Composite Experience Replay Based Deep Reinforcement Learning with Application in Wind Farm Control

Hongyang Dong and Xiaowei Zhao

***Abstract*—In this paper, a deep reinforcement learning (RL)-based control approach with enhanced learning efficiency and effectiveness is proposed to address the wind farm control problem. Specifically, a novel composite experience replay (CER) strategy is designed and embedded in the deep deterministic policy gradient (DDPG) algorithm. CER provides a new sampling scheme that can mine the information of stored transitions in-depth by making a trade-off between rewards and temporal-difference (TD) errors. Modified importance-sampling weights are introduced to the training process of neural networks to deal with the distribution mismatching problem induced by CER. Then our CER-DDPG approach is applied to optimizing the total power production of wind farms. The main challenge of this control problem comes from the strong wake effects among wind turbines and the stochastic features of environments, rendering it intractable for conventional control approaches. A reward regularization process is designed along with the CER-DDPG, which employs an additional neural network to handle the bias of rewards caused by the stochastic wind speeds. Tests with a dynamic wind farm simulator (WFSim) show that our method achieves higher rewards with less training costs than conventional deep RL-based control approaches, and it has the ability to increase the total power generation of wind farms with different specifications.**

***Index Terms*—Intelligent Control; Wind Farm Control; Model-Free Control; Reinforcement Learning; Neural Networks.**

I. Introduction

A. Deep Reinforcement Learning-Based Control

Reinforcement Learning (RL) is a booming artificial intelligence technology that integrates the core idea of many subjects, such as machine learning and control theory [1]. An essential principle of RL is trial-and-error: the agent learns its actions by interacting with the environment. This process forms a control policy, which is a mapping between states and actions. RL aims to find a control policy that can optimize the long-term rewards, despite the limitation of environment knowledge and the lack of detailed system models. Due to the significant application potential of RL in various industry fields, it has aroused extensive research interests in recent years [2], [3], [4], [5], [6], [7], [8]. Moreover, deep learning (DL) technique has greatly promoted the development of RL. Deep RL employs deep neural networks as function approximators, rendering it feasible to deal with complex problems. For example, the famous deep Q-network (DQN) [9] was proposed for problems with tabular control domains, and it has been successfully applied to playing video games. Then, the deep deterministic policy gradient (DDPG) algorithm was designed in [10] to handle tasks with continuous/infinite action domains.

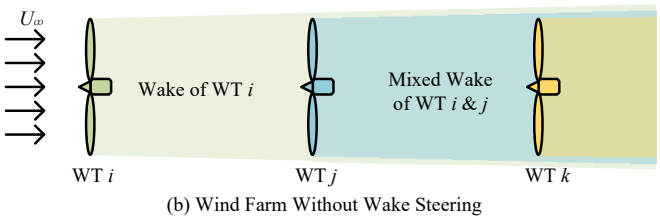
One issue that blocks the direct application of DL to RL is that the sequential observations received by RL agents usually have strong temporal correlations. This violates the independent and identical distribution requirement in many DL algorithms, making the learning process hard to converge or even unstable. Experience replay is an effective method to address this knotty problem, as demonstrated in DQN and DDPG. To be specific, it employs a memory buffer to store previous experiences (i.e., transitions), and a small batch of samples are randomly selected from the memory to train networks at every iteration. In this way, the temporal correlations among the sampled transitions can be largely broke. However, the original experience replay selects transitions through a uniformly random way. This can lead to low training efficiency since the transitions may be more or less surprising, redundant, or task-relevant to the deep RL learning [11]. Aiming to address this issue, Schaul et al. proposed the prioritized experience replay (PER) technique [11] and applied it to DQN. PER measures the priorities of transitions by temporal difference (TD) errors, i.e., the transitions with higher magnitudes of TD errors have higher chances to be sampled. As shown in many studies [11], [12], [13], [14], PER surpasses the performance of the original DQN, and some attempts have been made to extend PER to the DDPG framework [13], [15], [16].

However, as observed in some studies e.g. [13], directly transplanting PER to DDPG may not gain much (even lead to degraded performance) compared with original DDPG. This is because PER was initially designed for DQN where only the action-state value function (i.e. the Q function) is approximated. The TD-error is directly employed to drive this approximation process, rendering it perfect for prioritization purposes. However, compared with DQN, DDPG has an additional actor to generate continuous control policies. Thus, the priorities of transitions for DDPG should be re-designed, since the importance of TD-errors to the actor is questionable. This is one of the challenges that our paper aims to tackle. It motivates us to employ a composite set of priorities to sample transitions, aiming to exploit the information of data

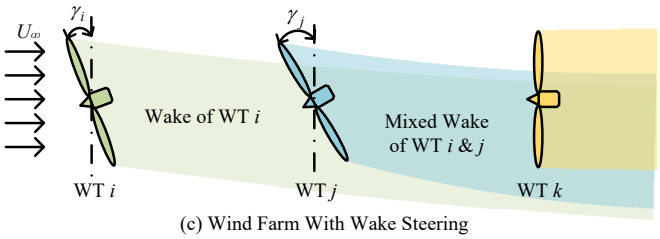
This work was funded by the UK Engineering and Physical Sciences Research Council (grant number: EP/S001905/1).

H. Dong and X. Zhao (corresponding author) are with the Intelligent Control & Smart Energy (ICSE) Research Group, School of Engineering, University of Warwick, Coventry, CV4 7AL, UK. Emails: hongyang.dong@warwick.ac.uk, xiaowei.zhao@warwick.ac.uk.

(a) Denmark's Horns Rev Offshore Wind Farm



(b) Wind Farm Without Wake Steering



(c) Wind Farm With Wake Steering

Figure 1: Illustrations of the wind farm, wake effect, and wake steering

in-depth and balance the learning process of actor and critic, and eventually lead to a novel data-driven model-free deep RL method with enhanced learning efficiency and effectiveness.

B. Control of Wind Farm

Wind power is one of the fastest-growing renewable energy. The development of wind farms (which consist of many individual wind turbines) has been growing quickly in recent years. Fig. 1.a, photoed by Christian Steiness, shows the Denmark's Horns Rev offshore wind farm, which represents the typical layout of wind farms. It also reveals an important feature/issue of wind farms - wake effect [17], [18], [19]. As illustrated in Fig. 1.b, the wake effect means that the upstream wind turbines' power generation process leads to wakes behind them, which can influence the power production of downstream turbines. Horns Rev offshore wind farm suffers a 20% loss on annual power production caused by wake effects [20].

