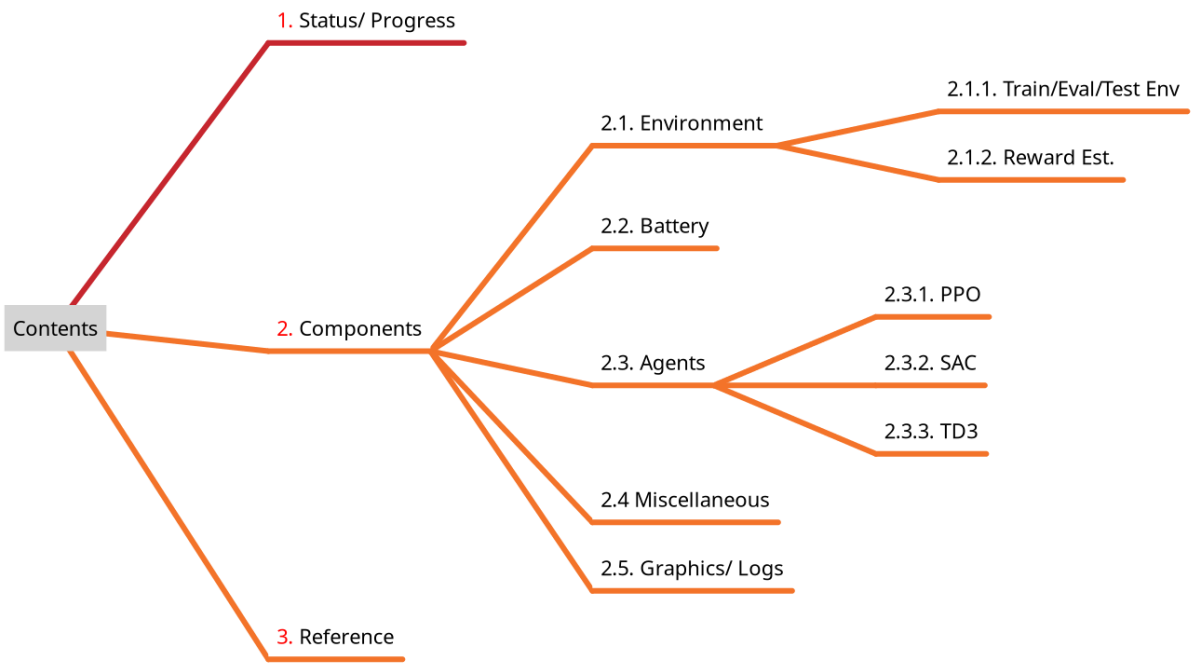
Training Prosumer Agents with Reinforcement Learning.

>>> **Biweekly Report 6.** ($25^{th}\,Apr - 1^{st}\,June : 2024$)



---

1. Status/ Progress

Current Iteration

- [x] comparison with rule based vs rl algorithms.
- [x] Shuffled batch training setup
- [x] Definition of reward function to address action & soc constraints
- [x] Impact of different combination of input observation, parameters on rewards.

Next Iteration (Plan)

- [ ] improved cost comparison with rule based vs rl models.
- [ ] improvement on reward function with better parameter search.

---

2. Components

Taking feedback from weekly catchups into account and updated understanding of the system following changes were made to different components.

2.1. Environment Development

2.1.1 **Train/Eval/Test Env**

- A SoC constraints was added for agent taking each step/action in the environment as below.

```
# soc constraint
if self.battery.current_soc == 0.0:  # if battery empty
    action = np.clip(action, 0, self.MAX_CHARGE_RATE)  # only charge action
elif self.battery.current_soc == 1.0:  # if battery full
    action = np.clip(
        action, self.MAX_DISCHARGE_RATE, 0
    )  # only discharge action
```

2.1.2 **Rewards Estimation**

- Additional soc objectives were added to the reward function that returns the energy cost based on net exchanges.

```
if self.battery.current_soc < 0.2:
    soc_reward = -1  # discourage less than 20% of battery
elif self.battery.current_soc > 0.8:
    soc_reward = -1  # discourage more than 80% of battery
else:
    soc_reward = 1  # encourage within 20-80 % of battery
```

2.2. Battery Module

- no change were made to the battery module.

