Energy and AI 5 (2021) 100092

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.sciencedirect.com/journal/energy-and-ai

Gym-ANM: Reinforcement learning environments for active network management tasks in electricity distribution systems

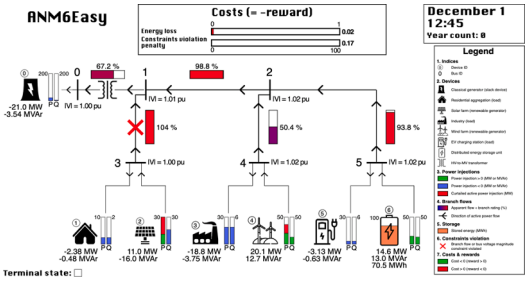
Robin Henry [*,a], Damien Ernst [b]

[a] School of Engineering, Sanderson Building, The University of Edinburgh, Edinburgh EH9 3FB, UK
[b] Department of Electrical Engineering and Computer Science, Montefiore Institute, University of Liége, Liége B-4000, Belgium

HIGHLIGHTS

- Software for training reinforcement learning agents to control distribution grids.
- Provided as customizable Gym Open AI environments.
- Results on a test system suggest RL algorithms are suited for such tasks.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Gym-ANM
Reinforcement learning
Active network management
Distribution networks
Renewable energy

ABSTRACT

Active network management (ANM) of electricity distribution networks include many complex stochastic sequential optimization problems. These problems need to be solved for integrating renewable energies and distributed storage into future electrical grids. In this work, we introduce Gym-ANM, a framework for designing reinforcement learning (RL) environments that model ANM tasks in electricity distribution networks. These environments provide new playgrounds for RL research in the management of electricity networks that do not require an extensive knowledge of the underlying dynamics of such systems. Along with this work, we are releasing an implementation of an introductory toy-environment, ANM6-Easy, designed to emphasize common challenges in ANM. We also show that state-of-the-art RL algorithms can already achieve good performance on ANM6-Easy when compared against a model predictive control (MPC) approach. Finally, we provide guidelines to create new Gym-ANM environments differing in terms of (a) the distribution network topology and parameters, (b) the observation space, (c) the modeling of the stochastic processes present in the system, and (d) a set of hyperparameters influencing the reward signal. Gym-ANM can be downloaded at https://github.com/robinhenry/gym-anm.

1. Introduction

Reinforcement learning (RL) is a vibrant field of machine learning aiming to mimic the human learning process. This allows us to solve numerous complex decision-making problems [1]. In the field of power systems (a term used to refer to the management of electricity

* Corresponding author.
E-mail addresses: robin@robinxhenry.com (R. Henry), dernst@uliege.be (D. Ernst).

https://doi.org/10.1016/j.egyai.2021.100092
Received 14 March 2021; Received in revised form 30 May 2021; Accepted 31 May 2021
Available online 1 June 2021

networks), researchers and engineers have used RL techniques for many years [2]. Over the last few years, however, decision-making challenges in power systems have drawn less attention than other domains in which RL has been successfully and extensively applied, such as the fields of games [3–6], robotics [7–10], and autonomous driving [11–13]. A plausible explanation for this is the lack of off-the-shelf simulators that model such problems. Indeed, despite its many recent breakthroughs, RL research remains largely dependent on the availability of artificial simulators that can be used as surrogates for the real world [14]. Training on real systems is often too slow and constraining, while simulators allow us to take advantage of large computational resources and do not constrain exploration.

Developing efficient and reliable algorithms to solve decision-making challenges in power systems is becoming more and more crucial for ensuring a smooth transition to sustainable energy systems. Power grids have experienced profound structural and operational changes over the last two decades [15]. The liberalization of electricity markets introduced a competitive aspect in their management, driving network improvements and cheaper energy generation [16]. The arrival of distributed generators, such as wind turbines and photovoltaic panels (PVs), has compromised the traditional model of decentralized generation. In particular, we have seen the appearance of (virtual) microgrids creating local energy ecosystems in which consumers are now also producers [17]. In the near future, we can expect the addition of even more distributed generators to the grid [18], along with an increase in the number of large loads due to the fast electric vehicle market growth [19]. Power grids are also facing the emergence of distributed energy storage (DES), with certain technologies already available, such as batteries [20] and power-to-gas [21]. As a result, system operators are facing many new complex decision-making problems (overvoltages, transmission line congestion, voltage coordination, investment issues, etc.), some of which might benefit from advances in the very active area of RL research.

Through this work, we seek to promote the application of RL techniques to active network management (ANM) problems, a class of sequential decision-making tasks in the management of electricity distribution networks (DNs)[22]. In the power system literature, ANM refers to the design of control schemes that modulate the generators, the loads, and/or the DES devices connected to the grid. This is done to avoid problems at the distribution level and maximize profitability through, e.g., avoidable energy loss [23]. This modulation, operated by distribution network operators (DNOs), may result in a necessary reduction in the output of generators from what they could otherwise have produced given available resources, often referred to as the process of curtailment. Such generation curtailment, along with storage and transmission losses, constitute the principal sources of energy loss that we would like to minimize through ANM. At the same time, the ANM scheme must ensure a safe and reliable operation of the DN. This is often expressed as a set of operational constraints that must be satisfied.

More specifically, we propose Gym-ANM, a framework that facilitates the design and the implementation of RL environments that model ANM tasks. Our goal was to release a tool that could be used without an extensive background in power system analysis. We thus engineered Gym-ANM so as to abstract away most of the complex dynamics of power system modeling. With its different customizable components, Gym-ANM is a suitable framework to model a wide range of ANM tasks, from simple ones that can be used for educational purposes, to complex ones designed to conduct advanced research. In addition, Gym-ANM is built on top of the OpenAI Gym toolkit [24], an interface with which a large part of the RL community is already familiar. Note that Gym-ANM environments do not solve ANM problems but, rather, provide a simple programming interface to test and compare various optimization and RL algorithms that aim to do so.

The remainder of this paper is organized as follows. First, we introduce a series of background concepts and notations in RL, DNs, and MPC in Section 2. Section 3 then formalizes the generic ANM task that we consider as a partially observable Markov decision process (POMDP). Next, we propose a specific Gym-ANM environment that highlights common ANM challenges, ANM6-Easy (Fig. 1), in Section 4. The performance of the state-of-the-art proximal policy optimization [25] (PPO) and soft actor-critic [26] (SAC) deep RL algorithms are evaluated on ANM6-Easy in Section 5. Finally, Section 6 concludes our work. To keep this paper accessible to a broad audience and to provide interested readers with a formal introduction to power system modeling, technical details about the inner working of the power grid simulator are gathered in Appendix A and Appendix B. Guidelines to design and implement new Gym-ANM environments are also provided in Appendix C and Appendix D, and more in-depth tutorials and documentation can be found on the project repository at https://github.com/robinhenry/gym-anm.

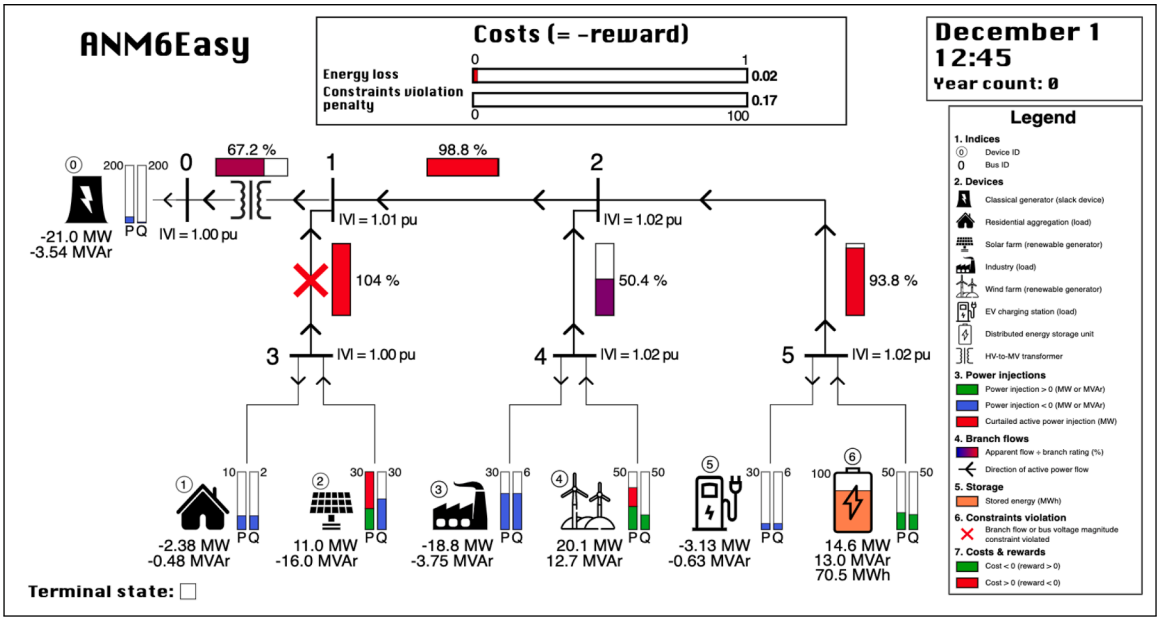


Fig. 1. A Gym-ANM environment. At this specific time, the agent is curtailing both renewable energy resources and discharging the DES unit. Transmission line 1–3 is overheating with a power flow of 104% of its capacity.

```
env = gym.make('MyANMEnv')    # Initialize the environment.
obs = env.reset()             # Reset the env. and collect o_0.

for t in range(1, T):
  env.render()          # Update the rendering.
  a = agent.act(obs)    # The agent takes o_t as input and chooses a_t.
  obs, r, done, info = env.step(a)
        # The action is applied, and are outputted:
        # - obs: the new observation o_{t+1},
        # - r: the reward r(s_t, a_t, s_{t+1}),
        # - done: True if s_{t+1} ∈ S^terminal,
        # - info: extra info about the transition.

env.close()  # Close the environment and stop rendering.
```

Fig. 1. A code snippet (Python 3) illustrating environment-agent interactions.

2. Background

2.1. Reinforcement learning

We consider the standard RL setting for continuing tasks where an agent interacts with an environment E over an infinite sequence of discrete timesteps $\mathcal{T} = \{0, 1, \ldots\}$, modelled as a Markov decision process (MDP). At each timestep t, the agent selects an action $a_t \in \mathcal{A}$ based on a state $s_t \in \mathcal{S}$ according to a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, such that $a_t \sim \pi(\cdot|s_t)$. After the action is applied, the agent transitions to a new state $s_{t+1} \sim p(\cdot|s_t, a_t) \in \mathcal{S}$ and receives the reward $r_t = r(s_t, a_t, s_{t+1}) \in \mathbb{R}$. The return from state s_t is defined as $R_t = \lim_{T\to\infty} \sum_{i=t}^{T-1} \gamma^{i-t} r_t$, where $\gamma \in [0, 1)$ is the discount factor determining the weight of short- versus long-term rewards. We also distinguish a set of terminal states $\mathcal{S}^{terminal} \subset \mathcal{S}$. Given the distribution of initial states $p_0(\cdot)$ and the set of stationary policies Π, a policy $\pi \in \Pi$ is considered optimal if it maximizes the expected return $J_\pi(s_0) = \mathbb{E}_{s_{i>0}, r_{i\geq 0} \sim E, a_i \sim \pi}[R_0|s_0]$ for all s_0 that belong to the support of $p_0(\cdot)$, and where rewards received after reaching a terminal state are always zero. In the context of problems with large and/or continuous state-action spaces, RL often focuses on learning a parameterized policy $\pi_\phi \in \Pi$ with parameters ϕ whose expected return $J_{\pi_\phi}(s_0)$ is as close as possible to that of an optimal policy.

In many cases, the environment may be partially observable so that the agent only has access to observations $o \in \mathcal{O}$. The agent must thus adequately infer, directly or indirectly, an approximation of the state s_t from the history of observation-action-reward tuples $h_t = (o_0, a_0, r_0 \ldots, a_{t-1}, r_{t-1}, o_t)$. We designed the Gym-ANM framework so that it is straightforward for researchers to experiment with different degrees of observability in each environment.

2.2. Distribution networks

An electricity distribution network can be represented as a directed graph $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{0, 1, \cdots, N-1\}$ is a set of positive integers representing the buses (or nodes) in the network, and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of directed edges linking buses together. The notation $e_{ij} \in \mathcal{E}$ refers to the directed edge with sending bus i and receiving bus j. Each bus might be connected to several electrical devices, which may inject into or withdraw power from the grid. The set of all devices is denoted by $\mathcal{D} = \{0, 1, \ldots, D-1\}$, the set of all devices connected to bus $i \in \mathcal{N}$ by $\mathcal{D}_i \subseteq \mathcal{D}$, and it is assumed that each device is connected to a single bus.

Several variables (complex phasors) are associated with each bus $i \in \mathcal{N}$: a bus voltage level V_i, a bus current injection I_i, an active (real) power injection $P_i^{(bus)}$, and a reactive power injection $Q_i^{(bus)}$. The bus power injections $P_i^{(bus)}$ and $Q_i^{(bus)}$ can also be obtained from $P_i^{(bus)} = \sum_{d\in\mathcal{D}_i} P_d^{(dev)}$ and $Q_i^{(bus)} = \sum_{d\in\mathcal{D}_i} Q_d^{(dev)}$, where $P_d^{(dev)}$ and $Q_d^{(dev)}$ denote the active and reactive power injections from device $d \in \mathcal{D}$ into the grid, respectively. The complex powers $S_i^{(bus)}, S_d^{(dev)} \in \mathbb{C}$ injected into the network at bus i, or device d, can then be obtained from the relation $S_i^{(bus)} = P_i^{(bus)} + \mathbf{i}Q_i^{(bus)}$ or $S_d^{(dev)} = P_d^{(dev)} + \mathbf{i}Q_d^{(dev)}$. Similarly, variables I_{ij}, P_{ij}, Q_{ij}, and S_{ij} refer to the directed flow of these quantities in branch $e_{ij} \in \mathcal{E}$, as measured at bus i. Note that, as a result of transmission losses, power and current flows may have different magnitudes at each end of the branch, e.g. $|P_{ij}| \neq |P_{ji}|$.

2.3. Model predictive control (MPC) and optimal power flow (OPF)

In this work, we also present a model predictive control (MPC) approach to solving the ANM tasks that we propose with Gym-ANM. MPC in discrete-time settings is a control strategy in which, based on a known model of the dynamics of the system, a multi-stage optimization problem is solved at each timestep over a finite time horizon. The solution found is applied to the system at the current timestep, and the process is repeated at the next one, indefinitely [27]. The fact that a multi-stage optimization problem based on a model of the system is solved at each time step allows MPC to plan ahead and anticipate the system's behavior. This leads to near-optimal performance as the optimization horizon is increased (assuming an accurate model of the system).

The optimization problem solved by our MPC control algorithm is a multi-stage optimal power flow (OPF) problem. Since its first formulation by Carpentier in 1962 [28], solving a single instance or multiple instances of the OPF problem at regular time intervals has been the dominant approach to tackling decision-making problems in the management of power systems when network constraints are taken into account. In its most general form, the OPF problem is a non-convex constrained optimization problem with equality and inequality constraints. The objective function to minimize is often a representation of network operating costs, the equality constraints model the physical flows of electricity, and the inequality constraints model operational constraints. There exist many different formulations of the OPF problem, each designed to solve a particular control task in power systems. Although many solution methods have been proposed using a wide range of optimization tools and techniques, no single formulation has been accepted as suitable for all forms of OPF problems and it remains an active area of research. For the interested reader, comprehensive surveys of such approaches can be found in [29,30].