To deal with the influence of wake effects on the efficiency and profitability of wind farms, many researchers studied wake steering strategies [21], [22], [23], [24], [25], [26], [27]. As illustrated in Fig. 1c, wake steering redirects wakes through controlling the yaw angle of every wind turbine. This may decrease the upstream turbines' generation but increase the whole-farm-level power production. However, since the wake effect is a complicated phenomenon, building an accurate model for control actions and wakes is quite difficult. Therefore, model-based wake steering strategies may suffer from handling the resulting modelling errors and uncertainties, leading to degraded performance.

Alternatively, data-driven approaches for wake steering have attracted research interests [28], [29], [30]. A data-driven parametric model for wake effects was proposed in [28], then a tabular-style random-search method based on the parametric model and game theory was designed for yaw settings optimization. A Bayesian ascent approach was proposed in [29], [30], and applied to wind farm control. Compared with the game theory, it can provide a more sophisticated searching strategy and can be applied to continuous state domains. However, these elegant results still require steady-state wind farm models to carry out searching processes, lacking adapting abilities to the time-varying wind speeds and dynamic wake effects. Also, they are all open-loop optimization approaches employing unrestricted searching on the state domain. They cannot provide constrained control policies to optimize power generation from arbitrary initial conditions gradually. These facts motivate us to develop a constrained wind-farm control strategy that is data-driven, model-free and has the ability to adapt to time-varying wind speeds and dynamic wake effects. We build upon RL to achieve this goal.

C. Our Contributions

In this paper, a deep RL-based control approach with enhanced learning efficiency and effectiveness is proposed and applied to wind farm control problems. Particularly, a new sampling strategy is introduced into the DDPG framework. We name it as the composite experience replay (CER) because it utilizes the information of not only TD-errors but also rewards to sample the training batch, and here rewards are employed to measure the priorities of transitions with respect to actor-networks. This design is based on the observation that the updates of actor-networks are implicitly driven by rewards in policy gradient algorithms. In fact, the early versions of policy gradient algorithms directly employed rewards for training, such as the well-known REINFORCE algorithm [1]. CER has the ability to balance the two sets of priorities (i.e., TD errors and rewards), making a trade-off between the learning of critic and actor. Moreover, modified importance-sampling weights based on the composite priorities are employed to correct the biases induced by the distribution mismatching in CER. Testing results show that the CER-DDPG algorithm proposed in this paper achieves higher rewards with less training episodes than the original and PER-based DDPG approaches.

As an application study, we apply CER-DDPG to the control problem of wind farms, aiming to maximize the farm-level power production through wake steering. To this end, an additional reward regularization process is designed along with

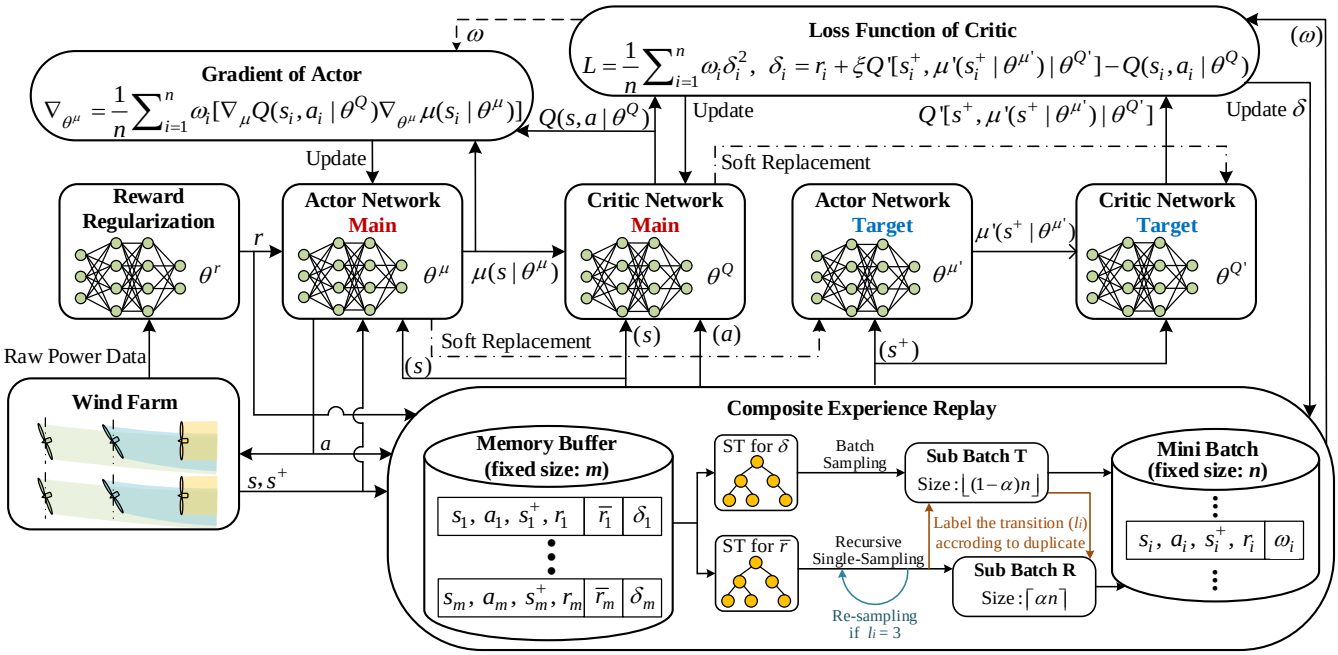


Figure 2: Overview of CER-DDPG with application in wind farm control

the CER-DDPG. It employs a neural network to handle the bias of rewards caused by stochastic wind speeds, enhancing the adapting ability of the algorithm. It is noteworthy that the whole approach is closed-loop and model-free. It can provide bounded control signals and achieve power production optimization under unknown dynamics of wakes and yaw actuators. The algorithm is tested with a dynamic wind farm simulator (WFSim) designed in [31]. Simulation tests indicate that our approach can lead to clear farm-level power production increase compared with conventional strategies.

The remainder of this paper are summarized as follows. The detailed design of our deep RL-based control approach is presented in Sec. II. Then, it is extended and applied to wind farm control in Sec. III. Test results under prototypical wind farm settings are demonstrated in Sec. IV. Finally, Sec. V concludes this paper.