2.3. Agent

- Algorithm in use:
    - Proximal Policy Optimization (PPO)
    - Soft Actor Critic (SAC)
    - Twin Delayed Deep Deterministic Policy Gradient (TD3)
- Different parameters for initialization(`model.init()`) and learning(`model.learn()`) were added and experimented, however no significant improvement was seen, e.g. Increasing policy network layers, discount factor for future rewards etc. Therefore the default parameters were used for graphics.

2.4. Miscellaneous

- adapted test for modified custom env.
- total cost comparision with rule based vs rl models as shown in the graphics/logs section.
- Addition of search script of relevant parameters using optuna for all the models.

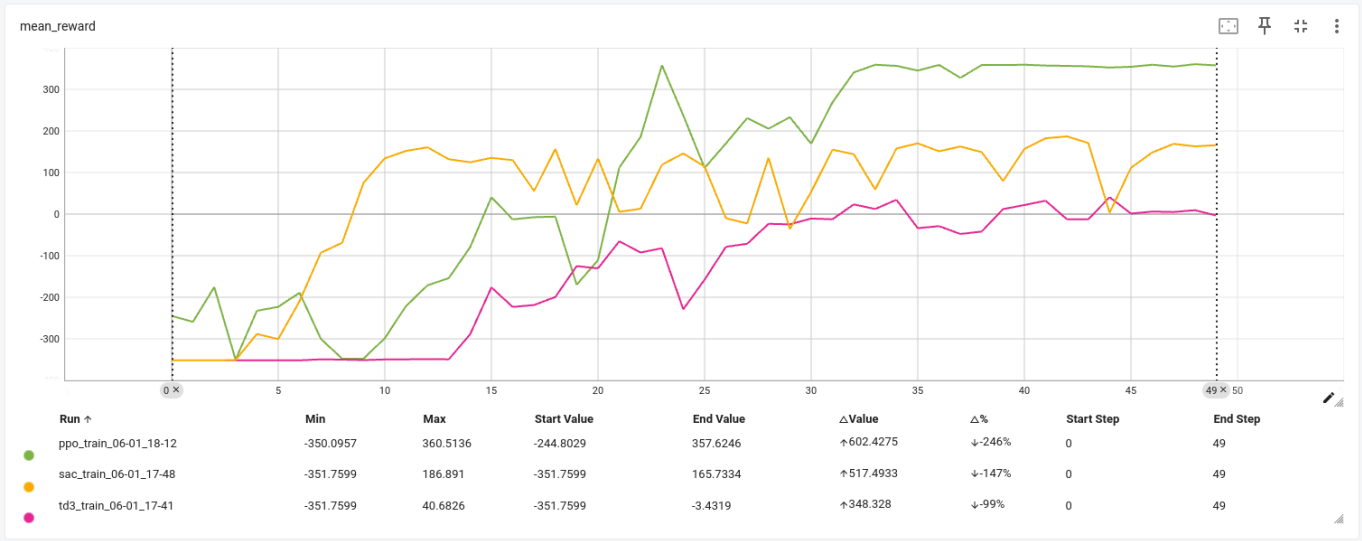
2.5. Graphics/ Logs

parameters config

```
data_config["train_test_split_ratio"] = 0.1
battery_config: dict = {
    "max_capacity": 50,  # kwh
    "max_charge_rate": 11,  # kwh
    "max_discharge_rate": -11,  # kwh
}
env_config: dict = {
    "env_id": "hems_env/HouseholdEnv-v0",
    "observation_window": int(data_config["train_test_split_ratio"] * 35904 *
0.1),
    "num_envs": 1,
    "energy_sell_price": 0.08,
    "energy_buy_price": 0.33,
}
policy_config: dict = {
    "policy_nw": "MlpPolicy",
    "reset_num_timesteps": False,
    "num_train_eval_cycles": 50,
    "num_retrain_eval_cycles": 20,
    "num_eval_episodes": 3,
    "num_test_episodes": 2,
    "train_timesteps": data_config["train_test_split_ratio"] * 35904 * 0.5,
    "retrain_timesteps": data_config["train_test_split_ratio"] * 35904 * 0.5,
}
ppo "learning_rate": 3.4e-5
sac "learning_rate": 0.0003
td3 "learning_rate": 0.0003
```