3. Gym-ANM

In this section, we propose Gym-ANM, a framework that can model a wide range of novel sequential decision-making ANM tasks to be solved by RL agents. Each Gym-ANM task is provided as a Gym [24] environment E that we describe by the MDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p_0, p, r, \gamma)_E$. Our formalization of these MDPs follows closely, and was inspired by, the work of Gemine et al. in [22].

For mathematical convenience, the set of electrical devices $\mathcal{D}$ con-

nected to the grid is divided into three disjoint subsets $\mathscr{D}_G$, $\mathscr{D}_L$, and $\mathscr{D}_{DES}$, so that $|\mathscr{D}_G| + |\mathscr{D}_L| + |\mathscr{D}_{DES}| = |\mathscr{D}|$. The set $\mathscr{D}_G$ contains the generators, $\mathscr{D}_L$ the loads, and $\mathscr{D}_{DES}$ the DES units. Generators represent devices that only inject power into the grid, such as renewable energy resources (RER) $\mathscr{D}_{RER} \subset \mathscr{D}_G$ or other traditional power plants $\mathscr{D}_G - \mathscr{D}_{RER}$. Loads group the passive devices that only withdraw power from the grid. Storage units, on the other hand, can both inject and withdraw power into/from the network. The only exception is the slack generator $g^{slack} \in \mathscr{D}_G - D_{RER}$, assumed to be the only device connected to the slack bus. The slack bus is a special bus used to balance power flows in the network and provide a voltage reference. The slack bus can also either inject or withdraw power into/from the network, such that the total generation remains equal to the total load plus transmission losses, at all times.

3.1. Overview

The structure of the Gym-ANM framework is illustrated in Fig. 2, in which grey blocks represent components (functions) that are fully customizable by the user to design unique ANM tasks. At each timestep t, the agent receives an observation $o_t \in \mathscr{O}$ and a reward $r_{t-1} \in \mathbb{R}$, based on which it then selects an action $a_t \in \mathscr{A}$ to be applied in the environment.

Once the environment has received the selected action a_t, it samples a series of internal variables using the `next_vars()` generative process conditioned on the current state $s_t \in \mathscr{S}$. These internal variables model the temporal stochastic evolution of the electricity demand and of the maximum renewable energy production before curtailment across the DN, as further described in later sections.

The internal variables are then passed, along with a_t and s_t, to the main function `next_state()`, which applies the action to the environment and outputs the new state $s_{t+1} \in \mathscr{S}$. The `next_state()` block behaves deterministically for a given DN. It first maps the selected action a_t to the current available action space $\mathscr{A}(s_t) \subseteq \mathscr{A}$ before applying it to the environment. All the currents, voltages, energy storage levels, and power flows and injections are then updated, resulting in a new state s_{t+1}. Most of the power system modeling of the environment is handled by the `next_state()` component, which we provide as a built-in part of the framework.

The new state s_{t+1} is then used to compute the new observation $o_{t+1} \in \mathscr{O}$ and reward $r_t \in \mathbb{R}$. Much like the `next_vars()` block, the behavior of the `observation()` component can be freely designed by the designer of the environment. This way, it becomes straightforward to investigate the impact of different observation vectors on the performance of a given algorithm on a given ANM task. To simplify the use of our framework, we also provide a set of default common observation spaces that researchers can experiment with.

Our framework provides a built-in `reward()` component that computes the reward r_t as:

$$r_t = clip\big(-r_{clip}, -(\Delta E_{t:t+1} + \lambda\phi(s_{t+1})), r_{clip}\big), \tag{3.1}$$

where $\Delta E_{t:t+1}$ is the total energy loss during $(t, t+1]$, $\phi(s_{t+1})$ is a penalty term associated with the violation of operating constraints, λ is a weighting hyperparameter, and $r_{clip} > 0$ keeps the rewards within a finite range $[-r_{clip}, r_{clip}]$. This reward function was designed to reflect the overall goal: learn a control policy π that ensures a secure operation of the DN while minimizing its operating costs. In the management of real-world DNs, there are many varied sources of operating costs. For simplicity, however, we consider energy losses and the violation of operational constraints to be the only sources of costs. Our reward formulation also assumes that the action is selected by the agent at time t, immediately applied in the environment at time $t + \epsilon$, with $\epsilon \to 0$, and that all power injections remain constant during $(t + \epsilon, t + 1]$.

The Gym-ANM framework allows for the creation of environments that model highly customizable ANM tasks. In particular, varying any of the following components will result in a different MDP, and therefore a different ANM task:

1. **Topology and characteristics of the DN.** Its topology is described by the tuple $(\mathscr{D}, \mathscr{N}, \mathscr{E})$ and its characteristics refer to the parameters of each of its device $d \in \mathscr{D}$, bus $i \in \mathscr{N}$, and transmission link $e_{ij} \in \mathscr{E}$. In particular, the number of devices $|\mathscr{D}|$ and their respective operating range will shape the resulting state space $\mathscr{S}$ and action space $\mathscr{A}$. A detailed list of all the DN parameters modelled in Gym-ANM is provided in Appendix D.
2. **Stochastic processes.** This corresponds to the design of the `next_vars()` component in Fig. 2. This component must model the temporal evolution of the electricity demand $P^{(dev)}_{l,t}$ of each load $l \in \mathscr{D}_L$, the maximum production $P^{(max)}_{g,t}$ that each generator $g \in \mathscr{D}_G - \{g^{slack}\}$ could produce at time t (before curtailment is applied if $g \in \mathscr{D}_{RER}$), and a set of K auxiliary variables $\{aux^{(k)}_t\}_{k=0}^{K-1}$.
3. **Observation space.** The observation space $\mathscr{O}$ can be changed to make the task more or less challenging for the agent by modifying the `observation()` function.
4. **Hyperparameters.** Although the `reward()` component is built-in as a part of the Gym-ANM framework, it nonetheless relies on three hyperparameters that can be chosen for each new task: the penalty weighting hyperparameter λ, the amount of time Δt (in fraction of hour) elapsed between subsequent discretization

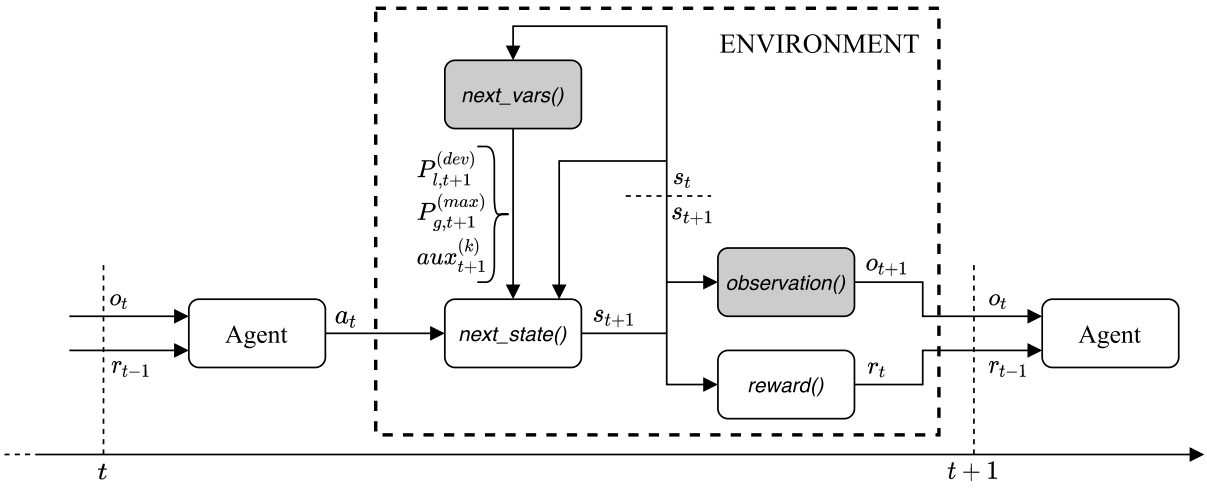


Fig. 2. The Gym-ANM framework.

timesteps, and the clipping hyperparameter r_{clip}. Because we consider a policy to be optimal if it minimizes the expected sum of discounted costs, we also consider the discount factor $\gamma \in [0, 1)$ to be another fixed hyperparameter part of the task description.

In the remainder of this section, we explore the resulting MDP in more detail.

3.2. State space

At any timestep t, the state of a Gym-ANM environment is fully described by the state of the DN that it models. We represent this state using a set of state variables aggregated into a vector $s_t \in \mathcal{S}$:

$$s_t = [\ \left\{P^{(dev)}_{d,t}\right\}_{d\in\mathcal{D}},\ \left\{Q^{(dev)}_{d,t}\right\}_{d\in\mathcal{D}},\ \left\{SoC_{d,t}\right\}_{d\in\mathcal{D}_{DES}},\ \left\{P^{(max)}_{g,t}\right\}_{g\in\mathcal{D}_G-\{g^{slack}\}},\ \left\{aux^{(k)}_t\right\}_{k=0}^{K-1}\ ], \tag{3.2}$$

where

- $P^{(dev)}_{d,t}$ and $Q^{(dev)}_{d,t}$ refer to the active and reactive power injections of device $d \in \mathcal{D}$ into the grid, respectively,
- $SoC_{d,t}$ is the charge level, or state of charge (SoC), of DES unit $d \in \mathcal{D}_{DES}$,
- $P^{(max)}_{g,t}$ is the maximum production that generator $g \in \mathcal{D}_G - \{g^{slack}\}$ can produce,
- $aux^{(k)}_t$ is the value of the $(k-1)^{\text{th}}$ auxiliary variable generated by the `next_vars()` block during the transition from timestep t to timestep $t+1$.

In (3.2), the first $2|\mathcal{D}| + |\mathcal{D}_{DES}|$ variables $(P^{(dev)}_{0,t}, \ldots, SoC_{|\mathcal{D}_{DES}|-1,t})$ can be used to compute any other electrical quantities of interest in the DN (i.e., currents, voltages, power flows and injections, and energy storage levels), as derived in Appendix A. We also include the maximum generation variables $P^{(max)}_{g,t}$ in s_t because, even though they do not affect the physical electric flows in the network, they are required to compute the reward signal (see Section 3.6).

These variables do not, however, provide any information about the temporal behavior of the system. Hence, they are not sufficient to describe the full state of the system from a Markovian perspective. For instance, it may not be enough to know the active and reactive power injections from a load $l \in \mathcal{D}_L$ at time t to fully describe the probability distribution of its next demand $P^{(dev)}_{l,t+1}$.

In order to make s_t Markovian, we chose to include a set of K auxiliary variables $\{aux^{(k)}_t\}_{k=0}^{K-1}$ that can be used to model other temporal factors that influence the outcomes $P^{(dev)}_{l,t+1}$ and $P^{(max)}_{g,t+1}$ during the `next_vars()` call of Fig. 2. This leads to state transitions that are only conditioned on the current state of the environment and on the action the agent selects, i.e., $s_{t+1} \sim p(\cdot|s_t, a_t)$. The overall task is thus indeed a MDP.

For example, the environment ANM6-Easy that we introduce in Section 4 uses a single auxiliary variable that represents the time of the day. This is sufficient to make s_t Markovian, since the underlying stochastic processes can all be expressed as a function of the time of day. Another example would be an environment in which the next demand of each load and the generation from each generator is solely dependent on their current value. In this case, s_t would not require any extra auxiliary variables. As environments become more and more complex, we expect state vectors to contain many auxiliary variables. Such examples could include solar irradiation and wind speed information to better represent the evolution of the electricity produced by renewable energy resources.

Finally, the environment may also reach a terminal state $s_t \in \mathcal{S}^{terminal}$, indicating that no solution to the power flow equations (see Appendix A.5) was found as a result of the action taken by the agent. This means that the power grid has collapsed and is often due to a voltage collapse problem [31].

3.3. Action space

Given the current state of the environment $s_t \in \mathcal{S}$, the available actions are denoted by the action space $\mathcal{A}(s_t)$. We define an action vector $a_t \in \mathcal{A}(s_t)$ as:

$$a_t = [\ \left\{a_{P_{g,t}}\right\}_{g\in\mathcal{D}_G-\{g^{slack}\}},\ \left\{a_{Q_{g,t}}\right\}_{g\in\mathcal{D}_G-\{g^{slack}\}},\ \left\{a_{P_{d,t}}\right\}_{d\in\mathcal{D}_{DES}},\ \left\{a_{Q_{d,t}}\right\}_{d\in\mathcal{D}_{DES}}\ ], \tag{3.3}$$

for a total of $N_a = 2|\mathcal{D}_G| + 2|\mathcal{D}_{DES}| - 2$ control variables to be chosen by the agent at each timestep. Each control variable belongs to one of four categories:

- $a_{P_{g,t}}$: an upper limit on the active power injection from generator $g \in \mathcal{D}_G - \{g^{slack}\}$. If $g \in \mathcal{D}_{DER}$, then $a_{P_{g,t}}$ is the curtailment value. For classical generators, it simply refers to a set-point chosen by the agent. The slack generator is excluded, since it is used to balance load and generation and, as a result, its power injection cannot be controlled by the agent. That is, g^{slack} will inject the amount of power needed to fill the gap between the total generation and demand into the network.
- $a_{Q_{g,t}}$: the reactive power injection from each generator $g \in \mathcal{D}_G - \{g^{slack}\}$. Again, the injection from the slack generator is used to balance reactive power flows and cannot be controlled by the agent.
- $a_{P_{d,t}}$: the active power injection from each DES unit $d \in \mathcal{D}_{DES}$.
- $a_{Q_{d,t}}$: the reactive power injection from each DES unit $d \in \mathcal{D}_{DES}$.

The resulting action space $\mathcal{A}(s_t)$ is bounded by three sets of constraints. First, individual control variables in $a_t \in \mathcal{A}(s_t)$ are restricted to finite ranges $\left[\underline{P}, \overline{P}\right]$ or $\left[\underline{Q}, \overline{Q}\right]$. This is because electrical devices cannot physically inject (withdraw) infinite active or reactive power into (from) the network. Second, generators and DES units may have additional constraints on their current injections, such as current limits of power converters. These constraints further restrict the range of (P, Q) injection points that these devices can apply, i.e. they cannot simultaneously operate at full capacity for both active and reactive power. Third, the range of possible active power injection from each DES unit depends on its current storage level (provided in s_t). Indeed, empty (full) units cannot inject (withdraw) any power into (from) the network. Note that the first two sets of constraints remain the same for all $s_t \in \mathcal{S}$ (see Appendix A.3).

For simplicity, the agent is never given the precise boundaries of the action space $\mathcal{A}(s_t)$. Instead, we let it choose an action within a larger set $\mathcal{A}$ bounded only by the first set of constraints, i.e. $\mathcal{A}$ ignores current limits in generators and DES units, as well as storage levels. In the case where the agent selects an action $a_t \in \mathcal{A}$ that falls outside of the current action space $\mathcal{A}(s_t)$, the action that is actually applied in the environment during the `next_state()` call is the action in $\mathcal{A}(s_t)$ that stands the closest to a_t, according to the Euclidean distance (see Appendix A.6).

As a result, $\mathcal{A}$ is always bounded. Its bounds can be retrieved by the agent through the built-in `action_space()` function. This allows users to follow good practices by working with agents that generate normalized action vectors in $[-1, 1]^{N_a}$.