II. Design of Deep Deterministic Policy Gradient with Composite Experience Replay

A. Algorithm Overview

RL is commonly modeled as a Markovian Decision Process (MDP). Specifically, consider an agent whose current state is denoted by $s_t \in \mathcal{S}$ at time t. After taking an action $a_t \in \mathcal{A}$, it transfers to a successor state s_t^+ (i.e. the state at $t+1$) and receives a scalar reward $r_t \in \mathcal{R}$ from the environment. And here $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ denote the state, action, and reward spaces of the agent, respectively. We aim to propose an RL algorithm to learn a control policy $\mu(s)$, such that the long-term return $R_{t_I} = \sum_{t=t_I}^{\infty} \xi^{t-t_I} r_t$ can be maximized, where t_I denotes the start time and $\xi \in (0,1]$ is a discount factor. In this paper, a CER-DDPG algorithm is designed to this end. Fig. 2 illustrates the main structures and data flow of CER-DDPG. It contains several parts: DDPG, composite experience replay, reward regularization, and the wind farm. The main body of CER-DDPG is introduced in the following subsections, then the reward regularization and the mechanism of the wind farm are given in the next section.

B. Deep Deterministic Policy Gradient

We base our approach on DDPG, which essentially is an advanced actor-critic algorithm. Specifically, the critic is designed to evaluate an action-state value function $Q_\mu(s,a)$. At a specific time point t, $Q_\mu(s_t,a_t)$ represents the long-term reward when action a_t is taken at state s_t and a control policy $\mu(s)$ is pursued thereafter. Based on this definition, an essential property of $Q_\mu(s_t,a_t)$ is

$$Q_\mu(s_t,a_t) = r_t + \xi Q_\mu(s_t^+, \mu(s_t^+)) \tag{1}$$

We aim to learn an optimal control policy $\mu^*(s)$ that maximizes $Q_\mu(s_t,a_t)$, formalized by

$$\mu^*(s) = \arg\max_\mu Q_\mu(s_t,a_t) \tag{2}$$

For complicated problems, neural networks (NNs) are usually employed in the actor-critic structure, and their update laws are based on Eqs. (1) and (2). Particularly, four NNs are employed in DDPG, i.e., main actor, main critic, target actor, and target critic. Their parameters are respectively denoted by θ^μ, θ^Q, $\theta^{\mu'}$, and $\theta^{Q'}$. Besides, their outputs are respectively denoted by $\mu(s|\theta^\mu)$, $Q(s,a|\theta^Q)$, $\mu'(s|\theta^{\mu'})$, and $Q'(s,a|\theta^{Q'})$.

Then we introduce the training process of DDPG. The target networks are updated by tracking the main networks via the "soft replacement" law:

$$\begin{aligned} \theta^{\mu'} &\leftarrow (1-\tau)\theta^{\mu'} + \tau\theta^{\mu} \\ \theta^{Q'} &\leftarrow (1-\tau)\theta^{Q'} + \tau\theta^{Q} \end{aligned} \tag{3}$$

and here $\tau \in (0, 1]$ is normally a small constant. The updating of main critic network is driven by the following loss function:

$$L = \frac{1}{n}\sum_{i=1}^{n} \omega_i \delta_i^2 \tag{4}$$

where n denotes the batch size of selected samples at every training step, ω_i is the importance weight of the i^{th} sample, its specific expression is given in the design of CER. Besides, δ_i is the TD-error of the i^{th} sample, defined as

$$\delta_i = r_i + \xi Q'[s_i^+, \mu'(s_i^+|\theta^{\mu'})|\theta^{Q'}] - Q(s_i, a_i|\theta^Q) \tag{5}$$

where s_i^+ denotes the successor state of s_i. The idea of utilizing TD-errors to update the critic is inspired by Eq. (1). Different from the conventional actor-critic approaches, DDPG employs an additional set of actor-critic (i.e. the target actor and critic) that slowly tracks the main actor-critic (as shown in Eq. (3)) to construct TD-errors. This design can reduce the variance induced by the changes of the main actor and critic, enhancing the overall training stability.

Moreover, following Eq. (2), the main actor is designed to achieve $\mu(s|\theta^\mu) = \arg\max_a Q(s, a|\theta^Q)$. However, directly searching the maximum of $Q(s, a|\theta^Q)$ at every training step is intractable. Alternatively, the main actor is updated by a policy gradient strategy:

$$\begin{aligned}\nabla_{\theta^\mu} =& \frac{1}{n}\sum_{i=1}^{n} \omega_i \frac{\partial Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q)}{\partial \theta^\mu} \\ =& \frac{1}{n}\sum_{i=1}^{n} \omega_i [\nabla_\mu Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q) \cdot \nabla_{\theta^\mu} \mu(s_i|\theta^\mu)]\end{aligned} \tag{6}$$

It should be emphasized that a small batch of transitions $\mathcal{B} = \{(s_i, a_i, s_i^+, r_i)\}_{i=1,\ldots,n}$ is employed in every training step. As mentioned in the Introduction, selecting these training batches through Monte Carlo or original PER may result in a degraded training efficiency. A novel sampling strategy, i.e. CER, is designed in the following subsection to address this issue.

C. Composite Experience Replay

CER aims to select transitions that can provide more information than others for the training process of NNs, ultimately improving the overall training effectiveness and efficiency of DDPG. To this end, the sampling priorities of transitions need to be defined. On the one hand, TD errors are directly involved in the loss construction of the critic (as shown in Eq. (4)). Thus they can measure how "surprise" the corresponding transition is, with respect to the critic. On the other hand, as discussed in the Introduction, the actor is implicitly driven by rewards. So rewards are proper choices for transition prioritization purposes with respect to the actor. Moreover, using rewards to measure the importance of transitions is a bio-inspired design since human is usually sensitive to peak rewards [32], [33].