During Learning/ Evaluation

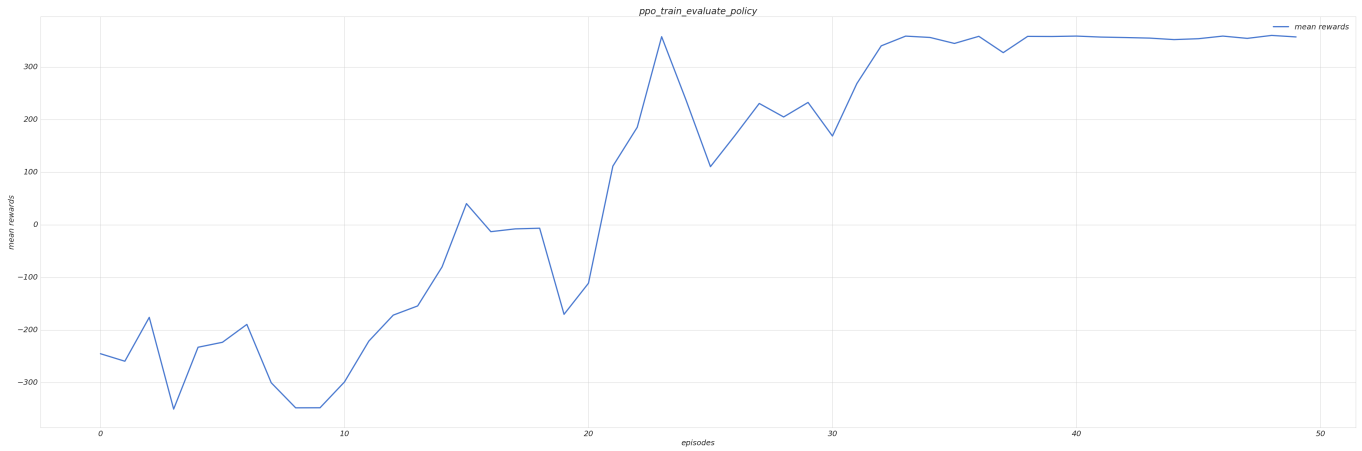
Evaluation of Mean Rewards per iteration : 50, with each model.



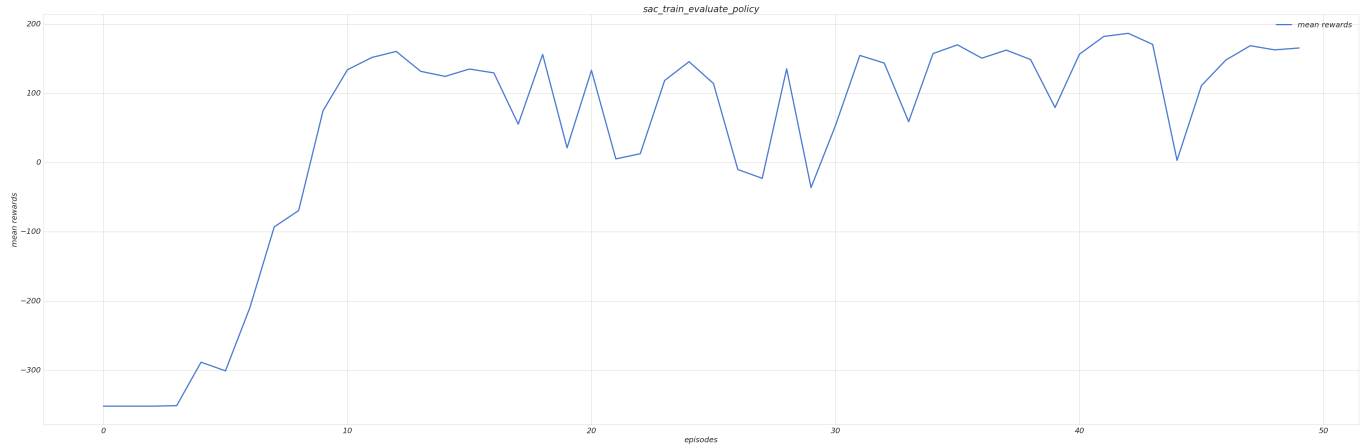
Run	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
ppo_train_06-01_18-12	-350.0957	360.5136	-244.8029	357.6246	↑602.4275	↓-246%	0	49
sac_train_06-01_17-48	-351.7599	186.891	-351.7599	165.7334	↑517.4933	↓-147%	0	49
td3_train_06-01_17-41	-351.7599	40.6826	-351.7599	-3.4319	↑348.328	↓-99%	0	49

- The agents learns to maximize the reward cost and progresses towards maximum value for the given parameter configurations. The detailed results for individual model evaluation is shown in section below.