3.4. Observation space

In general, DNOs rarely have access to the full state of the distribution network when doing ANM. To model these real-world scenarios, Gym-ANM allows the design of a unique observation space $\mathcal{O}$ through

the implementation of the `observation()` component, which may result in a partially observable task. We only assume that the size of o_t remains constant.

To simplify the design of customized observation spaces, Gym-ANM also allows researchers to simply specify a set of variables to include in the observation vectors (e.g., branch active power flows $\{P_{12}, P_{23}\}$ and bus voltage magnitudes $\{|V_0|, |V_2|\}$) of the new environment. The full list of available variables from which to choose is given in Appendix C.

The agent can access the bounds of the observation space through the function call `observation_space()`. This functionality may be of particular interest to agents that use neural networks to learn ANM policies, in which case normalized input vectors may increase training speed and stability.

3.5. Transition function

Each state transition occurs in two steps. First, the outcomes of the internal stochastic variables $\{P^{(dev)}_{l,t+1}\}_{l\in\mathscr{D}_L}$, $\{P^{(max)}_{g,t+1}\}_{g\in\mathscr{D}_G-\{g^{slack}\}}$, and $\{aux^{(k)}_{t+1}\}_{k=0}^{K-1}$ are generated by the `next_vars()` block of the Gym-ANM framework (see Fig. 2). Once the selected action $a_t \in \mathscr{A}$ has been passed to the environment, the remainder of the transition is handled by the `next_state()` component in a deterministic way. The reactive power injection of each load $d \in \mathscr{D}$ is directly inferred from its active power injection (assuming a constant power factor). The action a_t is then mapped to $\mathscr{A}(s_t)$ according to the Euclidean distance and applied in the environment. Finally, all electrical quantities are updated by solving a set of so-called network equations (see Appendix A.5). The computational steps taken by `next_state()` are described in more detail in Appendix A.6.

3.6. Reward function

The reward signal is implemented by the built-in `reward()` block of Fig. 2 and is given by:

$$r_t = \begin{cases} clip\big(-r_{clip}, c_t, r_{clip}\big), & \text{if } s_{t+1} \notin \mathscr{S}^{terminal}, \\ -\dfrac{r_{clip}}{1-\gamma}, & \text{if } s_t \notin \mathscr{S}^{terminal} \text{ and } s_{t+1} \in \mathscr{S}^{terminal}, \\ 0, & \text{else,} \end{cases} \tag{3.4}$$

where

$$c_t = -(\Delta E_{t:t+1} + \lambda\phi(s_{t+1})). \tag{3.5}$$

Using a reward clipping parameter r_{clip} ensures that any transition from a non-terminal state to a terminal one (i.e., when the power grid collapses), generates a much larger reward than any other transition does. As a result, it encourages the agent to learn a policy that avoids such scenarios at all costs. Subsequent rewards are always zero, until a new trajectory is started by sampling a new initial state s_0.

During all other transitions, the energy loss $\Delta E_{t:t+1}$ is computed in three parts:

$$\Delta E_{t:t+1} = \Delta E^{(1)}_{t:t+1} + \Delta E^{(2)}_{t:t+1} + \Delta E^{(3)}_{t:t+1}\ , \tag{3.6}$$

where:

- $\Delta E^{(1)}_{t:t+1}$ is the total transmission energy loss during $(t, t+1]$. This is a result of leakage in transmission lines and transformers.
- $\Delta E^{(2)}_{t:t+1}$ is the total net amount of energy flowing from the grid into DES units during $(t, t+1]$. Over a sufficiently large number of timesteps, the sum of these terms will approximate the amount of energy lost due to leakage in DES units. That is, taking an energy of ΔE from the grid using a DES unit $d \in \mathscr{D}_{DES}$ will yield a cost of ΔE. Given a charging and discharging efficiency factor of η_d for d, injecting the remaining energy after a total round-trip loss will result in a cost of $-\eta^2\Delta E$, totalling a round-trip cost of $(1-\eta^2)\Delta E$. This is the total energy loss over the round-trip.
- $\Delta E^{(3)}_{t:t+1}$ is the total amount of energy loss as a result of renewable generation curtailment of generators $\mathscr{D}_{RER}$ during $(t, t+1]$. Depending on the regulation, this can be thought of as a fee paid by the DNO to the owners of the generators that get curtailed, as financial compensation.

In the penalty term $\phi(s_{t+1})$, we consider two types of network-wide operating constraints. The first is the limit on the amount of power[1] that can flow through a transmission link $e_{ij} \in \mathscr{E}$, referred to as the rating of that link. These constraints are needed to prevent lines and transformers from overheating. The second type of constraint is a limit on the allowed voltage magnitude $|V_i|$ at each bus $i \in \mathscr{N}$. The latter are necessary conditions to maintain stability throughout the network and ensure proper operation of devices connected to the grid.

In practice, violating any network constraint can lead to damaging parts of the DN infrastructure (e.g., lines or transformers) or power outages. Both can have important economic consequences for the DNO. For that reason, ensuring that the DN operates within its constraints is often prioritized compared to minimizing energy loss. Although our choice of reward function does not guarantee that an optimal policy will never violate these constraints, choosing a large λ will ensure that these violations remain small. This would, in practice, have a negligible impact on the operation of the DN. In addition, the risk of violating real-life constraints in the DN could be further reduced by setting an over-restrictive set of constraints in the environment.

The technical details behind the computation of r_t can be found in Appendix A.7.

3.7. Model predictive control scheme

In order to quantify how well an agent is performing on a specific Gym-ANM task, we can cast the task as a MPC problem in which a multi-stage (N-stage) OPF problem is solved at each timestep. The resulting policy provides us with a loose lower bound on the best performance achievable in the environment.

The general MPC algorithm that we provide takes as input forecasts of demand for each load $l \in \mathscr{D}_L$ and of maximum generation for each non-slack generator $g \in \mathscr{D}_G - \{g^{slack}\}$ over the optimization horizon $[t+1, t+N]$. We refer to the resulting policy as π_{MPC-N}. We then consider two variants: policies $\pi^{constant}_{MPC-N}$ and $\pi^{perfect}_{MPC-N}$. The former, $\pi^{constant}_{MPC-N}$, uses constant forecasts over the optimization horizon. Its simplicity means that it can be used in any Gym-ANM environment[2] The other variant, $\pi^{perfect}_{MPC-N}$, assumes perfect predictions of future demand and generation are available for planning. Although it can only be used in simple environments such as ANM6-Easy (see Section 4.2), its performance is superior to that of $\pi^{constant}_{MPC-N}$. This means it provides the user with a tighter lower bound on the best achievable performance. Both variants are formally described in Appendix B.

Both MPC-based control schemes model the power grid using the DC power flow equations, a linearized version of the AC power flow equations. They thus solve a multi-stage DCOPF problem at each timestep. The DCOPF formulation relies on three assumptions: (a) transmission lines are lossless, (b) the difference between adjacent bus voltage angles is small, and (c) bus voltage magnitudes are close to unity.

It is worth stressing that the MPC method that we propose here is an example of a traditional approach to tackling ANM problems. Because

[1] In the literature, these limits are sometimes described in terms of current flows, instead of power flows.

[2] See the project repository for more information.

RL algorithms make less assumptions about the intrinsic structure of the problem, however, they have the potential to overcome the limitations of such optimization approaches and reach better solutions. This, of course, does not mean that RL should be blindly applied to most multi-step OPF-like problems, but, rather, that it might prove to be a good alternative when traditional approaches reach their limitations. This remains a hypothesis which, we hope, Gym-ANM will help confirm or deny.

4. Environments

4.1. Gym-ANM environments.

In conformity with the Gym framework, any Gym-ANM environment provides four main functions that allow the agent to interact with it: `reset()`, `step(action)`, `render()`, and `close()`. An example of code illustrating the interactions between an agent `agent` and an environment `env` is shown in Listing 1 (inspired from [24]). The agent-learning procedure is omitted for clarity. Guidelines to design and implement new Gym-ANM environments can be found in Appendix C.

4.2. ANM6-Easy.

Along with this paper we are also releasing ANM6-Easy, a Gym-ANM environment that models a series of ANM characteristic problems. ANM6-Easy is built around a DN consisting of six buses, with one high-voltage to low-voltage transformer, connected to a total of three passive loads, two renewable energy generators, one DES unit, and one fossil fuel generator used as slack generator. The topology of the network is shown in Fig. 1 and its technical characteristics are summarized in Appendix E. We use a time discretization of $\Delta t = 0.25$ (i.e., 15 minutes) by analogy with the typical duration of a market period, much like the work of [22]. The `observation()` component is the identity function. This leads to a fully observable environment with $o_t = s_t$. The discount factor is fixed to $\gamma = 0.995$, the reward penalty to $\lambda = 10^3$, and the reward clipping value to $r_{clip} = 100$.

In order to limit the complexity of the task, we also chose to make the processes generated by the `next_vars()` block deterministic. To do so, we use a fixed 24-hour time series that repeats every day, indefinitely. A single auxiliary variable $aux_t^{(0)} = (T_0 + t) \bmod \frac{24}{\Delta t}$ representing the time of day is used to index the time series, where $T_0 \in \left\{0, 1, \ldots, \frac{24}{\Delta t} - 1\right\}$ is the starting timestamp of the trajectory. During each timestep transition, the `next_vars()` function thus behaves as described by Algorithm 1, where $\boldsymbol{P}_l\left[0, \ldots, \frac{24}{\Delta t} - 1\right]$ and $\boldsymbol{P}_g\left[0, \ldots, \frac{24}{\Delta t} - 1\right]$ are the fixed daily time series of load injections $P_{l,t}^{(dev)}$ and maximum generations $P_{g,t}^{(max)}$, respectively. The initialization procedure of the environment is also provided in Appendix E.

The daily patterns were engineered so as to produce three problematic situations in the DN. Figs. 3, 4, and 5 show the power injections, power flows, and voltage levels that would result in each situation if the agent neither curtailed the renewable energies nor used the DES unit. Each situation lasts for seven, three, and three hours, respectively, during which the power injections remain constant. A two-hour-long period is used to transition between situations, during which each power injection is linearly incremented from its old to new value.

$aux_{t+1}^{(0)} \leftarrow (aux_t^{(0)} + 1) \mod \frac{24}{\Delta t}$
for $l \in \mathcal{D}_L$ **do**
  $P_{l,t+1}^{(dev)} \leftarrow \boldsymbol{P}_l[aux_{t+1}^{(0)}]$
end for
for $g \in \mathcal{D}_G - \{g^{slack}\}$ **do**
  $P_{g,t+1}^{(max)} \leftarrow \boldsymbol{P}_g[aux_{t+1}^{(0)}]$
end for

Algorithm 1. Implementation of `next_vars()` in ANM6-Easy.

Situation 1 This situation (Fig. 3) characterizes a windy night, when the consumption is low, the PV production null, and the wind production at its near maximum. Due to the very low demand from the industrial load, the wind production must be curtailed to avoid an overheating of the transmission lines connecting buses 0 and 4. This is also a period during which the agent might use this extra generation to charge the DES unit in order to prepare to meet the large morning demand from the EV charging garage (see Situation 2).

Situation 2 In this situation (Fig. 4), bus 5 is experiencing a substantial demand due to a large number of EVs being plugged-in at around the same time. This could happen in a large public EV charging garage. In the morning, workers of close-by companies would plug in their car after arriving at work and, in the evening, residents of the area would plug in their cars after getting home. In order to emphasize the problems arising from this large localized demand, we assume that the other buses (3 and 4) inject or withdraw very little power into/from the network. During those periods of the day, the DES unit must provide enough power to ensure that the transmission path from bus 0 to bus 5 is not over-rated, which would lead to an overheating of the line. For this to be possible, the agent must strategically plan ahead to ensure a sufficient charge level at the DES unit.

Situation 3 Situation 3 (Fig. 5) represents a scenario that might occur in the middle of a sunny windy weekday. No one is home to consume the solar energy produced by residential PVs at bus 1 and the wind energy production exceeds the industrial demand at bus 2. In this case, both renewable generators should be adequately curtailed while again storing some of the extra energy to anticipate the EV late afternoon charging period, as depicted in Situation 2.

5. Experiments

In this section, we illustrate the use of the Gym-ANM framework. We compare the performance of PPO and SAC, two model-free deep RL algorithms, against that of the MPC-based policies $\pi_{MPC-N}^{constant}$ and $\pi_{MPC-N}^{perfect}$ introduced in Section 3.7 on the ANM6-Easy task. For both algorithms, we used the implementations from Stable Baselines 3 [32], a popular library of RL algorithms. Since our goal was not to compute an excellent approximation of an optimal policy, but rather to show that existing RL algorithms can already yield good performance with very little hyperparameter tuning, most hyperparameters were set to their default value (see Appendix F). The code used for all experiments in this section can be found at https://github.com/robinhenry/gym-anm-exp.

5.1. Algorithms

Proximal Policy Optimization PPO is a stable and effective on-policy policy gradient algorithm. It alternates between collecting experience, in the form of finite-length trajectories starting from states $s_0 \sim p_0(\cdot)$ and following the current policy, and performing several epochs of optimization on the collected data to update the current policy (after which the collected experience is discarded). During each policy update step, the policy parameters θ are updated by maximizing (e.g., stochastic gradient ascent) a clipped objective function characterized by a hyperparameter ϵ that dictates how far away the new policy π_θ is allowed to diverge from the old $\pi_{\theta_{old}}$. The objective also requires the use of an advantage-function estimator, which is achieved using a learned-state value function $V_\phi(s)$. In the Stable Baselines 3 implementation that we used, both the policy π_θ and the state-value function $V_\phi(s)$ were represented using separate fully connected MLPs with weights θ and ϕ, respectively, each with two layers of 64 units and tanh nonlinearities.

Soft Actor-Critic SAC is an off-policy actor-critic algorithm based on the maximum entropy RL framework. The policy is trained to maximize a

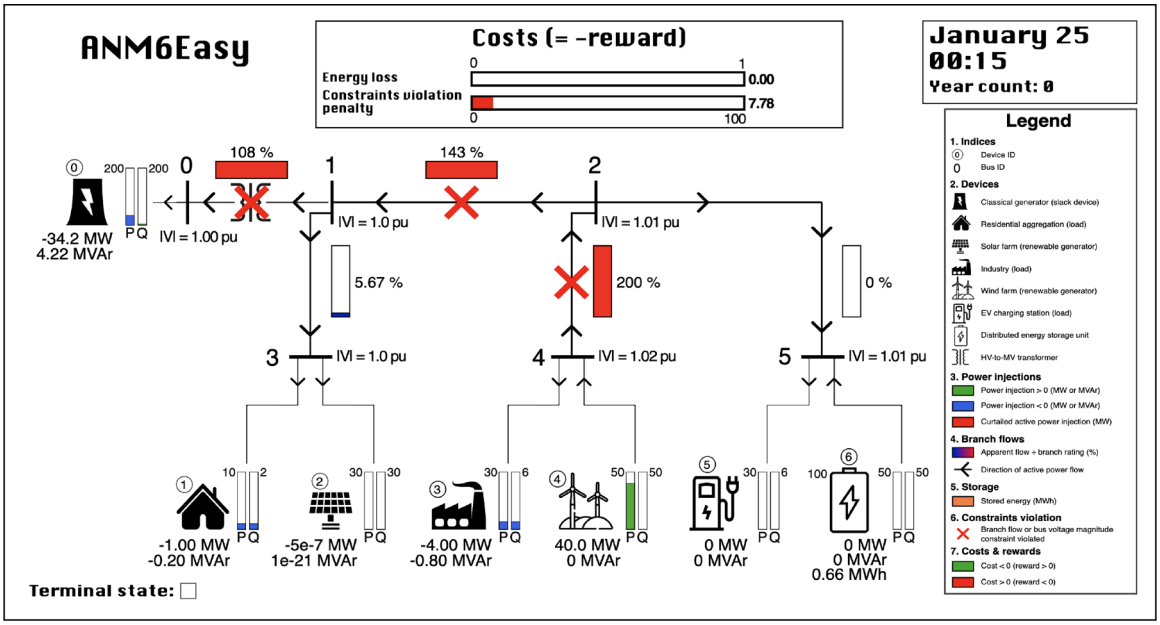


Fig. 3. Situation 1, lasting between 11:00 p.m. and 06:00 a.m. every day.