Motivated by these facts, two sets of priorities are stored in CER. The first set of priorities is based on TD errors, we denote it as $\mathcal{P}_\mathcal{T}$, and the priority of the i^{th} transition (i.e., (s_i, a_i, s_i^+, r_i)) is defined as

$$p_{T,i} = |\delta_i| + \epsilon \tag{7}$$

where $\epsilon > 0$ is a small constant employed to ensure that transitions with near-zero TD errors still have probability to be revisited. The other set of priorities is based on rewards, and we denote it as $\mathcal{P}_\mathcal{R}$. To ensure all priorities in $\mathcal{P}_\mathcal{R}$ are valid probabilities, we need to project the reward space from $\mathcal{R}$ to $\mathbb{R}^+$, and we denote the projected reward of the i^{th} transition as $\bar{r}_i$. A simple way to achieve this goal is to set

$$\bar{r}_i = e^{\eta r_i} \tag{8}$$

and here $\eta > 0$ is a constant. Besides, in some applications, $\mathcal{R}$ may be required to be projected to a bounded interval $[\bar{r}_{\min}, \bar{r}_{\max}]$, with $\bar{r}_{\min}, \bar{r}_{\max} > 0$ and $\bar{r}_{\min} < \bar{r}_{\max}$. This can be ensured by setting

$$\bar{r}_i = (\bar{r}_{\max} - \bar{r}_{\min})\mathrm{sig}(\eta r_i) + \bar{r}_{\min} \tag{9}$$

where $\mathrm{sig}(\eta r_i) = 1/(1+e^{-\eta r_i})$. Then, based on the projected reward $\bar{r}_i$, we design the priorities in $\mathcal{P}_\mathcal{R}$ as

$$p_{R,i} = \bar{r}_i + \epsilon \tag{10}$$

Based on these preliminaries, the detailed design of CER is presented in Algorithm 1.

Particularly, the mini batch $\mathcal{B}$ is concurrently constructed by both $\mathcal{P}_\mathcal{T}$ and $\mathcal{P}_R$ with a user-defined ratio α. To effectively sample transitions based on their priorities, two parallel sum-tree (ST) structures (one for $\mathcal{P}_\mathcal{T}$ and the other for $\mathcal{P}_R$) are employed [11]. This design ensures the sampling complexity does not depend on the size of the memory buffer, and it also renders an easier updating process of priorities.

Since the transitions are sampled by two sets of priorities, the duplicate problem needs to be addressed. To this end, special labels are employed in CER as described in Algorithm 1, and they are illustrated in Fig. 3. To be specific, in the mini batch $\mathcal{B}$, $l_i = 1$ means the transition is solely sampled by $\mathcal{P}_\mathcal{T}$; and $l_i = 2$ means the transition is solely sampled by $\mathcal{P}_\mathcal{R}$; finally, $l_i = 3$ means the transition is sampled by both $\mathcal{P}_\mathcal{T}$ and $\mathcal{P}_\mathcal{R}$. Based on these labels, CER ensures that all the transitions in $\mathcal{B}$ are different from each other, no matter they are repeated during the sampling process or not.

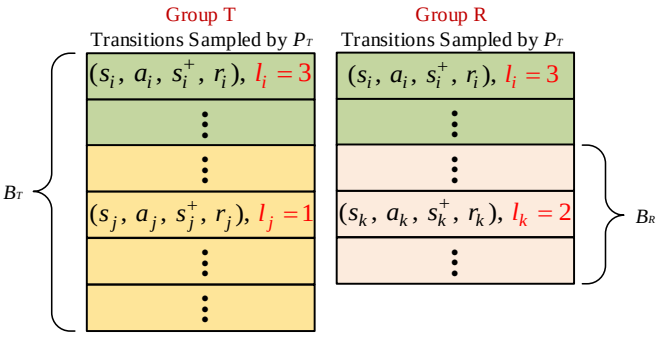


Figure 3: Transitions with different duplicate labels

In general, the estimation of expected values relies on the updates corresponding to the same distributions as their expectations. CER and also other prioritized sampling strategies (e.g. PER) change such distributions. Importance-sampling weights (ISWs) [34] can be utilized in CER to alleviate the potential

Algorithm 1 Composite Experience Replay

$\mathcal{M}$: the memory buffer, with size m.
$\mathcal{B}$: the resulting mini batch in each sampling, with size n.
$\mathcal{B}_\mathcal{T}$: a sub mini batch for the transitions sampled by $\mathcal{P}_\mathcal{T}$.
$\mathcal{B}_\mathcal{R}$: a sub mini batch for the transitions sampled by $\mathcal{P}_\mathcal{R}$.
$P_T(i)$: the sampling probability of the i^{th} transition under $\mathcal{P}_\mathcal{T}$.
$P_R(i)$: the sampling probability of the i^{th} transition under $\mathcal{P}_\mathcal{R}$.
α: the ratio of transitions sampled by $\mathcal{P}_\mathcal{R}$.
l_i: the duplicate label of the i^{th} transition.
λ: the number of transitions in $\mathcal{B}_\mathcal{R}$.
$\omega_{T,i}$: the importance-sampling weight (ISW) of the i^{th} transition under $\mathcal{P}_\mathcal{T}$.
$\omega_{R,i}$: the ISW of the i^{th} transition under $\mathcal{P}_\mathcal{R}$.
ω_i: the final ISW of the i^{th} transition.
β: the exponent in ISW, and $\beta \in (0, 1]$.

for each sampling **do**
  Set l_i, ω_i, and λ to zero, $i = 1, 2, ..., m$.
  Sample $\lfloor(1-\alpha)n\rfloor$ transitions from $\mathcal{M}$ with probability $P_T(j) = p_{T,j}/(\sum_{i=1}^{m} p_{T,i})$ and store them in $\mathcal{B}_\mathcal{T}$.
  Label every sampled transition by setting $l_j = 1$.
  Calculate ISW for every sampled transition: $\omega_j = \omega_{T,j}$, with $\omega_{T,j} = [m \cdot P_T(j)]^{-\beta} / \max_i\{\omega_{T,i}\}$.
  while $\lambda + \lfloor(1-\alpha)n\rfloor) < n$ **do**
    Sample a single transition (s_k, a_k, s_k^+, r_k) from $\mathcal{M}$ with probability $P_R(k) = p_{R,k}/(\sum_{i=1}^{m} p_{R,i})$.
    if $l_k = 0$ **then**
      Label the transition by setting $l_k = 2$.
      Calculate ISW: $\omega_k = \omega_{R,k}$, with $\omega_{R,k} = [m \cdot P_R(k)]^{-\beta} / \max_i\{\omega_{R,i}\}$.
      Save the transition to $\mathcal{B}_\mathcal{R}$ and set $\lambda = \lambda + 1$.
    else if $l_k = 1$ **then**
      Label the transition by setting $l_k = 3$.
      Calculate $\omega_k = \omega_{R,k}$ with $\omega_{R,k} = [m \cdot P_R(k)]^{-\beta} / \max_i\{\omega_{R,i}\}$.
      Modify the ISW to be: $\omega_k = \omega_{T,k} + \omega_{R,k}$.
    else
      Discard the transition and re-sample (go to 7).
    end if
  end while
  Build $\mathcal{B}$ by concatenating $\mathcal{B}_\mathcal{T}$ and $\mathcal{B}_\mathcal{R}$.
end for