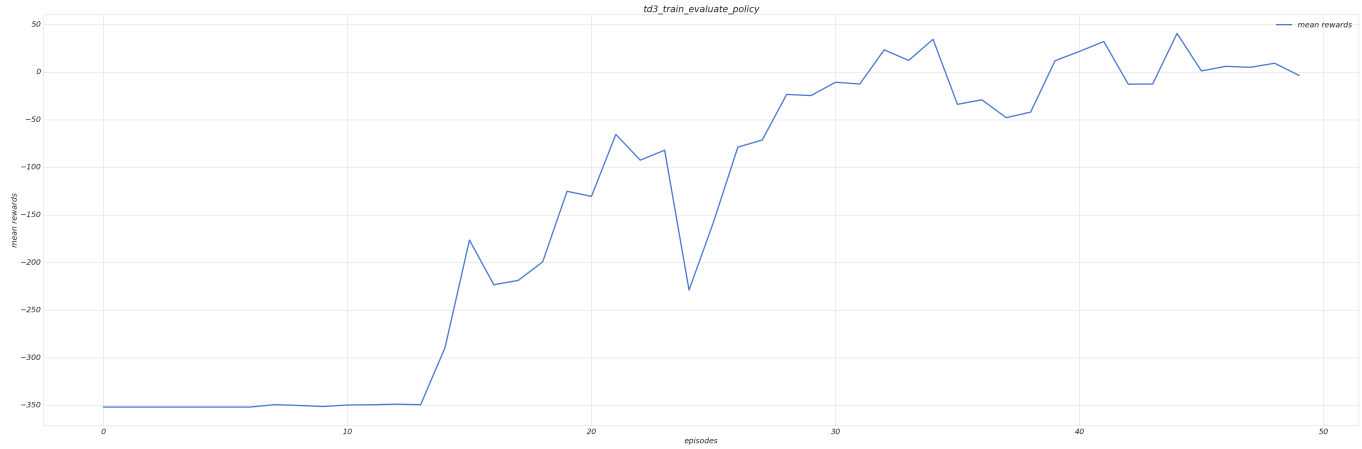
PPO



SAC



TD3

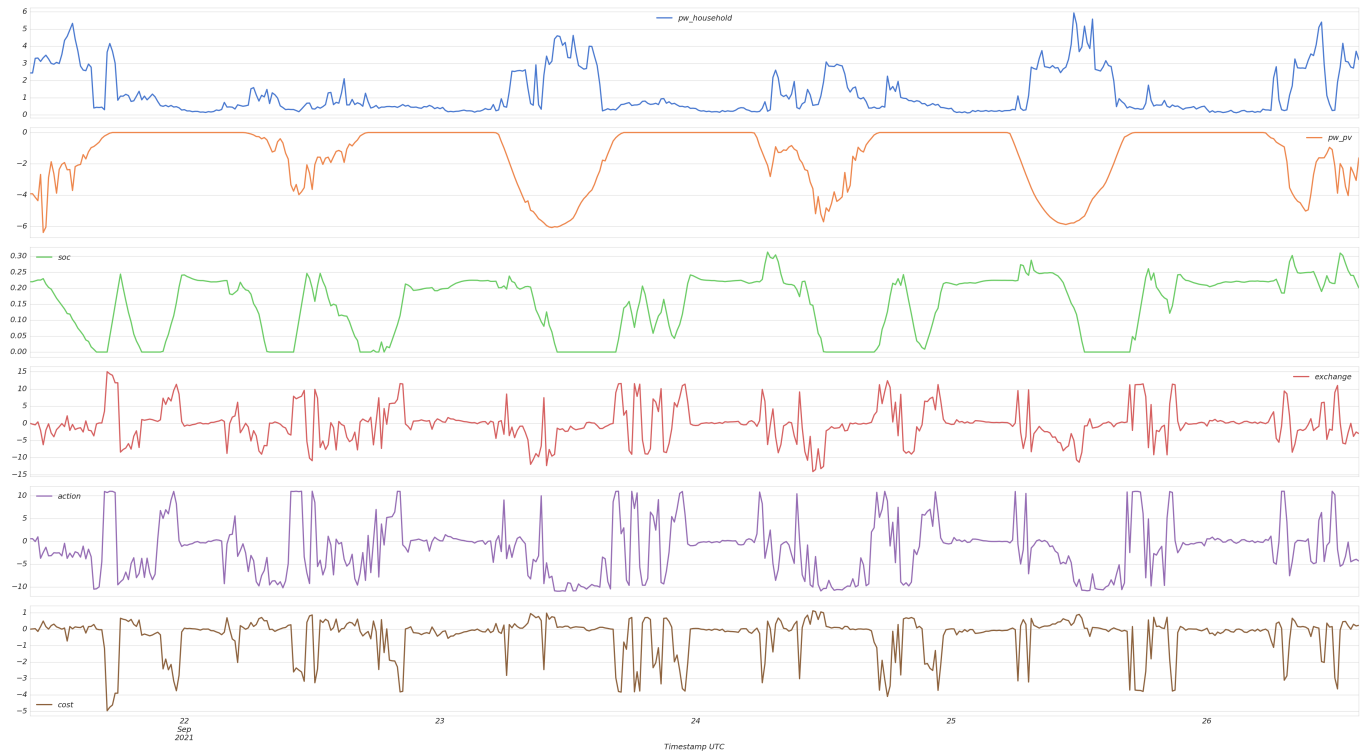


During Sampling Actions from trained policy (deterministic)