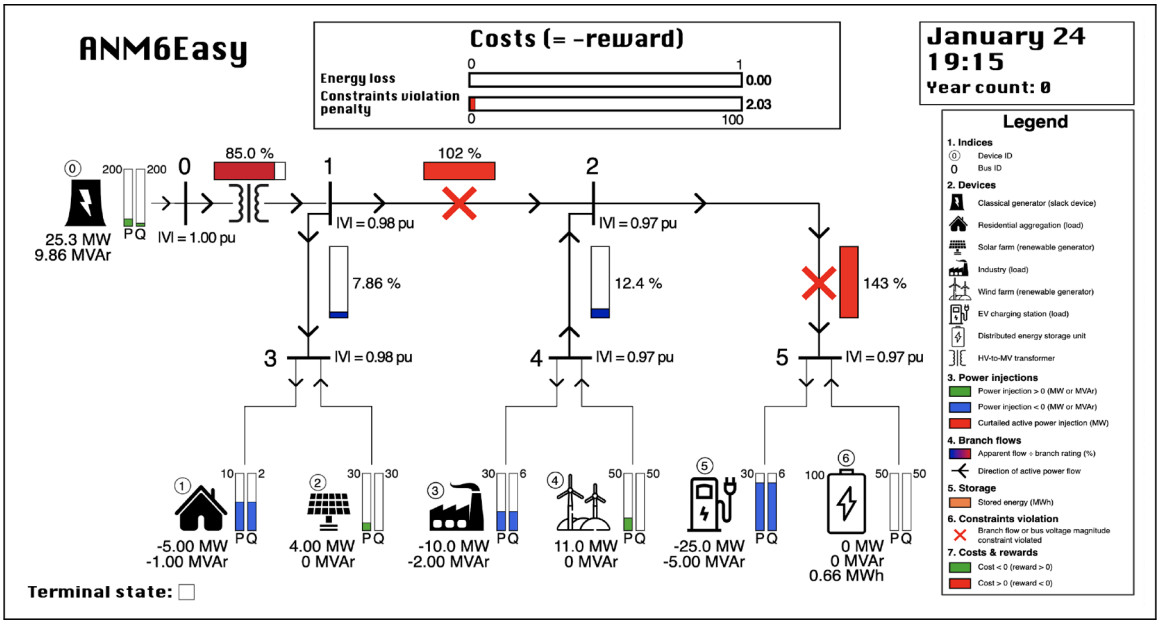


Fig. 4. Situation 2, lasting between 08:00 a.m. and 11:00 a.m. and between 06:00 p.m. and 09:00 p.m. every day.

trade-off between expected return and entropy, a measure of randomness in the policy. It alternates between collecting and storing experience of the form (s_t, a_t, r_t, s_{t+1}) into a replay buffer, regularly ending the current trajectory to start from a new initial state $s_0 \sim p_0(\cdot)$, and updating the policy π_θ (actor) and a soft Q-function $Q_\phi(s_t, a_t)$ (critic) from batches sampled from the replay buffer (e.g., stochastic gradient descent), in an offline manner. In the same manner as the work of Haarnoja et al. [26], the implementation that we used makes use of two Q-functions to mitigate positive bias in the policy improvement step. Both the policy π_θ and the Q-functions Q_{ϕ_1}, Q_{ϕ_2} were represented using separate fully connected MLPs with weights θ, ϕ_1, and ϕ_2, respectively, each with two layers of 64 units and ReLU nonlinearities. Separate target Q-networks that slowly track Q_{ϕ_1}, Q_{ϕ_2} were also used to improve stability, using an exponentially moving average with smoothing constant τ.

5.2. Performance metric

We evaluate the performance of the different algorithms on the ANM6-Easy task as follows. Every N_{eval} steps the agent takes in the environment (i.e., selects an action), we freeze the training procedure and evaluate the current policy on another instance of the environment. To do so, we collect N_r rollouts of T timesteps each, using the current policy π_θ, and report:

$$J_{\pi_\theta} = \mathbb{E}_{s_0 \sim p_0(\cdot)}[J_{\pi_\theta}(s_0)] \approx \frac{1}{N_r} \sum_{i=1}^{N_r} \sum_{t=0}^{T-1} \gamma^t r_t^{(i)}, \tag{5.1}$$

where $s_0^{(i)} \sim p_0(\cdot)$ and $r_t^{(i)}$ are the initial state and rewards obtained in the i^{th} rollout, respectively. Because the reward signal is bounded by a finite

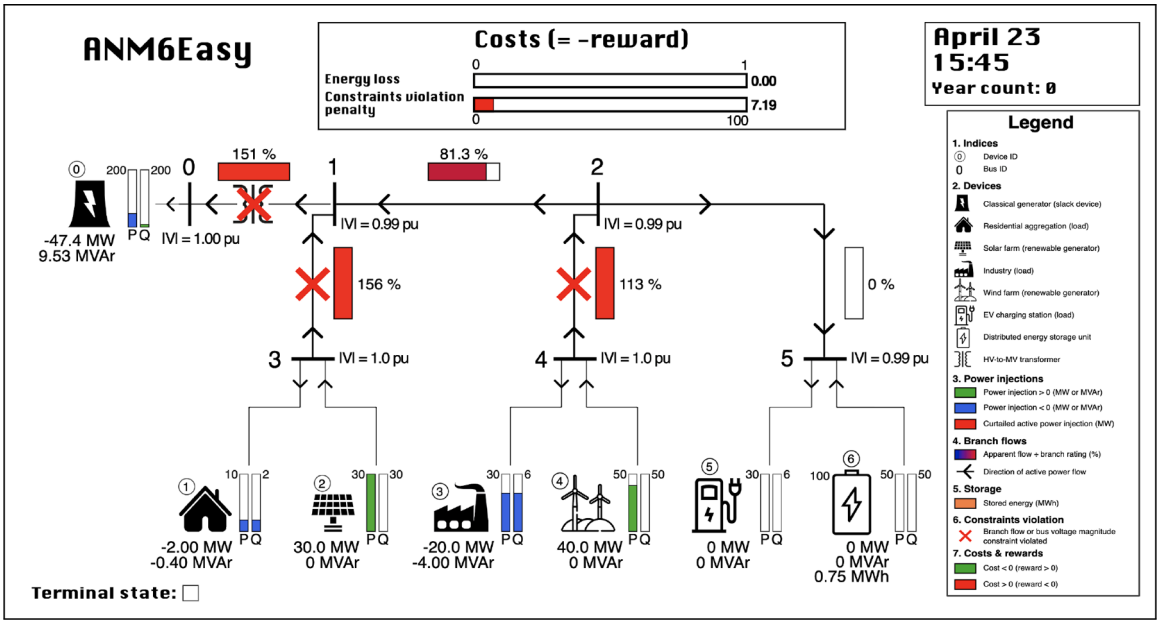


Fig. 5. Situation 3, lasting between 01:00 p.m. and 04:00 p.m. every day.

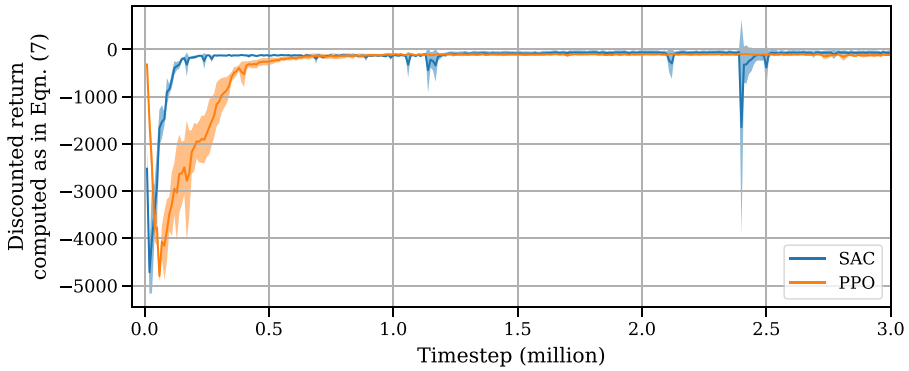


Fig. 6. Evolution of the empirical discounted return J_{π_θ} ($T = 3000$) during training.

constant $r_{clip} \in \mathbb{R}$ (i.e., $|r_t| \in [-r_{clip}, r_{clip}]$, $\forall t$), approximating $J_{\pi_\theta}(s_0) = \lim_{T\to\infty} \sum_{t=0}^{T-1} \gamma^t r_t$ by $\sum_{t=0}^{T-1} \gamma^t r_t$ may result in a deviation of up to $r_{clip}\frac{\gamma^T}{1-\gamma}$ from the true infinite discounted return, since:

$$J_{\pi_\theta}(s_0) \leq r_{clip}\frac{1}{1-\gamma} \quad \text{and} \quad \sum_{t=0}^{T-1} \gamma^t r_t \leq r_{clip}\left(\frac{1}{1-\gamma} - \frac{\gamma^T}{1-\gamma}\right). \tag{5.2}$$

In our experiments, we used $N_r = 5$ and set $T = 3000$, such that $r_{clip}\frac{\gamma^T}{1-\gamma} < 10^{-2}$ results in negligible error terms.

5.3. Results

We trained both the PPO and SAC algorithms on the ANM6-Easy environment for three million steps, starting from a new initial state $s_0 \sim p_0(\cdot)$ every 5000 steps (or earlier if a terminal state is reached), and evaluated their performance every $N_{eval} = 10^4$ steps. Both algorithms used normalized observation and action vectors. We repeated the same procedure with 5 random seeds and plotted the mean and standard deviation of the evolution of their performance during training in Fig. 6.

Table 1 reports the average performance of policies $\pi^{constant}_{MPC-N}$ and $\pi^{perfect}_{MPC-N}$ for different planning steps N and safety margin hyperparameters β (see Appendix B). As expected, the performance of $\pi^{perfect}_{MPC-N}$ increases with N, since the algorithm has access to perfect demand and generation forecasts. In the case of $\pi^{constant}_{MPC-N}$, the best average return is capped at 129.1 and increasing N does not improve performance.

Table 2 compares the best performance of the trained agents against that of the MPC policies. Note that both RL agents reach better performances than $\pi^{constant}_{MPC-N}$. That is, both PPO and SAC outperform a MPC-based policy in which future demand and generation are assumed constant.

Table 1
Average discounted returns J_{π_θ} for $\pi^{constant}_{MPC-N}$ (left) and $\pi^{perfect}_{MPC-N}$ (right), for different planning horizons N and safety margin hyperparameters β.

$\beta \backslash N$	8	16	32	
0.92	**-129.1**	**-129.1**	**-129.1**	
0.94	-129.3	-129.3	-129.2	
0.96	-129.6	-129.5	-129.5	
0.98	-130.5	-130.5	-130.5	
1	-134.8	-134.7	-134.7	
β N	8	16	32	64
0.92	-100.6	-60.3	-16.0	-16.0
0.94	-99.7	-58.2	**-14.7**	**-14.7**
0.96	-102.1	-57.5	-14.8	-14.8
0.98	-102.4	-59.6	-19.0	-19.0
1	-108.0	-68.5	-29.1	-29.1

Table 2
Top row: mean and standard deviation of the best discounted returns over 5 random seeds. Bottom row: mean and standard deviation of the CPU time required to select an action on a MacBook 2.3 GHz Intel Core i5 with 8GB of RAM. .

	PPO	SAC	$\pi^{constant}_{MPC-16}$	$\pi^{perfect}_{MPC-32}$
J_{π_θ}	-93.6 ± 15.3	-56.1 ± 26.8	-129.1 ± 0.4	-14.7 ± 0.2
Time (ms)	0.47 ± 0.19	0.52 ± 0.28	31.60 ± 30.38	61.75 ± 31.21

Finally, Table 2 also summarizes computational CPU times required for each control policy to select an action on a MacBook 2.3 GHz Intel Core i5 with 8GB of RAM. Clearly, RL policies have the advantage of requiring significantly less time for action selection, since the mapping from state (or observation) to action is stored in the form of function approximators, which can be efficiently evaluated. Nevertheless, the learning of these function approximators may require significant computational times, which vary greatly between different RL algorithms.

6. Conclusion

In this paper, we proposed Gym-ANM, a framework for designing and implementing RL environments that model ANM problems in electricity distribution networks. We also introduced ANM6-Easy, a particular instance of such environments that highlights common challenges in ANM. Finally, we showed that state-of-the-art RL algorithms can already reach performances similar to that of MPC-based policies that solve multi-stage DCOPF problems, with little hyperparameter tuning.

We hope that our work will inspire others in the RL community to tackle decision-making problems in electricity networks, potentially through the use of our framework. We believe that Gym-ANM has the potential to model tasks of a wide range of complexity, creating a novel extensive playground for advanced RL research.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. noindent *Funding:* This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Acknowledgements

We would like to thank Raphael Fonteneau, Quentin Gemine, and Sébastien Mathieu at the University of Liége for their valuable early feedback and advice, as well as Gaspard Lambrechts and Bardhyl Miftari for the feedback they provided as the first users of Gym-ANM.

Appendix A. Electricity Distribution Network Simulator

This appendix describes in more detail the dynamics of the alternative current (AC) power grid on top of which Gym-ANM environments are built. Section A.1 introduces some technical power system notions used in later analyses. Sections A.2, A.3.1, A.3.2, and A.3.3 describe the mathematical model and assumptions used to simulate the behavior of transmission links, passive loads, distributed generators, and DES units, respectively. Section A.4 then introduces the set of network constraints that we would like the learned ANM control scheme to satisfy, and Section A.5 derives the set of equations that govern the network electricity flows. Finally, Sections A.6 and A.7 derive the sequence of computational steps that make up the environment transition and reward functions, respectively.

A1. Preliminaries

Today, the majority of AC transmission and distribution networks dispatch electricity using the so-called three-phase system. In this system, electricity flows in three parallel circuits, each associated with its own phase. In a balanced three-phase network, the electrical quantities of each phase have the same magnitude and differ by a 120° phase shift, i.e. phase 3 is a time-delayed version of phase 2, which is itself a time-delayed version of phase 1. Conveniently, any balanced three-phase system can thus be analyzed using an equivalent single-phase representation, where only one of the phases is taken into account. The complex phasors corresponding to the other two phases can be obtained by applying a 120° or 240° phase shift to the first-phase phasors. All systems implemented by Gym-ANM are assumed to be such three-phase balanced networks, and we adopt its equivalent single-phase representation in the following derivations.