estimation biases induced by the distribution mismatching. Different from PER, the ISWs in CER are related to the duplicate conditions of the corresponding transitions. Fig. 3 gives an insight into this issue. In fact, CER essentially constructs two parallel groups of transitions by $\mathcal{P}_T$ and $\mathcal{P}_R$, i.e., Group T and R as illustrated in Fig. 3. And there is no internal duplicate in either Group T or R. Thus, for the transitions that are only in Group T or R (e.g., the transition j and k in Fig. 3), their ISWs are directly decided by their sampling probabilities:

$$\begin{aligned} \omega_j &= \omega_{T,j},\ \omega_{T,j} = \frac{[m \cdot P_T(j)]^{-\beta}}{\max_i\{\omega_{T,i}\}} \\ \omega_k &= \omega_{R,k},\ \omega_{R,k} = \frac{[m \cdot P_R(k)]^{-\beta}}{\max_i\{\omega_{R,i}\}} \end{aligned} \tag{11}$$

Then, for the transitions that are in both Group T and R (e.g., the transition i in Fig. 3), their ISWs are linear combinations of the original ISWs in Eq. (11), as given in Eq. (12).

$$\omega_i = \omega_{T,i} + \omega_{R,i} \tag{12}$$

It should be emphasized again that, no matter how many transitions are repeated in the two groups, CER ensures the resulting mini batch $\mathcal{B}$ always has a fixed size and all the transitions in $\mathcal{B}$ are different from each other.

III. Control of Wind Farm

We consider a wind farm with N individual wind turbines denoted by $\mathcal{WT}_1$, $\mathcal{WT}_2$, ..., $\mathcal{WT}_N$, respectively. Based on the actuator disc theory, the power generated by $\mathcal{WT}_i$ can be calculated by

$$E_i = \frac{1}{2}\rho A_i U_i^3 C_p(a_i, \gamma_i) \tag{13}$$

where ρ is the air density and A_i is the rotor area. Also, U_i denotes the wind speed at $\mathcal{WT}_i$, which is related to the free-stream wind speed U_∞ in front of the wind farm and also the wake effects among turbines. Besides, C_p is called the power coefficient, which is a function of the induction factor a_i and the yaw angle offset γ_i (with respect to the wind direction), and here a_i is related to the rotor's rotation speed and pitch angle. Particularly, one has

$$C_p(a_i, \gamma_i) = 4\eta a_i(1 - a_i)^2 f(\gamma_i) \tag{14}$$

where η is a positive constant, and the function $f(\gamma_i)$ describes the relationship between γ_i and C_p, which will be approximated by a neural network in this work.

By Eq. (13), one can change the turbine's power production by adjusting a_i and γ_i. In conventional control strategy, every wind turbine in the wind farm aims to maximize its own power production. Based on the actuator disc theory, the Nash equilibrium of this game problem (solved by Eqs. (13) and (14)) is $a_i = 1/3$ and $\gamma_i = 0$. This control strategy is commonly referred to as the greedy strategy in the literature [27], [28], [29], [35]. However, as illustrated in Figs. 1.b and 1.c, the wake of upstream turbines have influence on the effective wind speed U_i at the downstream turbines. Therefore, the greedy strategy cannot maximize the overall output of the whole farm: $E = \sum_{i=1}^{N} E_i$.

In this work, we design a yaw control method to achieve wake steering and optimize the farm-level power production. We set the turbines' induction factors to be consistent with the greedy strategy. On this basis, C_p is only related to γ_i. We emphasize that the wake is a complex phenomenon with stochastic properties, and accurately modeling the wake effect is still an open problem. Thus, the mapping from the yaw settings $\{\gamma_1, ..., \gamma_i, ..., \gamma_N\}$, the free-stream wind speed U_∞, and the time t to the total power production is unknown. This mapping is formalized as follows.

$$E = h(\gamma_1, \gamma_2, ..., \gamma_N, U_\infty, t) \tag{15}$$

It also brings some severe barriers for us to directly employ E as reward in CER-DDPG: 1) As indicated by Eq. (15), E is related to U_∞, while U_∞ is time-varying (through its mean

speed is relatively stable). Therefore, a new reward signal that can adapt to the change of U_∞ should be considered. 2) Wake is a dynamic process since it slowly propagates at the wind farm. Therefore, as indicated by Eq. (15), E can change with t even under steady $\{\gamma_1, ..., \gamma_i, ..., \gamma_N\}$ and U_∞. Thus, employing instantaneous power outputs to construct rewards is improper.

A reward regularization process is employed in CER-DDPG to address the problems mentioned above. First, we note that the front wind turbines (i.e., the most upstream turbines in the farm, denoted as F_i with a total number K) are not affected by wake effects, and their greedy power outputs can show the change of U_∞:

$$E_{F_i}^g = \frac{1}{2}\rho A_{F_i} U_\infty^3 C_p(\frac{1}{3}, 0) \tag{16}$$

Thus, $E_F^g = \sum_i^K E_{F_i}^g$ can be employed to normalize E, making E robust to the change of U_∞. Then, for every transition, we utilize the mean power output in a period of time after every action to construct rewards. This can effectively alleviate the influence induced by the wake propagation process, making the whole problem quasi-Markovian. Moreover, large yaw offsets can result in loads and fatigues to the structure of turbines, thus penalty items should also be introduced into the reward function to mitigate this problem.

Based on these observations and analyses, the regularized reward is designed to be

$$r_t = \frac{1}{t_a}\int_t^{t+t_a} [\frac{E(h)}{\kappa_f \sum_i^K E_{F_i}^g(h)} - \kappa_\gamma \sum_{i=1}^N \gamma_i(h) - \kappa_r]\mathrm{d}h \tag{17}$$

where t_a denotes the time period for averaging, and κ_r, κ_f, and κ_γ are user-defined gains for scaling, weighting and offsetting purposes. A remaining issue is that $E_{F_i}^g$ is not measurable during the real-time control process, and we need to estimate it through γ_i and $E_{F_i}^{\gamma_i}$. Based on (14), we formalize their relationship by

$$E_{F_i}^{\gamma_i} = E_{F_i}^g f(\gamma_i) \tag{18}$$

Some studies [27], [28], [29], [35], [36] utilize different empirical formulas to approximate $f(\gamma_i)$. A commonly-used method [28], [36] is to employ a cosine function to correct the power loss induced by the yaw offset γ_i:

$$E_{F_i}^{\gamma_i} = E_{F_i}^g \cos(\gamma_i)^{p_P} \tag{19}$$

where p_P is a positive constant to be identified by data.