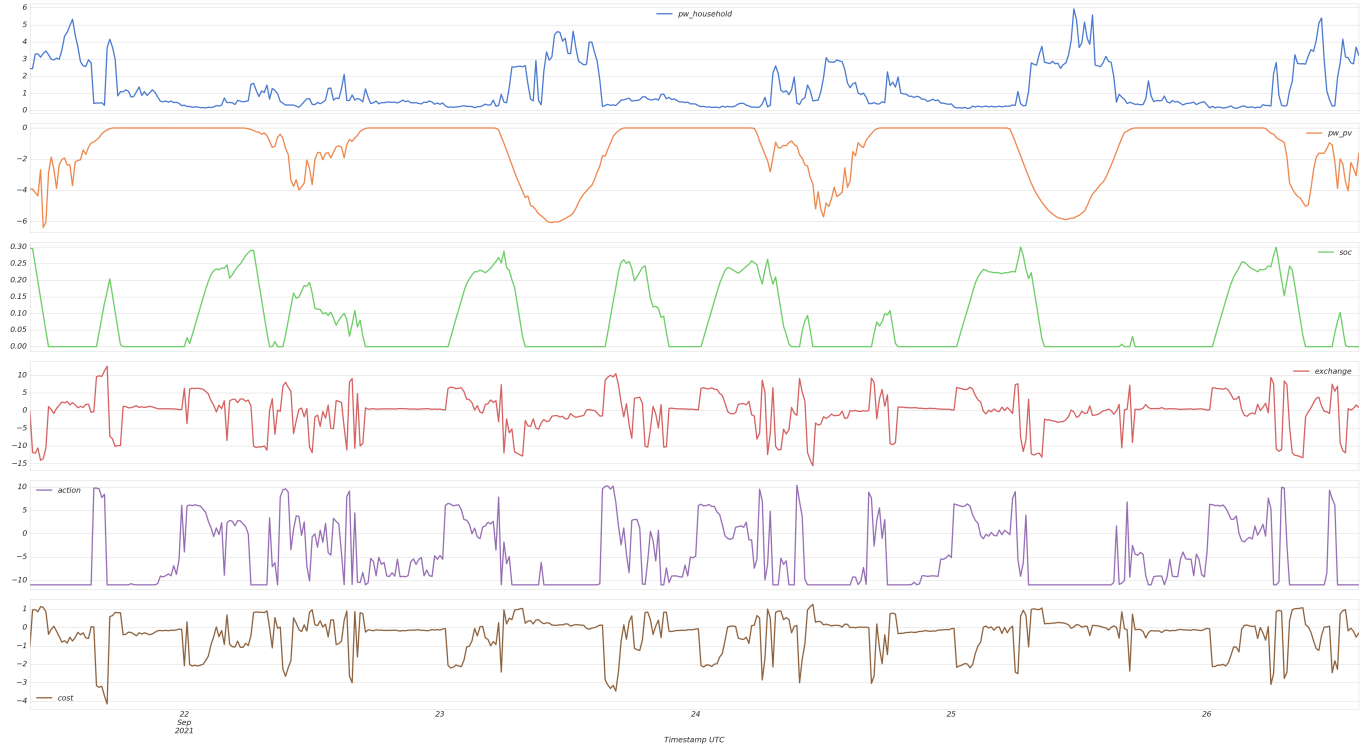
PPO


pw_household
pw_pv
soc
exchange
action
cost
22
Sep
2021
23
24
25
26
Timestamp UTC