In order to efficiently generate and distribute electricity, power grids are also divided into so-called voltage zones. Each zone is characterized by a particular nominal voltage level that represents the average voltage level of the nodes in that zone. For instance, a 220kV (ultra-high voltage) transmission network may be connected to an intermediary 150kV (high voltage) network, which is then connected to a 30kV (medium voltage) distribution network. Transitions between the different voltage levels are carried out by power transformers that bring up (step-up transformers) or down (step-down transformers) voltages while minimizing power losses. For mathematical convenience, power systems that include several voltage zones are often analyzed using the per-unit (p.u.) notation, in which all electrical quantities are normalized with respect to a set of base quantities chosen for the whole system. In practice, the per-unit analysis method becomes very handy as it removes the need to include nominal voltage levels in derivations. This allows us to analyze the network as a single circuit and cancels out the effect of transformers whose tap ratio is identical to the ratio of the base voltages of the zones it connects. In other words, only so-called off-nominal transformers need to be considered. In the remainder of this appendix, all quantities are expressed in p.u.

A2. Branches

As introduced in Section 2.2, we model a distribution network as a set of nodes $\mathcal{N}$ connected by a set of directed edges $\mathcal{E}$. Each edge $e_{ij} \in \mathcal{E}$ may represent a sequence of (a) transmission lines, (b) power transformers, and/or (c) phase shifters linking buses i and j. Any combination of (a)-(c) components can be equivalently mapped to the common branch representation adapted from [33] and shown in Fig. 7. Formally, branch $e_{ij} \in \mathcal{E}$ is characterized by five parameters: a series resistance r_{ij}, a series reactance x_{ij}, a total charging susceptance b_{ij}, a tap ratio magnitude τ_{ij}, and a phase shift θ_{ij}. The branch series admittance is given by $y_{ij} = (r_{ij} + \mathbf{i}x_{ij})^{-1}$, each shunt admittance by $y^{sh}_{ij} = \mathbf{i}\frac{b_{ij}}{2}$, and the complex tap ratio of the off-nominal transformer by $t_{ij} = \tau_{ij}e^{\mathbf{i}\theta_{ij}}$. Note that one can use a value of $t_{ij} = 1$ to represent the absence of a transformer, or, equivalently, the presence of an

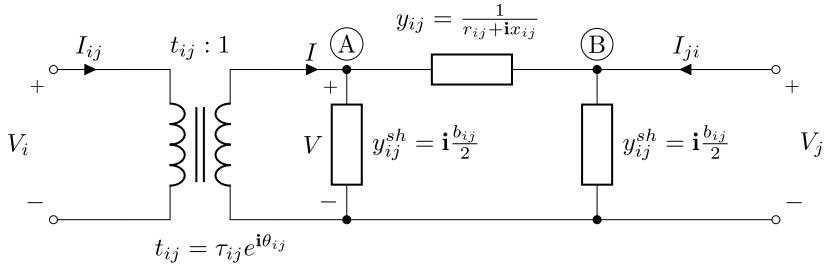


Fig. 7. Common branch model, consisting of a π transmission line model in series with an off-nominal phase-shifting transformer, connecting bus $i \in \mathcal{N}$ and bus $j \in \mathcal{N}$.

on-nominal transformer.

A3. Electrical devices

The different electrical devices $\mathcal{D}$ connected to the grid are classified as passive loads $\mathcal{D}_L$, generators $\mathcal{D}_G$, or DES units $\mathcal{D}_{DES}$. Within generators, we further differentiate between renewable generators $\mathcal{D}_{RER} \subset \mathcal{D}_G$ and the slack generator $g^{slack} \in \mathcal{D}_G - \mathcal{D}_{RER}$. Much like what was done by Gemine et al. [22], the range of operation of each device $d \in \mathcal{D}$ is modelled by a set $\mathcal{R}_{d,t} \subset \mathbb{R}^2$ of valid $(P_{d,t}^{(dev)}, Q_{d,t}^{(dev)})$ power injection points for timestep t. These constraints are enforced by the environment at all times (see Appendix A.6).

A3.1. Passive loads

We define passive loads as the devices that only withdraw power from the network. We also assume that each passive load $l \in \mathcal{D}_L$ has a constant power factor $\cos\phi_l$ and that its negative injection $P_{l,t}^{(dev)}$ is lower bounded[3] by $\underline{P}_l$. Formally, the range of operation $\mathcal{R}_{l,t} = \mathcal{R}_l$ of l is defined by:

$$\mathcal{R}_l = \left\{ (P, Q) \in \mathbb{R}^2 \;\middle|\; \underline{P}_l \leq P \leq 0, \frac{Q}{P} = \tan\phi_l \right\}, \quad \forall l \in \mathcal{D}_L \,, \tag{A.1}$$

for all $t \in \mathcal{T}$.

A3.2. Generators

Generators, with the exception of g^{slack}, refer to devices that only inject power into the network. The physical limitations of any generator $g \in \mathcal{D}_G$ are modelled by a range of allowed active power injections $\left[\underline{P}_g, \overline{P}_g\right]$ and of reactive power injections $\left[\underline{Q}_g, \overline{Q}_g\right]$. Additional linear constraints $Q_{g,t}^{(dev)} \leq \tau_g^{(1)} P_{g,t}^{(dev)} + \rho_g^{(1)}$ and $Q_{g,t}^{(dev)} \geq \tau_g^{(2)} P_{g,t}^{(dev)} + \rho_g^{(2)}$ can also be added to limit the flexibility of reactive power injection when P is close to its maximum value.[4] These constraints result in the range of operation shown in Fig. 8. Finally, a dynamic upper bound $P_{g,t}^{(max)} \in \left[\underline{P}_g, \overline{P}_g\right]$ is also generated by the

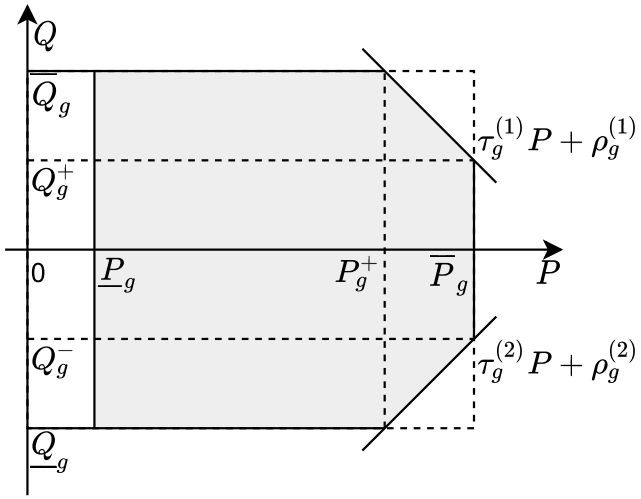


Fig. 8. Fixed power injection constraints of distributed generators $g \in \mathcal{D}_G - \{g^{slack}\}$.

[3] When designing a new environment, the user can set $\underline{P}_l = -\infty$ to model an unbounded load. Note that a finite lower bound value is required to have a bounded state space $\mathcal{S}$.

[4] These additional linear flexibility constraints can be used to approximate current limits of power converters and/or of electric generators [34]. They can also be ignored by setting $Q_g^+ = \overline{Q}_g$ and $Q_g^- = \underline{Q}_g$.

`next_vars()` block to model time-dependent constraints on $P_{g,t}^{(dev)}$.

The resulting dynamic region of operation $\mathscr{R}_{g,t}$ is formally expressed as:

$$\mathscr{R}_{g,t} = \left\{(P,Q) \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} \underline{P}_g \le P \le P_{g,t}^{(max)}, \underline{Q}_g \le Q \le \overline{Q}_g, Q \le \tau_g^{(1)} P + \rho_g^{(1)}, \\ Q \ge \tau_g^{(2)} P + \rho_g^{(2)} \end{array}\right\}, \quad \forall g \in \mathscr{G} - \left\{g^{slack}\right\}, \tag{A.2}$$

where $\tau_g^{(1)}, \rho_g^{(1)}, \tau_g^{(2)}, \rho_g^{(2)}$ are computed based on the parameters $\left\{\overline{P}_g, P_g^+, \underline{Q}_g, \overline{Q}_g, Q_g^+, Q_g^-\right\}$ provided in the network input dictionary (see Appendix D) as:

$$\begin{aligned} \tau_g^{(1)} &= \frac{Q_g^+ - \overline{Q}_g}{\overline{P}_g - P_g^+}, \quad \rho_g^{(1)} = \overline{Q}_g - \tau_g^{(1)} P_g^+, \\ \tau_g^{(2)} &= \frac{Q_g^- - \underline{Q}_g}{\overline{P}_g - P_g^+}, \quad \rho_g^{(2)} = \underline{Q}_g - \tau_g^{(2)} P_g^+. \end{aligned} \tag{A.3}$$

In order to ensure that a solution to the network equations derived in Section A.5 is found at each timestep, we do not restrict the range of operation of the slack generator g^{slack}. Instead, we assume that it can provide unlimited active and reactive power to the network.

A3.3. Distributed energy storage (DES)

DES units can both inject power into (discharge) and withdraw power from (charge) the network. Their time-independent physical constraints are modelled much like that of generators, as shown in Fig. 9, where:

$$\tau_d^{(1)} = \frac{Q_d^+ - \overline{Q}_d}{\overline{P}_d - P_d^+}, \quad \rho_d^{(1)} = \overline{Q}_d - \tau_d^{(1)} P_d^+, \tag{A.4}$$

$$\tau_d^{(2)} = \frac{Q_d^- - \underline{Q}_d}{\overline{P}_d - P_d^+}, \quad \rho_d^{(2)} = \underline{Q}_d - \tau_d^{(2)} P_d^+, \tag{A.5}$$

$$\tau_d^{(3)} = \frac{\underline{Q}_d - Q_d^-}{P_d^- - \underline{P}_d}, \quad \rho_d^{(3)} = \underline{Q}_d - \tau_d^{(3)} P_d^-, \tag{A.6}$$

$$\tau_d^{(4)} = \frac{\overline{Q}_d - Q_d^+}{P_d^- - \underline{P}_d}, \quad \rho_d^{(4)} = \overline{Q}_d - \tau_d^{(4)} P_d^-. \tag{A.7}$$

Unlike generators, however, the active power injection of a DES unit $d \in \mathscr{D}_{DES}$ is further constrained by its current state of charge $SoC_{d,t} \in \left[\underline{SoC}_d, \overline{SoC}_d\right]$. For instance, a fully charged unit would not be able to withdraw even the slightest amount of active power. Consequently, we chose to impose additional limits on their next active power injection $P_{d,t+1}^{(dev)}$. This is to ensure that the injection can stay constant within $(t, t+1]$ without violating any storage level constraints, i.e. that $\underline{SoC}_d \le SoC_{d,t+1} \le \overline{SoC}_d$. Given that $SoC_{d,t+1}$ is obtained from:

$$SoC_{d,t+1} = \begin{cases} SoC_{d,t} - \Delta t \eta P_{d,t+1}^{(dev)}, & \text{if } P_{d,t+1}^{(dev)} \le 0, \\ SoC_{d,t} - \dfrac{\Delta t}{\eta} P_{d,t+1}^{(dev)}, & \text{else,} \end{cases} \tag{A.8}$$

where $\eta \in [0,1]$ is the charging and discharging efficiency factor (assumed equal), the condition $\underline{SoC}_d \le SoC_{d,t+1} \le \overline{SoC}_d$ can be re-expressed as a

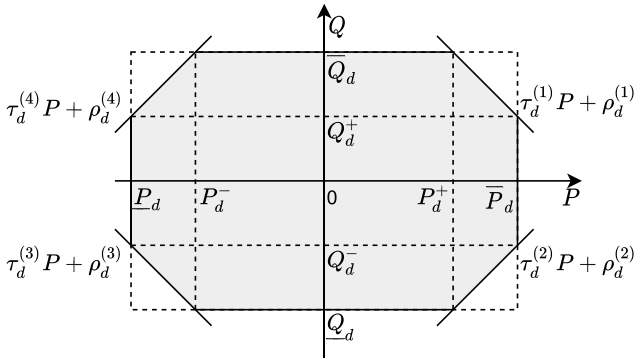


Fig. 9. Fixed power injection constraints for DES units $d \in \mathscr{D}_{DES}$.

constraint on $P^{(dev)}_{d,t+1}$ as:

$$\frac{1}{\Delta t \eta}\left(SoC_{d,t} - \overline{SoC}_d\right) \leq P^{(dev)}_{d,t+1} \leq \frac{\eta}{\Delta t}\left(SoC_{d,t} - \underline{SoC}_d\right) . \tag{A.9}$$

In summary, the range of operation of each DES unit $d \in \mathscr{D}_{DES}$ is modelled by the time-varying constrained set $\mathscr{R}_{d,t}$:

$$\begin{aligned} \mathscr{R}_{d,t} = \big\{(P,Q) \in \mathbb{R}^2 \mid \quad & \underline{P}_d \leq P \leq \overline{P}_d , \\ & \underline{Q}_d \leq Q \leq \overline{Q}_d , \\ & Q \leq \tau^{(1)}_d P + \rho^{(1)}_d , \\ & Q \geq \tau^{(2)}_d P + \rho^{(2)}_d , \\ & Q \geq \tau^{(3)}_d P + \rho^{(3)}_d , \\ & Q \leq \tau^{(4)}_d P + \rho^{(4)}_d , \\ & P \geq \frac{1}{\Delta t \eta}\left(SoC_{d,t-1} - \overline{SoC}_d\right) , \\ & P \leq \frac{\eta}{\Delta t}\left(SoC_{d,t-1} - \underline{SoC}_d\right)\Big\} , \quad \forall d \in \mathscr{D}_{DES} \end{aligned} \tag{A.10}$$

A4. Network constraints

Constraints on the operating range of each electrical device in $\mathscr{D}$ (derived in Appendix A.3) get enforced by the environment during each timestep transition (see Appendix A.6). Unlike these constrains, however, network constraints will be left unchecked but will generate a large negative reward when not met, as further detailed in Appendix A.7. That is, the simulator will allow the network to operate past the following network constraints, but will penalize through negative rewards any policy that does so.

As introduced in Section 3.6, we consider two types of such network constraints that network operators should ensure are satisfied at all times: voltage and line current constraints. The first one is a constraint on bus voltage magnitudes, which must be kept within a close range of their nominal value to ensure stability of the grid:

$$\underline{V}_i \leq \left|V_{i,t}\right| \leq \overline{V}_i , \quad \forall i \in \mathscr{N}, \forall t \in \mathscr{T} , \tag{A.11}$$

where $\underline{V}_i$ and $\overline{V}_i$ are often chosen close to 1 p.u. and voltages are expressed as root mean squared (RMS) values.

The second one is an upper limit on line currents, which are determined by materials and environmental conditions. Let $\overline{I}_{ij}$ be the maximum physical current magnitude allowed through branch $e_{ij} \in \mathscr{E}$. In practice, such limits are often expressed as apparent power flow limits $\overline{S}_{ij}$ at a 1 p.u. nodal voltage. The reason behind this choice is the fact that the apparent power flow $|S_{ij}| = |V_i I^*_{ij}|$ is close to $|I_{ij}|$ when voltage magnitudes are kept close to unity by constraint (A.11). For consistency with existing optimization tools that model line current limits as apparent power flow constraints, we chose to adopt the same approach in Gym-ANM. In addition, for a given branch $e_{ij} \in \mathscr{E}$, the branch current at the sending end $|I_{ij}|$ may be different to the current injection at the receiving end $|I_{ji}|$. This is due to the asymmetry of the common branch model of Fig. 7. The constraints must thus be respected at each end of the branch:

$$|S_{ij,t}| \leq \overline{S}_{ij} \quad \text{and} \quad \left|S_{ji,t}\right| \leq \overline{S}_{ij} , \quad \forall e_{ij} \in \mathscr{E}, \forall t \in \mathscr{T} . \tag{A.12}$$

A5. Network equations

The flow of electricity within a power network is dictated by a set of network equations, or power flow equations, which we will now derive. The following derivations assume that all AC quantities are expressed in RMS terms.