In this paper, a more general method is designed to map the relationship in (18) (i.e. approximate $f(\gamma_i)$). Specifically, a neural network is employed, which takes γ_i and $E_{F_i}^{\gamma_i}$ as inputs and the estimates of $E_{F_i}^g$ as outputs. The performance of this method in comparison with (19) is provided in Sec. IV.

Moreover, we also take into account the dynamics of yaw actuators:

$$\ddot{\gamma}_i = z(\dot{\gamma}_i, \gamma_i) + g(\dot{\gamma}_i, \gamma_i, u_i) \tag{20}$$

where the specific expression of z and g are unknown for controller design. And u_i denotes the control input (usually in terms of torque or force), which should be within $[u_{i,\min}, u_{i,\max}]$. An assumption regarding the dynamics in Eq. (20) is that $\gamma_i^+, \dot{\gamma}_i^+ \in \mathcal{L}_\infty$ if $\gamma_i, \dot{\gamma}_i, u_i \in \mathcal{L}_\infty$, where γ_i^+ and $\dot{\gamma}_i^+$ respectively denote the responses of γ_i and $\dot{\gamma}_i$ after taking the action u_i. We note that most of the yaw actuators satisfy this assumption. Moreover, since the response of yaw actuators is much quicker than the wake propagation process, a stable γ_i^+ with $\dot{\gamma}_i^+ = 0$ can be expected at the end of each learning step.

Algorithm 2 CER-DDPG for wind-farm control

Initialize the parameters θ^μ, $\theta^{\mu'}$, θ^Q, and $\theta^{Q'}$ for NNs, and set the learning rates.
Decide the size of the mini batch $\mathcal{B}$ and the memory buffer $\mathcal{M}$, i.e., n and m.
Initialize other user-defined parameters involved in the algorithm, include: ξ, τ, α, β, η, ϵ, κ_r, κ_f, and κ_γ.

for each episode **do**
  Observe the initial state s_0.
  for $t = 0$ to T **do**
    Choose the action $a_t = \mu(s_t|\theta^\mu) + \phi_t$, where ϕ_t is the exploration noise.
    Apply a_t to the wind farm, calculate the reward r_t based on Eq. (17), and observe s_t^+.
    Store the transition (s_t, a_t, s_t^+, r_t) and its priorities in $\mathcal{M}$.
    if training = TRUE **then**
      Sample the mini batch $\mathcal{B}$ based on the CER in Algorithm 1.
      Update the parameter θ^Q of the main critic network by minimizing the loss L as in Eq. (4).
      Update the parameter θ^μ of the main actor network by the policy gradient strategy in Eq. (6).
      Update the parameters ($\theta^{Q'}$ and $\theta^{\mu'}$) of the target critic and actor networks by the soft replacement strategy in Eq. (3).
      Update the priorities for all the transitions in $\mathcal{B}$.
    end if
  end for
end for

Now we are ready to fit the CER-DDPG framework into the wind-farm power control problem. We aim to maximize a long-term reward: $R_{t_I} = \sum_{t=t_I}^\infty \gamma^{t-t_I} r_t$ from any start time $t = t_I$, with r_t is calculated by Eq. (17). The objective of CER-DDPG is to develop a control policy $\mu(s)$, which can decide the constrained control signal $a = \{u_1, u_2, ..., u_N\}$ in real time, based on the state $s = \{\gamma_1, \gamma_2, ..., \gamma_N\}$. This algorithm is summarized in Algorithm 2 in detail. Please also refer to Algorithm 1 for notations and the specific design of CER.

IV. Numerical Simulations

A. WFSim Introduction and General Simulation Settings

We test the performance of the proposed CER-DDPG on the wind farm simulator (WFSim) [31]. WFSim is open-access and its code & data can be found in [37]. WFsim is a dynamic wind farm model with high computational efficiency, making it suitable for case study purposes. Briefly, WFSim models the

Table I: Simulation Settings.

Properties & Parameters	value
Layers of actor and critic	3
Hidden-layer neurons in actor and critic	32
Soft replacement rate, τ	0.01
Discount factor, ξ	0.95
Initial, incremental and maximal values of α	0, 0.00025, 0.5
Initial, incremental and maximal values of β	0.4, 0.001, 1
Sizes of the memory buffer and the sampling batch	10000, 128
Layers of the reward regularization module	3
Hidden-layer neurons in reward regularization	20
Y_I (kg·m^2)	38.9
Y_D (N·m·s)	102.2
Y_S (N·m)	85.9
$u_{i,\min}$, $u_{i,\max}$ (N·m)	-1.5, 1.5
Yaw angle range (deg)	[-30, 30]
κ_r, κ_f, κ_γ	1, 1.8, 0.1
η, ϵ	5, 0.01

wind flow by utilizing the 2D Navier-Stokes equations [31], [38], [39]:

$$\begin{aligned} \rho\frac{\partial u}{\partial t}+\rho\nabla(u\varpi) &= -\frac{\partial p}{\partial x}+\nabla(\sigma\nabla u)+S_x+T_x \\ \rho\frac{\partial v}{\partial t}+\rho\nabla(v\varpi) &= -\frac{\partial p}{\partial y}+\nabla(\sigma\nabla v) \\ \rho\nabla(\varpi) &= 0 \end{aligned} \tag{21}$$

where ρ and σ are respectively the air density and viscosity, p denotes the pressure field, and $\varpi=[u,v]$ denotes the velocity vector field. Besides, T_x is the turbulence model, and the wind turbine models are incorporated into Eq. (21) by S_x. Then, Eq. (21) is discretized by employing a hybrid differencing scheme, over a staggered grid of $(N_x \times N_y)$ cells.