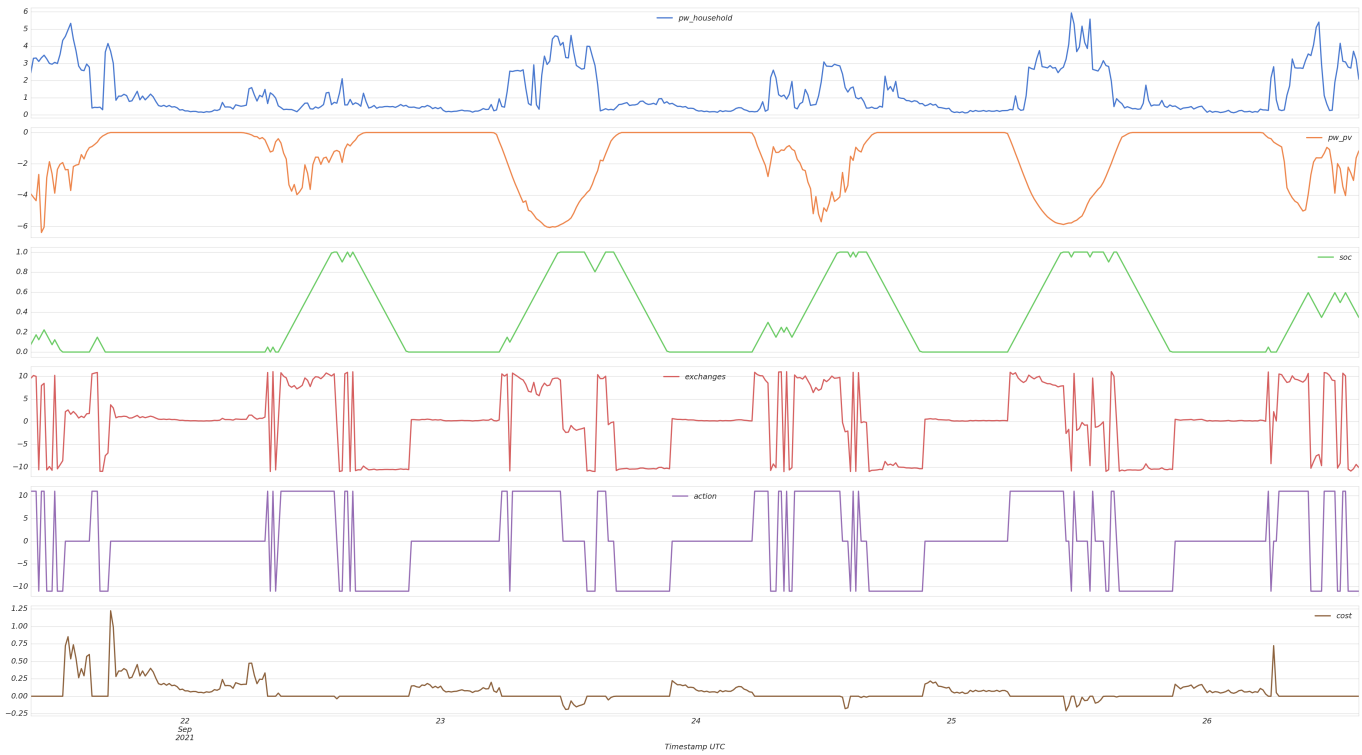
SAC


pw_household
pw_pv
soc
exchange
action
cost
22
Sep
2021
23
24
25
26
Timestamp UTC

TD3



RBC



Total Cost Comparison

Initial comparison with rule based vs rl models(PPO, SAC, TD3) was made on first 500 timesteps of test set as shown in the table below.

Models	**PPO**	**SAC**	**TD3**	**RBC Heuristic**
Total Cost	-163	-194	-185	30

From this naive comparison TD3 is the best cost saving model and RBC is the least cost saving model. Where -ve sign suggests the generated photovoltaic exceeds the household consumption.

Remark: It is calculated for very small set of data, and the RBC implementation needs to be rechecked and verified, since it is generating inverted results.