The ideal transformer with complex tap ratio $t_{ij} : 1$ used in the common branch model introduced in Section A.2 can be further described by the relations:

$$V = \frac{V_i}{t_{ij}} \quad \text{and} \quad I = t^*_{ij} I_{ij} . \tag{A.13}$$

Applying Kirchhoff's current law at nodes A and B of Fig. 7 yields:

$$\begin{cases} I = V y^{sh}_{ij} + (V - V_j) y_{ij} \\ I_{ji} = V_j y^{sh}_{ij} + (V_j - V) y_{ij} \end{cases} , \tag{A.14}$$

which, after substituting (A.13), becomes:

$$\begin{cases} I_{ij} = \dfrac{1}{|t_{ij}|^2}\left(y_{ij} + y_{ij}^{sh}\right)V_i - \dfrac{1}{t_{ij}^*}y_{ij}V_j \\ \\ I_{ji} = -\dfrac{1}{t_{ij}}y_{ij}V_i + \left(y_{ij} + y_{ij}^{sh}\right)V_j \end{cases} . \tag{A.15}$$

Expressions (A.15) can be equivalently presented in matrix form:

$$\begin{bmatrix} I_{ij} \\ I_{ji} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{|t_{ij}|^2}\left(y_{ij} + y_{ij}^{sh}\right) & -\dfrac{1}{t_{ij}^*}y_{ij} \\ \\ -\dfrac{1}{t_{ij}}y_{ij} & \left(y_{ij} + y_{ij}^{sh}\right) \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix} , \tag{A.16}$$

which is one possible formulation of the power flow equations.

However, the most commonly used formulation in practice is obtained after applying Kirchhoff's current law at each bus $i \in \mathcal{N}$, which results in the classical matrix formulation:

$$\mathbf{I} = \mathbf{YV} , \tag{A.17}$$

where $\mathbf{I} = [I_0, I_1, \ldots, I_{|\mathcal{N}|-1}]^T$ is the vector of bus current injections, $\mathbf{V} = [V_0, V_1, \ldots, V_{|\mathcal{N}|-1}]^T$ the vector of corresponding bus voltages, and $\mathbf{Y} \in \mathbb{C}^{|\mathcal{N}|\times|\mathcal{N}|}$ the nodal admittance matrix with elements:

$$\mathbf{Y}_{ij} = \begin{cases} -\dfrac{1}{t_{ij}^*}y_{ij} , & \text{if } i \neq j \text{ and } e_{ij} \in \mathcal{E} , \\ \\ -\dfrac{1}{t_{ji}}y_{ji} , & \text{if } i \neq j \text{ and } e_{ji} \in \mathcal{E} , \\ \\ \displaystyle\sum_{e_{ik}\in\mathcal{E}} \frac{1}{|t_{ik}|^2}\left(y_{ik} + y_{ik}^{sh}\right) + \sum_{e_{ki}\in\mathcal{E}}\left(y_{ki} + y_{ki}^{sh}\right) , & \text{if } i = j , \\ \\ 0 , & \text{otherwise} . \end{cases} \tag{A.18}$$

Finally, (A.17) can also be formulated in terms of nodal power injections and voltage levels, removing the need to compute current injections:

$$P_i^{(bus)} + \mathbf{i}Q_i^{(bus)} = V_i I_i^* = V_i(\mathbf{Y}_i\mathbf{V})^* = V_i\mathbf{Y}_i^*\mathbf{V}^* , \quad \forall i \in \mathcal{N} , \tag{A.19}$$

where $\mathbf{Y}_i$ denotes the i^{th} row of the admittance matrix $\mathbf{Y}$.

The power flow equations (A.19) represent a set of $|\mathcal{N}|$ complex-valued equations that the environment solves during the `next_state()` call of each timestep transition. To do so, every bus is modelled as a PQ bus: the $P_i^{(bus)}$ and $Q_i^{(bus)}$ variables are set by the environment (based on the agent's action) and the V_i variables are left as free variables for the solver. The only exception is the slack bus, where the opposite is true: V_i is fixed to $1\angle 0^\circ$ and $P_i^{(bus)}$, $Q_i^{(bus)}$ are the variables. This setup results in a system of $2|\mathcal{N}|$ quadratic real-valued equations with $2|\mathcal{N}|$ free real variables.

A6. Transition function

Based on the current state $s_t \in \mathcal{S}$, each timestep transition starts by sampling the internal variables through the `next_vars()` block of Fig. 2. Note that this block can be uniquely designed for different environments. The remainder of the transition function happens with the `next_state()` component in a deterministic manner, which we now describe as a series of steps analogous to the underlying implementation.

1. Load injection point First, the reactive power injection $Q_{l,t+1}^{(dev)}$ of each load $l \in \mathcal{D}_L$ is inferred from its new demand $P_{l,t+1}^{(dev)}$ outputted by `next_vars()`, according to (A.1):

$$Q_{l,t+1}^{(dev)} = P_{l,t+1}^{(dev)}\tan\phi_l , \tag{A.20}$$

where $P_{l,t+1}^{(dev)}$ is first clipped to $\left[\underline{P}_l, 0\right]$.

2. Distributed generator injection point The power injection point of each distributed generator $g \in \mathcal{D}_G - \{g^{slack}\}$ is computed based on its allowed range of operation $\mathcal{R}_{g,t+1}$ given by (A.2). The active and reactive injections $a_{P_{g,t}}$, $a_{Q_{g,t}}$ are then set by the agent in a_t:

$$\left(P_{g,t+1}^{(dev)}, Q_{g,t+1}^{(dev)}\right) = \underset{(P,Q)\in\mathcal{R}_{g,t+1}}{argmin} \parallel \left(a_{P_{g,t}}, a_{Q_{g,t}}\right) - (P, Q) \parallel . \tag{A.21}$$

In the case where the $(a_{P_{g,t}}, a_{Q_{g,t}})$ injection point set by the agent falls outside of $\mathcal{R}_{g,t+1}$, the environment selects the closest point in $\mathcal{R}_{g,t+1}$, according to

the Euclidean distance.

3. DES injection point Similarly, the power injection point of each DES unit $d \in \mathscr{D}_{DES}$ is computed based on the $(a_{P_{d,t}}, a_{Q_{d,t}})$ point chosen by the agent in a_t and the operating range $\mathscr{R}_{d,t+1}$ of d given by (A.11). We again use the Euclidean distance as the distance metric, resulting in:

$$\left(P^{(dev)}_{d,t+1}, Q^{(dev)}_{d,t+1}\right) = \underset{(P,Q)\in\mathscr{R}_{d,t+1}}{argmin} \quad \| \left(a_{P_{d,t}}, a_{Q_{d,t}}\right) - (P,Q) \| \ . \tag{A.22}$$

4. Power flows & bus voltages Now that the power injection point of each device, with the exception of the slack generator, is known, the total nodal active and reactive power injection for each non-slack bus i is computed using:

$$P_i = \sum_{d\in\mathscr{D}_i} P_d \quad \text{and} \quad Q_i = \sum_{d\in\mathscr{D}_i} Q_d \ . \tag{A.23}$$

After fixing the slack bus voltage to unity, the environment then solves the network equations given by (A.19). Our implementation uses the Newton-Raphson procedure[35] to do so. From the solution, we obtain the voltage $V_i^{(bus)}$ at each non-slack bus and the slack generator power injection point $(P_{g^{slack},t+1}, Q_{g^{slack},t+1})$.

5. State construction The new state vector s_{t+1} can now be constructed according to the structure defined by (3.2). The active and reactive power injection points $P_{d,t+1}$, $Q_{d,t+1}$ have already been computed. The new charge level $SoC_{d,t+1}$ of each DES unit $d \in \mathscr{D}_{DES}$ is obtained using expression (A.8). Finally, the $P^{(max)}_{g,t+1}$ and $aux^{(k)}_{t+1}$ variables are simply copied from the output of `next_vars()`.

A7. Reward function

The main component of the reward signal, as introduced in (3.4) and (3.5), is a sum of three energy losses and a penalty term associated with violating operating constraints:

$$c_t = -(\Delta E_{t:t+1} + \lambda\phi(s_{t+1})) \tag{A.24}$$

We chose to compute both terms in p.u. to ensure similar orders of magnitude.

A7.1. Energy loss

The transmission energy loss, $\Delta E^{(1)}_{t:t+1}$, is computed as:

$$\Delta E^{(1)}_{t:t+1} = \Delta t \sum_{d\in\mathscr{D}} P^{(dev)}_{d,t+1}, \tag{A.25}$$

where Δt is used to get the energy loss in p.u. per hour. The net amount of energy flowing from the grid into DES units, $\Delta E^{(2)}_{t:t+1}$, is obtained using:

$$\Delta E^{(2)}_{t:t+1} = -\Delta t \sum_{d\in\mathscr{D}_{DES}} P^{(dev)}_{d,t+1}. \tag{A.26}$$

Finally, the amount of energy loss as a result of renewable energy curtailment, $\Delta E^{(3)}_{t:t+1}$, is:

$$\Delta E^{(3)}_{t:t+1} = \Delta t \sum_{g\in\mathscr{D}_{RER}} \left(P^{(max)}_{g,t+1} - P^{(dev)}_{g,t+1}\right). \tag{A.27}$$

Summing (A.25)-(A.27) together yields the total energy loss:

$$\Delta E_{t:t+1} = \Delta t \left(\sum_{d\in\mathscr{D}_G\cup\mathscr{D}_L} P^{(dev)}_{d,t+1} + \sum_{g\in\mathscr{D}_{RER}} \left(P^{(max)}_{g,t+1} - P^{(dev)}_{g,t+1}\right) \right) \tag{A.28}$$

A7.2. Constraint-violation penalty

Let $\Phi : \mathscr{S} \to \mathbb{R}$ be the penalty function that adds a large cost $\lambda\Phi(s_{t+1})$ to a policy that leads to a violation of operating constraints. To compute $\Phi(s_{t+1})$, the environment first computes the node voltages $V_{i,t+1}$ using (A.19) and the directed branch currents $I_{ij,t+1}$ and $I_{ji,t+1}$ for each branch $e_{ij} \in \mathscr{E}$ using (A.16). The obtained values are then plugged into $|S_{ij,t+1}| = |V_{i,t+1} I^*_{ij,t+1}|$ and $|S_{ji,t+1}| = |V_{j,t+1} I^*_{ji,t+1}|$ to compute the corresponding branch's apparent power flows. The penalty term $\Phi(s_{t+1})$ is finally obtained using:

$$\Phi(\mathbf{s}_{t+1}) = \Delta t\Big(\sum_{i\in\mathcal{N}} \Big(\max\Big(0, \Big|V_{i,t+1}\Big| - \overline{V}_i\Big) + \max\Big(0, \underline{V}_i - \Big|V_{i,t+1}\Big|\Big)\Big) + \sum_{e_{ij}\in\mathcal{E}} \max\Big(0, \Big|S_{ij,t+1}\Big| - \overline{S}_{ij}, \Big|S_{ji,t+1}\Big| - \overline{S}_{ij}\Big)\Big) . \tag{A.29}$$

Appendix B. Model Predictive Control Scheme

B1. Introduction

This appendix describes the MPC problem solved by the MPC-based policy π_{MPC-N} introduced in Section 3.7. At each timestep, the policy solves a multi-stage DCOPF problem with an optimization horizon of N timesteps. As a linear approximation of the actual ACOPF that we would like to solve, the DCOPF formulation relies on three assumptions, included here again for clarity:

1. Transmission lines are lossless: $r_{ij} = 0,\ \forall e_{ij} \in \mathcal{E}$,
2. The difference between adjacent bus voltage angles is small: $\angle V_i \approx \angle V_j,\ \forall e_{ij} \in \mathcal{E}$,
3. Bus voltage magnitudes are close to unity: $|V_i| \approx 1,\ \forall i \in \mathcal{N}$.

We start by giving a general formulation of the MPC problem in which the algorithm takes as input predictions of future demand and generation in Section B.2. We call this policy π_{MPC-N}. We then consider two particular forecasting methods in Section B.3: one which assumes constant values over the optimization horizon, policy $\pi^{constant}_{MPC-N}$, and another that generates perfect forecasts, policy $\pi^{perfect}_{MPC-N}$.

B2. General formulation

B2.1. Policy overview

The action selection procedure followed by π_{MPC-N} at timestep t is given by Algorithm 2. In this algorithm, `solveMPC()` refers to solving[5] the optimization problem (B.2)-(B.11) and extracting the vector of device active power injection $\mathbf{P}^{(dev)}_{t+1}$ from the solution. The considered-optimal power injections from all non-slack generators and DES units are then concatenated into an action vector a_t. Since reactive power flows are ignored by the DCOPF formulation, we chose to simply set the reactive power set-points in a_t to zero.

In this general formulation, Algorithm 2 takes as inputs the network state s_t (directly extracted from the Gym-ANM simulator) and forecasts of demand and generation over the optimization horizon $k = t+1, \dots, t+N$. We denote these forecasted values as $\tilde{P}^{(dev)}_{l,k}$ and $\tilde{P}^{(max)}_{g,k}$, respectively. An additional safety margin hyperparameter, $\beta \in [0,1]$, is also introduced to further constrain the power flow on each transmission line in the OPF. This is done with the hope that it will account for any errors introduced with the linear DC approximation, thus ensuring that line current constraints are respected. The penalty hyperparameter λ is taken to be the same as in the reward function.

B2.2. The optimization problem

We now describe the optimization problem (B.2)-(B.11) in more detail. The objective function is a simplified version of the cost function used in the reward signal, originally defined as:

$$\sum_{g\in\mathcal{D}_G\cup\mathcal{D}_L} P^{(dev)}_{d,t+1} + \sum_{g\in\mathcal{D}_{RER}} \left(P^{(max)}_{g,t+1} - P^{(dev)}_{g,t+1}\right) + \lambda\phi(s_{t+1}). \tag{B.1}$$

Input: State s_t, demand forecasts $\{\tilde{P}^{(dev)}_{l,t+k}\}_{l\in\mathcal{D}_L,k=1,\dots,N}$, generation forecasts $\{\tilde{P}^{(max)}_{g,t+k}\}_{g\in\mathcal{D}_G-\{g^{slack}\},k=1,\dots,N}$
Parameter: Safety margin $\beta \in [0,1]$, penalty hypeparameter λ
$\{P^{(dev)}_{d,t+1}\}_{d\in\mathcal{D}} \leftarrow$ lstinlinesolveMPC($s_t, \{\tilde{P}^{(dev)}_{l,t+k}\}, \{\tilde{P}^{(max)}_{g,t+k}\}, \beta, \lambda$, grid_characteristicslstinline)
for $g \in \mathcal{D}_G - \{g^{slack}\}$ **do**
    $a_{P_{g,t}} \leftarrow P^{(dev)}_{g,t+1}$
    $a_{Q_{g,t}} \leftarrow 0$
end for
for $d \in \mathcal{D}_{DES}$ **do**
    $a_{P_{d,t}} \leftarrow P^{(dev)}_{d,t+1}$
    $a_{Q_{d,t}} \leftarrow 0$
end for

Algorithm 2. MPC (multi-stage DCOPF) policy π_{MPC-N}.