The main settings for the simulations are summarized in Table I. We consider a wind farm with six turbines. For this wind farm, we set the critics and actors in our CER-DDPG to be three-layer fully-connected NNs. Their hidden layers have 32 neurons and use the relu function as the activation function. The soft replacement rate for the target critic and actor is $\tau = 0.01$. The discount factor is $\xi = 0.95$. For CER, the ratio α is initially set to be 0, then it is slowly increased with a small increment $\delta\alpha = 0.00025$ for every training step. The maximum of α is set to be 0.5. The size of the memory buffer $\mathcal{M}$ is 10000 with 128 transitions being selected from $\mathcal{M}$ to form the mini batch $\mathcal{B}$ in every training step. There are 200 steps in each training episode. At the beginning of each episode, all turbines follow the greedy strategy. We use Eqs. (7) and (8) to design priorities. Moreover, the model of the yaw actuator follows the FAST code designed by NREL (National Renewable Energy Laboratory of US) [40]:

$$Y_I\ddot{\gamma}_i(t)+Y_D\dot{\gamma}_i(t)+Y_S\gamma_i(t)-\upsilon_i(t)=0,\ i=1,2,...,N \tag{22}$$

where Y_I, Y_D, and Y_S are respectively the inertia, torsional damping constant, and torsional spring stiffness. They are chosen to be: $Y_I = 38.9$kg·m^2, $Y_D = 102.2$N·m·s, and $Y_S = 85.9$N·m. And υ denotes the cumulative torque applied to the yaw actuator. It should be emphasized that Y_I, Y_D, and Y_S are all unknown, and the control policy aims to decide the change of $\upsilon(t)$ at every time step, i.e.,

$$\upsilon_i(t+1)=\upsilon_i(t)+u_i(t) \tag{23}$$

Moreover, the bounds of u_i is set to be $u_{i,\min} = -1.5$N·m and $u_{i,\max} = 1.5$N·m, $i = 1, 2, ..., N$.

Based on these settings, simulations under different wind-farm layouts, wind-turbines types and environmental conditions are given in the following subsections, providing comprehensive evaluation of the proposed CER-DDPG method.

B. Simulation Results with the NREL 5MW Wind Turbines

In the subsection, six NREL 5MW wind turbines are employed to construct the wind farm. The flow field in WFSim is set to be 2040m×1050m, which is discretized into 80×40 spatial grids. Two scenarios are considered here, as qualitatively illustrated in Fig. 4, where $D = 126.4$m is the turbines' rotor diameter. The difference between these two scenarios is that the turbines in the scenario 2 are moved in the crosswind direction to reduce overlaps among the turbine rotors. Following relevant studies [28], [29], we restrict turbines' yaw offsets (with respect to the free-stream wind direction) to $[-30°, 30°]$ per safety requirements, and we set the clockwise direction as the forward direction of yaw maneuvers.

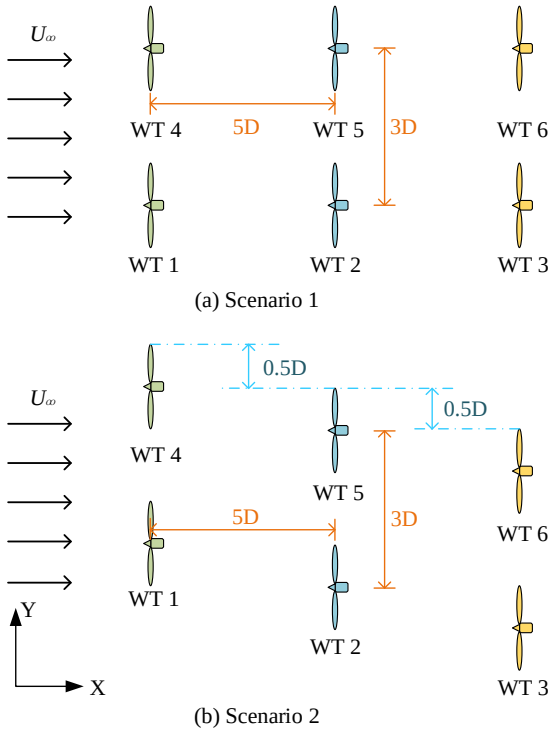


Figure 4: Illustration of the two layouts of wind farms.

As discussed in Sec. III, a neural network (NN) is required for reward regularization purposes in the CER-DDPG. To this end, the power outputs of the front turbines under different wind conditions ($U_\infty = 8, 9, 10$m/s) are collected. Then a fully connected NN with a single hidden layer is employed. The

hidden layer has 20 neurons and employs the sigmoid function as its activation function. The input of this NN is the yaw angle: $x_i = \gamma_i$, and the output is set to be: $y_i = E^{\gamma_i}_{F_i}/E^{g}_{F_i}$. A total of 90 training pairs (x_i, y_i) are collected, and the training results are given in Fig. 5. There are two main facts that can be observed in Fig. 5: 1) one can see that the ratio $E^{\gamma_i}_{F_i}/E^{g}_{F_i}$ is irrelevant to the wind speed, and it is only decided by γ_i. 2) The NN can successfully map the relationship between γ_i and $E^{\gamma_i}_{F_i}/E^{g}_{F_i}$. Thus, by employing this NN and the measurements of γ_i and $E^{\gamma_i}_{F_i}$, one can accurately estimate $E^{g}_{F_i}$ without any knowledge of the time-varying free-stream wind speed U_∞ and use it to construct the regularized reward in Eq. (17).

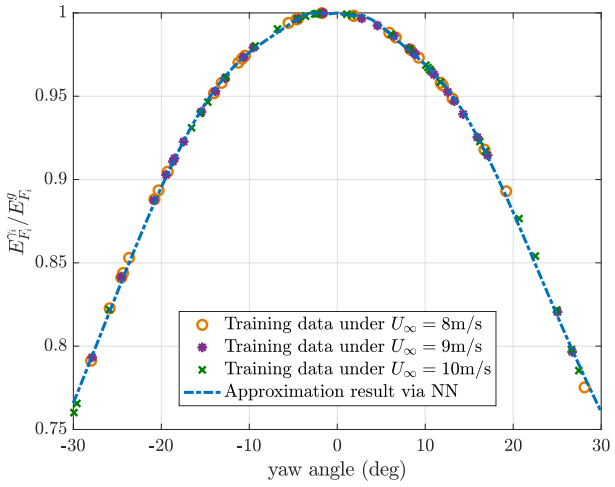


Figure 5: Approximation result via NN.

