets.
- **current_time_idx**: Current trading timestep within the day.

7.2.4 Commitment Tracking and Updates

A critical aspect of the intraday trading environment is the tracking and updating of power commitments across delivery periods. Commitments represent the cumulative net power trades that have been executed for future delivery, enabling the system to support retrading strategies while maintaining physical feasibility.

Commitment Initialization At the beginning of each trading episode (day), the commitment vector is initialized to zero for all delivery periods:

$$\mathbf{c}_0 = [0, 0, \ldots, 0] \in \mathbb{R}^{N_{\text{periods}}} \tag{7.9}$$

Commitment Update Mechanism When the agent executes an action $\mathbf{a}_t$ at trading timestep t, the commitments are updated according to:

$$c_{t+1}^{(i)} = c_t^{(i)} + a_t^{(i)} \quad \text{for all delivery periods } i \geq t \tag{7.10}$$

where $c_t^{(i)}$ represents the total net commitment for delivery period i at trading time t, and $a_t^{(i)}$ is the new trade size for that period.

This additive update mechanism enables several key capabilities:

- **Position Building**: Multiple trades can be accumulated for the same delivery period
- **Retrading**: Previous positions can be partially or fully reversed by trading in the opposite direction
- **Risk Management**: The agent can adjust its exposure as market conditions evolve

Constraint Enforcement The commitment state directly constrains future trading actions through the dynamic rescaling mechanism described in the action space section. Specifically, the available trading capacity for period i at time t is:

$$\text{Available Capacity}_t^{(i)} = \begin{cases} C_{\text{norm}} - c_t^{(i)} & \text{for additional selling} \\ C_{\text{norm}} + c_t^{(i)} & \text{for additional buying} \end{cases} \tag{7.11}$$

This ensures that total commitments (positive and negative) never exceed the battery's physical power capacity C_{norm}.

Physical Execution At each delivery period i, the actual power dispatch equals the final net commitment for that period:

$$P_{\text{actual}}^{(i)} = c_{\text{final}}^{(i)} \tag{7.12}$$

This actual dispatch then updates the battery's state of charge according to the efficiency-adjusted dynamics, providing the physical link between trading decisions and battery operation.

Example Commitment Evolution Consider a simplified 3-period scenario:

1. **t=0**: Initial commitments $[0, 0, 0]$, agent trades $[-5, 10, 0]$ MW
2. **t=0 result**: Updated commitments $[-5, 10, 0]$ MW
3. **t=1**: Agent observes new prices and retrades $[0, -10, 10]$ MW
4. **t=1 result**: Final commitments $[-5, 0, 10]$ MW

In this example, the agent initially planned to charge 5 MW in period 0 and discharge 10 MW in period 1, but later revised the strategy to charge in period 0 and discharge in period 2 instead, demonstrating the flexibility enabled by commitment tracking.

7.3 Reward Structure and Physical Feasibility

To provide a dense and informative reward signal, our environment rewards the agent based on the total intrinsic value of its actions at each timestep. This approach directly attributes profit to the specific trading decisions made, while ensuring that the valuation is always physically feasible by correcting for the battery's state-of-charge limitations.

At each trading timestep t, the total reward R_t is calculated by valuing the agent's action vector $\mathbf{a}_t$. This is achieved by evaluating the effect of the action on the battery from the current time until the end of the day. The value for a single future period k within this simulation is:

$$v_k(\mathbf{a}_t) = \begin{cases} -\frac{\Delta\text{SOC}^{(k)}_{\text{feasible}}}{\eta_{\text{ch}}} \cdot p_k \cdot E_{\max} & \text{if } a_t^{(k)} > 0 \text{ (charging)} \\ \Delta\text{SOC}^{(k)}_{\text{feasible}} \cdot \eta_{\text{dch}} \cdot p_k \cdot E_{\max} & \text{if } a_t^{(k)} < 0 \text{ (discharging)} \end{cases} \tag{7.13}$$

where $\Delta\text{SOC}^{(k)}_{\text{feasible}}$ is the SOC change possible given the simulated SOC at period k, and p_k is the market price.

The total immediate reward R_t given to the agent at step t is the sum of these future values:

$$R_t = \sum_{k=t}^{N-1} v_k(\mathbf{a}_t) \tag{7.14}$$

This structure provides a direct and immediate reward for the quality of the agent's marginal trading decisions. Terminal penalties for operational constraints are still applied at the end of the episode:

$$P_{\text{terminal}} = \begin{cases} \lambda_{\text{violation}} \cdot (N_{\text{cycles}} - N_{\max}) & \text{if } N_{\text{cycles}} > N_{\max} \\ \lambda_{\text{underuse}} \cdot (0.5 \cdot N_{\max} - N_{\text{cycles}}) & \text{if } N_{\text{cycles}} < 0.5 \cdot N_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{7.15}$$

Constraints are not working as expected. Cycle limit stays exceeded even with high penalty. What is a good way to handle this cycle constraint. The final reward received after the terminal state is reached is $R_{\text{final}} = R_t - P_{\text{terminal}}$.

7.4 Expert Guidance through Rolling Intrinsic Integration

To accelerate learning and improve policy quality, we integrate expert guidance from Rolling Intrinsic (RI) optimization strategies (linear optimization with current visible prices and commitments). The RI strategy provides optimal trading schedules based on perfect foresight within a rolling window, serving as a benchmark and learning signal for the RL agent.

The integration works as follows:

1. Pre-calculate optimal RI schedules for all training days using historical price data
2. Store these schedules in a lookup table indexed by trading day
3. During training, provide the RI schedule as additional context (when enabled)
4. Use RI performance as a baseline for evaluating RL agent improvement

This approach enables curriculum learning where the agent can learn from expert demonstrations while still developing its own adaptive strategies for unseen market conditions.

7.5 Training Protocol

Our training protocol emphasizes efficiency and stability:

- **Episode Structure**: Each episode corresponds to one trading day with 96 quarter-hourly periods
- **Multi-Epoch Training**: Train for multiple epochs over the same dataset to ensure convergence
- **Progress Tracking**: Evaluate agent performance periodically during training to monitor learning progress
- **Computational Optimization**: Pre-calculate RI schedules once per day rather than at every timestep to reduce computational overhead

The resulting RL agent learns to balance immediate trading profits with long-term battery health constraints, adapting to market volatility while respecting physical limitations of the energy storage system.

8 Future research

- Theoretical comparison between rolling intrinsic and extrinsic value.
- Lack of liquidity considered by making it impossible to trade at certain times.