[5] Our implementation uses the CVXPY Python optimization package [36,37] to solve the optimization program.

In the above formulation, load injections and maximum generations of generators are non-controllable variables (i.e., constants), which can thus be removed from the objective function. In addition, the DCOPF assumptions have $|V_i| = 1$, which leads to $|S_{ij}| = |P_{ij}|$, from which the penalty term $\phi(s_{t+1})$ can be greatly simplified. The resulting objective function to minimize is given by (B.2). Note that we define it as the discounted sum of costs over the optimization horizon. This is to reflect the agent's objective of learning a policy that minimizes the expected discounted return.

Constraints (B.3) and (B.4) express the relationships between nodal power injections, device power injections, and bus voltage angles. Branch power flow equations are formalized by (B.5). Equalities (B.6) constrain the load power injections in the vector $\mathbf{P}_k^{(dev)}$ to the specified forecasted values. Similarly, expression (B.7) uses the forecasted generation upper bounds to limit generator injections in $\mathbf{P}_k^{(dev)}$. Both DES devices and non-slack generators are restricted to their physical range of operation in (B.8), assuming reactive power injections of zero. In (B.9), power injections from DES units are limited to values that ensure $SoC_{d,k+1} \in \left[\underline{SoC}_d, \overline{SoC}_d\right]$. Finally, (B.10) constrains voltage angles to be within $[0, 2\pi]$ radians and (B.11) provides a voltage angle reference by fixing the slack voltage angle to 0.

$$\underset{\substack{\mathbf{P}_k^{(dev)}, \mathbf{V}_k, \\ k=t+1,\dots,t+N}}{\text{minimize}} \sum_{k=t+1}^{t+N} \gamma^{k-t-1} \left(\sum_{g \in \mathscr{D}_G - \mathscr{D}_{RER}} P_{g,k}^{(dev)} + \lambda \sum_{e_{ij} \in \mathscr{E}} \max\left(0, \left|P_{ij,k}\right| - \beta \overline{S}_{ij}\right) \right) \tag{B.2}$$

$$\text{subject to } P_{i,k}^{(bus)} = \sum_{d \in \mathscr{D}_i} P_{d,k}^{(dev)}, \quad \forall i \in \mathscr{N}, \forall k \tag{B.3}$$

$$P_{i,k}^{(bus)} = \sum_{e_{ij} \in \mathscr{E}} B_{ij}\left(\angle V_{i,k} - \angle V_{j,k}\right) + \sum_{e_{ji} \in \mathscr{E}} B_{ji}\left(\angle V_{i,k} - \angle V_{j,k}\right), \forall i \in \mathscr{N}, \forall k \tag{B.4}$$

$$P_{ij,k} = B_{ij}\left(\angle V_{i,k} - \angle V_{j,k}\right), \quad \forall e_{ij} \in \mathscr{E}, \forall k \tag{B.5}$$

$$P_{l,k}^{(dev)} = \widetilde{P}_{l,k}^{(dev)}, \quad \forall l \in \mathscr{D}_L, \forall k \tag{B.6}$$

$$P_{g,k}^{(dev)} \leq \widetilde{P}_{g,k}^{(max)}, \quad \forall g \in \mathscr{D}_G - \left\{g^{slack}\right\}, \forall k \tag{B.7}$$

$$\underline{P}_d \leq P_{d,k}^{(dev)} \leq \overline{P}_d, \quad \forall d \in \mathscr{D}_G \cup \mathscr{D}_{DES} - \left\{g^{slack}\right\}, \forall k \tag{B.8}$$

$$\frac{1}{\Delta t \eta}\left(SoC_{d,k} - \overline{SoC}_d\right) \leq P_{d,k}^{(dev)} \leq \frac{\eta}{\Delta t}\left(SoC_{d,k} - \underline{SoC}_d\right), \forall d \in \mathscr{D}_{DES}, \forall k \tag{B.9}$$

$$0 \leq \angle V_{i,k} \leq 2\pi, \quad \forall i \in \mathscr{N}, \forall k \tag{B.10}$$

$$\angle V_{0,k} = 0, \quad \forall k \tag{B.11}$$

B2.3. Further considerations

The performance achieved by π_{MPC-N} provides a lower bound on the best performance achievable in a given environment. This bound is not tight, however, since the achieved performance depends on (a) the quality of the DC linear approximation, (b) the accuracy of the forecasted values, and (c) the length of the optimization horizon N. Note that, in general, the performance of an MPC-based policy increases as $N \to \infty$. Because of (a) and (b), however, this may not be the case with π_{MPC-N}, since, e.g., erroneous long-term forecasts may harm policies with larger N's. As a result, N may have to be tuned, depending on the environment.

B3. Special cases: Constant and perfect forecast

We now consider two special cases of the MPC-based policy π_{MPC-N}. Both policies were used in the ANM6-Easy environment in Section 5.

B3.1. Constant forecast

The first variant that we consider is $\pi_{MPC-N}^{constant}$. It assumes that load injections $P_{l,t}^{(dev)}$ and maximum generations $P_{g,t}^{(max)}$ remain constant during the optimization horizon. As such, it is one of the simplest variants of π_{MPC-N} that one could use. Formally, we can describe $\pi_{MPC-N}^{constant}$ by its constant forecasts:

$$\begin{cases} \widetilde{P}_{l,k}^{(dev)} = P_{l,t}^{(dev)}, & \forall l \in \mathscr{D}_L,\ k = t+1, \dots, t+N, \\ \widetilde{P}_{g,k}^{(max)} = P_{g,t}^{(max)}, & \forall g \in \mathscr{D}_G - \left\{g^{slack}\right\},\ k = t+1, \dots, t+N. \end{cases} \tag{B.12}$$

The main advantage of $\pi_{MPC-N}^{constant}$ is that it can be used out-of-the-box in any Gym-ANM environment. More information on how to do this can be found on the project repository.

B3.2. Perfect forecast

The second variant that we consider is $\pi_{MPC-N}^{perfect}$. This variant is specifically tailored for the ANM6-Easy environment introduced in Section 4.2. This is because it assumes perfect forecasts of load injections and maximum generations. In other words, it relies on the fact that ANM6-Easy is a deterministic environment in which future demand and generation can be perfectly predicted. Formally, $\pi_{MPC-N}^{perfect}$ uses perfect forecasts:

$$\begin{cases} \widetilde{P}_{l,k}^{(dev)} = P_{l,k}^{(dev)}, & \forall l \in \mathscr{D}_L,\ k = t+1, \ldots, t+N, \\ \widetilde{P}_{g,k}^{(max)} = P_{g,k}^{(max)}, & \forall g \in \mathscr{D}_G - \left\{g^{slack}\right\},\ k = t+1, \ldots, t+N. \end{cases} \tag{B.13}$$

Unlike $\pi_{MPC-N}^{constant}$, policy $\pi_{MPC-N}^{perfect}$ can only be used in deterministic environments of the like of ANM6-Easy. Nevertheless, it offers a large advantage in that it yields a much better performance in such environments. This provides the user with a tighter lower bound on the best achievable performance in the environment.

Appendix C. New Gym-ANM Environments

This appendix gives an overview of the procedure to follow to design new Gym-ANM environments[6] New Gym-ANM environments can be implemented as Python subclasses that inherit the provided `ANMEnv` superclass, following the template presented in Listing C.2.

`__init__()` This method, known as a constructor in object-oriented languages, is called when a new instance of the new environment `MyANMEnv` is created. In order to initialize the environment, the following arguments need to be passed to the superclass, through the call `super().__init__()`:

```
from gym_anm.envs import ANMEnv

class MyANMEnv(ANMEnv):
    def __init__(self):
        network = {'baseMVA':..., 'bus':..., 'device':...,
                   'branch':...}
        obs = [('bus_p', [0,1], 'MW'), ('dev_q', [2], 'MW')]
              # or a callable
        K = 1
        delta = 0.25
        gamma = 0.999
        lamb = 1000      # 'lambda' is a reserved keyword in Python
        r_clip = 100
        seed = None
        super().__init__(network, obs, K, delta, gamma, lamb,
                         r_clip, seed)
    def init_state(self):
        ...
    def next_vars(self, s_t):
        ...
    def observation_bounds(self):  # optional
        ...
```

Fig. 2. Implementation template for new Gym-ANM environments.

Table C1
Available combinations for the `observation` parameter. Each type of observation should be provided as a tuple with the corresponding bus/device indices or with the 'all' keyword. Units can also be specified. For instance: `[('bus_p', 'all', 'pu'), ('dev_q', [1,2], 'MVAr'), ('branch_s', [(1,2)])]` would lead to observation vectors $o_t = [P_1^{(bus)}, \ldots, P_N^{(bus)}, Q_1^{(dev)}, Q_2^{(dev)}, |S_{12}|]$.

Keyword	Description	Units
bus_p (dev_p)	Bus (Device) active power injection $P_i^{(bus)}$ ($P_d^{(dev)}$)	MW, pu
bus_q (dev_q)	Bus (Device) reactive power injection $Q_i^{(bus)}$ ($Q_d^{(dev)}$)	MVAr, pu
bus_v_magn	Bus voltage magnitude $\lvert V_i \rvert$	pu, kV
bus_v_ang	Bus voltage angle $\angle V_i$	degree, rad
bus_i_magn	Bus current injection magnitude $\lvert I_i \rvert$	pu, kA
bus_i_ang	Bus current injection angle $\angle I_i$	degree, rad
branch_p	Branch active power flow P_{ij}	MW, pu
branch_q	Branch reactive power flow Q_{ij}	MVAr, pu
branch_s	Branch apparent power flow $\lvert S_{ij} \rvert$	MVA, pu
branch_i_magn	Branch current magnitude $\lvert I_{ij} \rvert$	pu
branch_i_ang	Branch current angle $\angle I_{ij}$	degree, rad
des_soc	SOC of DES SoC_d	MWh, pu
gen_p_max	Generator dynamic upper bound $P_g^{(max)}$	MW, pu
aux	Vector of K auxiliary variables $aux^{(K)}$	–

[6] Further guidelines and tutorials can be found on the project repository.

- `network`: a Python dictionary that describes the structure and characteristics of the distribution network G and the set of electrical devices $\mathscr{D}$. Its structure should follow the one given in Appendix D.
- `obs`: a list of tuples corresponding to the variables to include in observation vectors. In Listing C.2, o_t is constructed as $(P_{0,t}^{(bus)}, P_{1,t}^{(bus)}, Q_{2,t}^{(dev)})$, all in MW units. The full list of supported combinations is given in Table C.3. Alternatively, the `obs` object can be defined as a customized callable object (function) that returns observation vectors when called (i.e., $o_t = \texttt{obs}(s_t)$), or as a string '`state`'. In the later case, the environment becomes fully observable and observations $o_t = s_t$ are emitted.
- `K`: the number of auxiliary variables K in the state vector given by (3.2).
- `delta`, `gamma`, `lamb`, `r_clip`: the hyperparameters Δt, γ, λ, and r_{clip}, respectively, used to compute the rewards and returns, as introduced in Section 3.1.
- `seed`: an integer to be used as random seed.

`init_state()` This method will be called once at the start of each new trajectory and it should return an initial state vector s_0 that matches the structure of (3.2). In the case where s_0 falls outside of $\mathscr{S}$ because, for instance, the (P, Q) injection point of a device falls outside of its operating range, the environment will map s_0 to the closest element of $\mathscr{S}$ according to the Euclidean distance. In short, `init_state()` implements $p_0(\cdot)$.

`next_vars()` As introduced in Section 3.1, `next_vars()` is a method that receives the current state vector s_t and should return the outcomes of the internal variables for timestep $t+1$. It must be implemented by the designer of the task, with the only constraint being that it must return a list of $|\mathscr{D}_L| + |\mathscr{D}_{RER}| + K$ values.

`observation_bounds()` This method is optional and only useful if the observation space is specified as a callable object. In the latter case, `observation_space()` should return the (potentially loose) bounds of the observation space $\mathscr{O}$, so that agents can easily normalize emitted observation vectors.

Additional `render()` and `close()` methods can also be implemented to support rendering of the interactions between the agent and the new environment. `render()` should update the visualization every time it gets called, and `close()` should end the rendering process. For more information, we refer to the official OpenAI Gym documentation [24].[7]

Appendix D. Network Input Dictionary

This appendix describes the structure of the Python dictionary required to build new Gym-ANM environments. The dictionary should contain four keys: '`baseMVA`', '`bus`', '`device`', and '`branch`'. The value given to the key '`baseMVA`' should be a single integer, representing the base power of the system (in MVA) used to normalize values to per-unit. Each of the other three keys should be associated with a numpy 2D array, in which each row represents a single bus, device, or branch of the distribution network. The structures of the '`bus`', '`device`', and '`branch`' arrays are described in Tables 4, 5, and 6, respectively.

Table 4
Bus data: description of each row in the '`bus`' numpy array.

Column	Description
0	Bus unique ID i (0-indexing).
1	Bus type (0 = slack, 1 = PQ).
2	RMS base voltage of the zone (kV).
3	Maximum RMS voltage magnitude $\overline{V}_i$ (p.u.).
4	Minimum RMS voltage magnitude $\underline{V}_i$ (p.u.).

Table 5
Device data: description of each row in the '`device`' numpy array.

Column	Description
0	Device unique ID d (0-indexing).
1	Bus unique ID i to which d is connected.
2	Type of device (-1 = load; 0 = slack; 1 = classical generator; 2 = distributed renewable energy generator; 3 = DES unit).
3	Constant ratio of reactive power over active power $(Q/P)_d$ (loads only).
4	Maximum active power output $\overline{P}_d$ (MW).
5	Minimum active power output $\underline{P}_d$ (MW).
6	Maximum reactive power output $\overline{Q}_d$ (MVAr).
7	Minimum reactive power output $\underline{Q}_d$ (MVAr).
8	Positive active power output of PQ capability curve P_d^+ (MW).
9	Negative active power output of PQ capability curve P_d^- (MW).
10	Positive reactive power output of PQ capability curve Q_d^+ (MVAr).
11	Negative reactive power output of PQ capability curve Q_d^- (MVAr).
12	Maximum state of charge of storage unit $\overline{SoC}_d$ (MWh).
13	Minimum state of charge of storage unit $\underline{SoC}_d$ (MWh).
14	Charging and discharging efficiency coefficient of storage unit η_d.

[7] https://gym.openai.com/docs/

Table 6
Branch data: description of each row in the 'branch' numpy array.

Column	Description
0	Sending-end bus unique ID i.
1	Receiving-end bus unique ID j.
2	Branch series resistance r_{ij} (p.u.).
3	Branch series reactance x_{ij} (p.u.).
4	Branch total charging susceptance b_{ij} (p.u.).
5	Branch rating $\overline{S}_{ij}$ (MVA).
6	Transformer off-nominal turns ratio τ_{ij}.
7	Transformer phase shift angle θ_{ij} (degrees) (> 0 = delay).

Appendix E. ANM6-Easy Environment

This appendix describes in more detail the ANM6-Easy Gym-ANM environment introduced in Section 4.2.

E1. Network characteristics

Tables 7, 8, and 9 summarize the characteristics of buses, electrical devices, and branches, respectively, following the network input dictionary structure given in Appendix Appendix D. Based on the values given in Table 8, the range of operation (see Appendices A.3.2 and A.3.3) of the distributed generators and the DES unit of ANM6-Easy are also plotted in Fig. 10.

The fixed time series used by the `next_vars()` component of ANM6-Easy, in order to model the evolution of the loads and of the maximum generation from renewable energy resources, are provided in Table 10.

Table 7
Description of each bus $i \in \mathcal{N}$ of ANM6-Easy .

i	Type	Base voltage	$\overline{V}_i$	$\underline{V}_i$
0	0	132	1.04	1.04
1	1	33	1.1	0.9
2	1	33	1.1	0.9
3	1	33	1.1	0.9
4	1	33	1.1	0.9
5	1	33	1.1	0.9

Table 8
Description of each electrical device $d \in \mathcal{D}$ of ANM6-Easy.

d	i	Type	$(Q/P)_d$	$\overline{P}_d$	$\underline{P}_d$	$\overline{Q}_d$	$\underline{Q}_d$	P_d^+	P_d^-	Q_d^+	Q_d^-	$\overline{SoC}_d$	$\underline{SoC}_d$	η_d
0	0	0	-	-	-	-	-	-	-	-	-	-	-	-
1	3	-1	0.2	0	-10	-	-	-	-	-	-	-	-	-
2	3	2	-	30	0	30	-30	20	-	15	-15	-	-	-
3	4	-1	0.2	0	-30	-	-	-	-	-	-	-	-	-
4	4	2	-	50	0	50	-50	35	-	20	-20	-	-	-
5	5	-1	0.2	0	-30	-	-	-	-	-	-	-	-	-
6	5	3	-	50	-50	50	-50	30	-30	25	-25	100	0	0.9

Table 9
Description of each branch $e_{ij} \in \mathcal{E}$ of ANM6-Easy.

i	j	r_{ij}	x_{ij}	b_{ij}	$\overline{S}_{ij}$	τ_{ij}	θ_{ij}
0	1	0.0036	0.1834	0	32	1	0
1	2	0.03	0.022	0	25	1	0
1	3	0.0307	0.0621	0	18	1	0
2	4	0.0303	0.0611	0	18	1	0
2	5	0.0159	0.0502	0	18	1	0

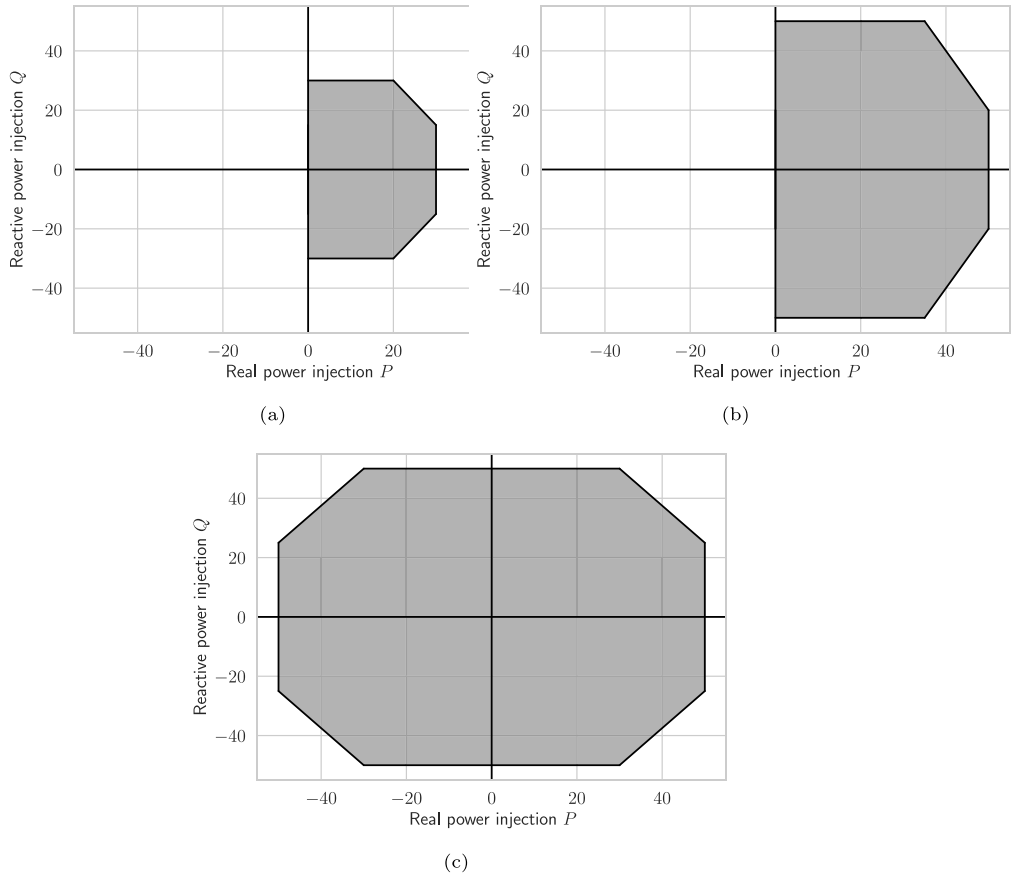


Fig. 10. The range of valid (P, Q) injection points for (a) the solar farm, (b) the wind farm, and (c) the DES unit of ANM6-Easy, as formalized in Appendices A.3.2 and A.3.3.

E2. Environment initialization

The initialization procedure of ANM6-Easy, according to which initial states $s_0 \sim p_0(\cdot)$ are drawn, is illustrated in Algorithm 3. Time series $\mathbf{P}_{1-5}$ refer to Table 10. An initial time of day t_0 is sampled and used to initialize the $aux^{(0)}$ variable (lines 2–3) and to index the fixed time series of active power demand and maximum generation (lines 5, 8). In line 5, the (P, Q) power injection from each load is obtained based on their respective constant power factor. We assume that each distributed generator operates at its maximum active power (i.e., no generator is curtailed) and that its reactive power injection is sampled uniformly. The initial power injection point of each generator is then mapped onto the generator's allowed region of

Table 10
The fixed time series $\mathbf{P}_{1-5}$ used to model the temporal evolution of the loads $P_{l,t}^{(dev)}$, $l \in \{1,3,5\}$, and the maximum generations $P_{g,t}^{(max)}$, $g \in \{2,4\}$, in the ANM6-Easy environment, creating the three challenging situations described in Section 4. The auxiliary variable $aux_t^{(0)} = (t_0 + t) \bmod \frac{24}{\Delta t}$ is used as an index to those time series.

Situation	$aux_t^{(0)}$		$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_5$
1	0–24	92–95	-1	0	-4	40	0
1–2	25	91	-1.5	0.5	-4.75	36.375	-3.125
1–2	26	90	-2	1	-5.5	32.75	-6.25
1–2	27	89	-2.5	1.5	-6.25	29.125	-9.375
1–2	28	88	-3	2	-7	25.5	-12.5
1–2	29	87	-3.5	2.5	-7.75	21.875	-15.625
1–2	30	86	-4	3	-8.5	18.25	-18.75
1–2	31	85	-4.5	3.5	-9.25	14.625	-21.875
2	32–44	72–84	-5	4	-10	11	-25
2–3	45	71	-4.625	7.25	-11.25	14.625	-21.875
2–3	46	70	-4.25	10.50	-12.5	18.25	-18.75
2–3	47	69	-3.875	13.75	-13.75	21.875	-15.625
2–3	48	68	-3.5	17	-15	25.5	-12.5
2–3	49	67	-3.125	20.25	-16.25	29.125	-9.375
2–3	50	66	-2.75	23.5	-17.5	32.75	-6.25
2–3	51	65	-2.375	26.75	-18.75	36.375	-3.125
3	52–64		-2	30	-20	40	0

1: **Output:** $s_0 \in \mathcal{S}$
2: $t_0 \sim U\{0, \frac{24}{\Delta t} - 1\}$
3: $aux_0^{(0)} \leftarrow t_0$
4: **for** $l \in \mathcal{D}_L$ **do**
5: $(P_{l,0}^{(dev)}, Q_{l,0}^{(dev)}) \leftarrow (\mathbf{P}_l[t_0], \mathbf{P}_l[t_0] \tan \phi_l)$
6: **end for**
7: **for** $g \in \mathcal{D}_G - \{g^{slack}\}$ **do**
8: $P_{g,0}^{(max)} \leftarrow \mathbf{P}_g[t_0]$
9: $(P_{g,0}^{(dev)}, Q_{g,0}^{(dev)}) \leftarrow \text{argmin}_{(P,Q)\in\mathcal{R}_{g,0}} \|(P_{g,0}^{(max)}, q) - (P, Q)\|$, with $q \sim U[\underline{Q}_g, \overline{Q}_g]$
10: **end for**
11: **for** $d \in \mathcal{D}_{DES}$ **do**
12: $SoC_{d,0} \sim U[\underline{SoC}_d, \overline{SoC}_d]$
13: $(P_{d,0}^{(dev)}, Q_{d,0}^{(dev)}) \leftarrow (0, 0)$
14: **end for**
15: $(P_{g^{slack},0}^{(dev)}, Q_{g^{slack},0}^{(dev)}) \leftarrow$ solution of (A.20) with $V_0 = 1\angle 0$

Algorithm 3. Initialization of ANM6-Easy, $p_0(\cdot)$.

operation $\mathscr{R}_{g,0}$ (line 9). The initial state of charge of the DES unit is also uniformly sampled and its power injection point is set to zero (lines 12–13). Finally, the slack power injection is obtained after solving the set of network equations (line 15).

Appendix F. Experimental hyperparameters

The hyperparameters used for the experiments presented in Section 5 are summarized in Table 11 for the PPO and in Table 12 for the SAC algorithms. Both implementations were taken from the Stable Baselines 3 library [32]. The horizon T is the maximum number of steps per episode used during training.

Table 11
PPO hyperparameters.

Hyperparameter	Value
Horizon (T)	5000
Adam learning rate	3×10^{-4}
Steps per update	2048
Num. epochs	10
Minibatch size	64
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.2
VF coeff. c1	0.5
Entropy coeff. c2	0.0
Normalized observations	True

Table 12
SAC hyperparameters.

Hyperparameter	Value
Horizon (T)	5000
Adam learning rate	3×10^{-4}
Replay buffer size	10^6
Steps per update	1
Minibatch size	256
Target smoothing coefficient (τ)	0.005
Target update interval	1
Gradient steps	1
Entropy regularization coefficient	'auto'
Normalized observations	True

References

[1] Sutton RS, Barto AG. Reinforcement learning: an introduction. MIT press; 2018.
[2] Glavic M, Fonteneau R, Ernst D. Reinforcement learning for electric power system decision and control: past considerations and perspectives. IFAC-PapersOnLine 2017;50(1):6918–27.20th IFAC World Congress
[3] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:13125602 2013.
[4] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33.
[5] Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, et al. Mastering the game of go with deep neural networks and tree search. Nature 2016; 529(7587):484.
[6] Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 2019;575(7782):350–4.
[7] Deisenroth MP, Neumann G, Peters J, et al. A survey on policy search for robotics. Foundat Trend® Robotic 2013;2(1–2):1–142.
[8] Kormushev P, Calinon S, Caldwell DG. Reinforcement learning in robotics: applications and real-world challenges. Robotics 2013;2(3):122–48.
[9] Kober J, Bagnell JA, Peters J. Reinforcement learning in robotics: a survey. Int J Rob Res 2013;32(11):1238–74.
[10] Gu S, Holly E, Lillicrap T, Levine S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. 2017 IEEE international conference on robotics and automation (ICRA). IEEE; 2017. p. 3389–96.
[11] Sallab AE, Abdou M, Perot E, Yogamani S. Deep reinforcement learning framework for autonomous driving. Electron Imag 2017;2017(19):70–6.
[12] O'Kelly M, Sinha A, Namkoong H, Tedrake R, Duchi JC. Scalable end-to-end autonomous vehicle testing via rare-event simulation. Advances in Neural Information Processing Systems. 2018. p. 9827–38.
[13] Li D, Zhao D, Zhang Q, Chen Y. Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. IEEE Comput Intell Mag 2019;14(2):83–98.
[14] Dulac-Arnold G, Mankowitz D, Hester T. Challenges of real-world reinforcement learning. arXiv preprint arXiv:190412901 2019.
[15] Fang X, Misra S, Xue G, Yang D. Smart gridthe new and improved power grid: asurvey. IEEE Commun Surv Tutor 2011;14(4):944–80.
[16] Joskow PL. Lessons learned from electricity market liberalization. Energy J 2008; 29(Special Issue #2).
[17] Lasseter RH. Microgrids. 2002 IEEE power engineering society winter meeting. Conference proceedings (Cat. No. 02CH37309). 1. IEEE; 2002. p. 305–8.
[18] Capitanescu F, Ochoa LF, Margossian H, Hatziargyriou ND. Assessing the potential of network reconfiguration to improve distributed generation hosting capacity in active distribution systems. IEEE Trans Power Syst 2014;30(1):346–56.
[19] Lutsey N, Slowik P, Jin L. Sustaining electric vehicle market growth in us cities. Int Council Clean Transp (2016) 2016.
[20] Divya K, Østergaard J. Battery energy storage technology for power systemsan overview. Electr Power Syst Res 2009;79(4):511–20.
[21] Götz M, Lefebvre J, Mörs F, Koch AM, Graf F, Bajohr S, et al. Renewable power-to-gas: a technological and economic review. Renew Energy 2016;85:1371–90.
[22] Gemine Q, Ernst D, Cornélusse B. Active network management for electrical distribution systems: problem formulation, benchmark, and approximate solution. Opt Eng 2017;18(3):587–629.
[23] Gill S, Kockar I, Ault GW. Dynamic optimal power flow for active distribution networks. IEEE Trans Power Syst 2013;29(1):121–31.
[24] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. Openai gym. arXiv preprint arXiv:160601540 2016.
[25] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347 2017.
[26] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv: 180101290 2018.
[27] Camacho EF, Alba CB. Model predictive control. Springer Science & Business Media; 2013.
[28] Carpentier J. Contribution lE̦tude du dispatching E̦conomique. Bulletin de la SociE̦tE̦ FranA̸aise des Electriciens 1962;3(1):431–47.
[29] Frank S, Steponavice I, Rebennack S. Optimal power flow: a bibliographic survey I. Energy Syst 2012;3(3):221–58.
[30] Frank S, Steponavice I, Rebennack S. Optimal power flow: a bibliographic survey II. Energy Syst 2012;3(3):259–89.
[31] Chiang H-D, Dobson I, Thomas RJ, Thorp JS, Fekih-Ahmed L. On voltage collapse in electric power systems. IEEE Trans Power Syst 1990;5(2):601–11.
[32] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, N. Dormann, Stable Baselines3, 2019, (https://github.com/DLR-RM/stable-baselines3).
[33] Zimmerman RD, Murillo-Sánchez CE, Thomas RJ. Matpower: steady-state operations, planning, and analysis tools for power systems research and education. IEEE Trans Power Syst 2010;26(1):12–9.
[34] Engelhardt S, Erlich I, Feltes C, Kretschmann J, Shewarega F. Reactive power capability of wind turbines based on doubly fed induction generators. IEEE Trans Energy Convers 2010;26(1):364–72.
[35] Sun DI, Ashley B, Brewer B, Hughes A, Tinney WF. Optimal power flow by newton approach. IEEE Trans Power Apparatu Syst 1984;(10):2864–80.
[36] Diamond S, Boyd S. CVXPY: a python-embedded modeling language for convex optimization. J Mach Learn Res 2016;17(83):1–5.
[37] Agrawal A, Verschueren R, Diamond S, Boyd S. A rewriting system for convex optimization problems. J Control Decis 2018;5(1):42–60.