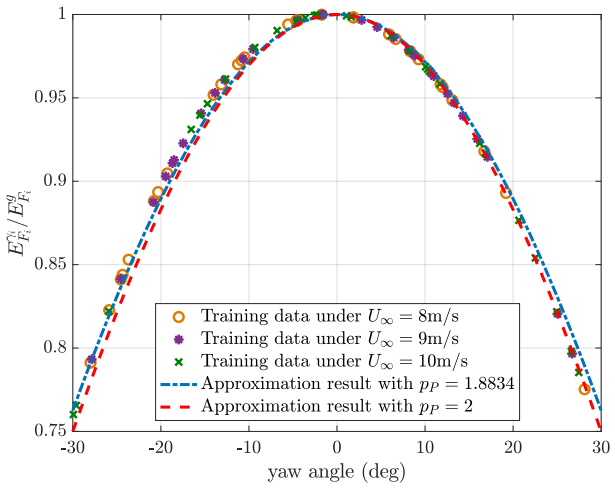


Figure 6: Approximation result via empirical formula.

We also compare the performance of our method with the commonly-used approximation method in (19). Fig. 6 illustrates the approximation results via (19) under different p_P: 1) $p_P = 2$, which follows the experimental result in [36]; 2) $p_P = 1.8834$, which is deduced by the least-square method to minimize the root mean square (RMS) error with respect to training data. It is noteworthy that the result $p_P = 1.8834$ also fits perfectly with [28] - the authors set $p_P = 1.88$ in this reference to fit high-fidelity simulation data. From Figs. 5 and 6, one can clearly see that the performance of our NN is superior to the approximation method in (19). The RMS error with respect to training data are 3×10^{-4} under our NN, while that of the approximation method in (19) are 8×10^{-3} and 5×10^{-3} under the settings $p_P = 2$ and $p_P = 1.8834$, respectively.

The free-stream wind speed U_∞ is changing within $[8, 10.5]$m/s during the training process of CER-DDPG, driven by a random-walk noise. In addition, for comparison purposes, the original DDPG and the PER-DDPG are also employed to carry out simulations. The simulation scenario with the layout 1 follows the settings given in Table I. It is observed that all three deep RL approaches can increase the total power production through wake steering. This fact is shown in Fig. 7, which illustrates the changes of the cumulative reward per episode under different algorithms. For ease of analysis, the cumulative rewards are normalized by the maximum reward observed during the learning process of all algorithms. One can see that, though all the three algorithms can finally approximate the maximum episode reward, the CER-DDPG proposed in this paper accelerates the learning process: it almost gets the maximum episode reward after only 100 learning episodes.

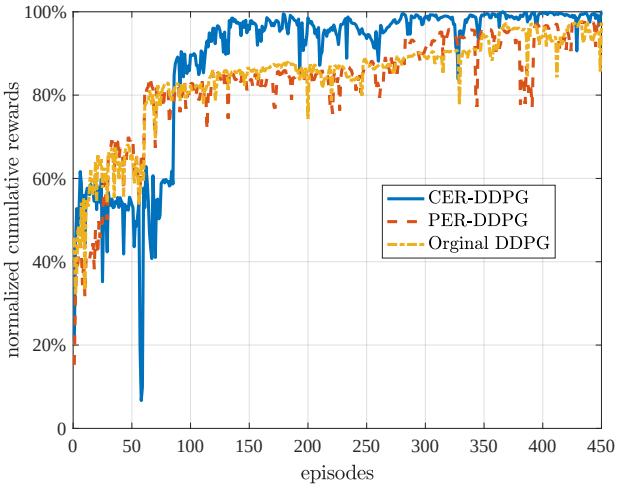


Figure 7: Normalized cumulative rewards under different algorithms (scenario 1).

Then, we analyze the performance of CER-DDPG in detail. To this end, we use the CER-DDPG networks (after 200 training episodes) to perform the closed-loop control for the wind farm. The initial condition is set to be $\gamma_i(0) = 0$ and $\dot{\gamma}_i(0) = 0$, $i = 1, 2, 3, 4, 5, 6$, and the control signals (i.e., u_i) are updated by CER-DDPG per 2 seconds. The initial wind speed is $U_\infty = 10.5$m/s. The time responses of γ_i and u_i of all turbines (denote by WT$_i$, $i = 1, 2, 3, 4, 5, 6$) are given in Fig. 8. One can see that our CER-DDPG successfully drives the yaw actuators to achieve wake steering with limited control inputs, and the resulting yaw angles after 500s are around $\{24.9511, 18.1793, 0.1785, 21.0259, 20.8083, 0.7567\}$deg.

Moreover, the total power productions during the wake steering process are illustrated in Fig. 9. To clearly show the relative changes of the power production, the data in this figure has been normalized by the standard power output under the

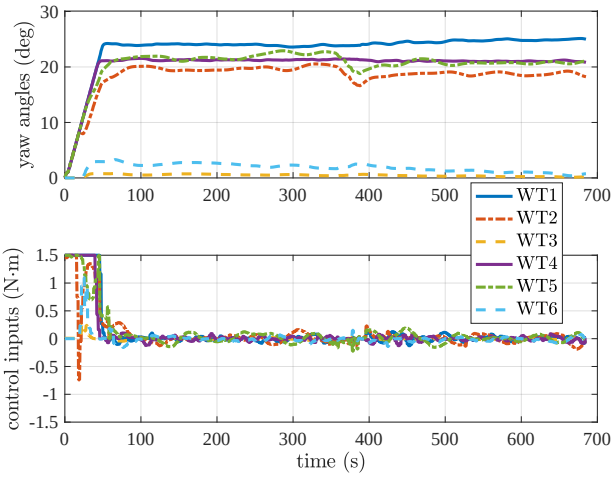


Figure 8: Simulation results of yaw angles and control inputs under CER-DDPG (scenario 1).

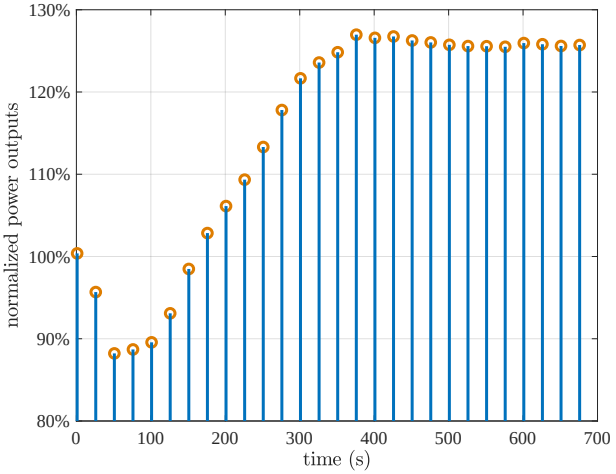
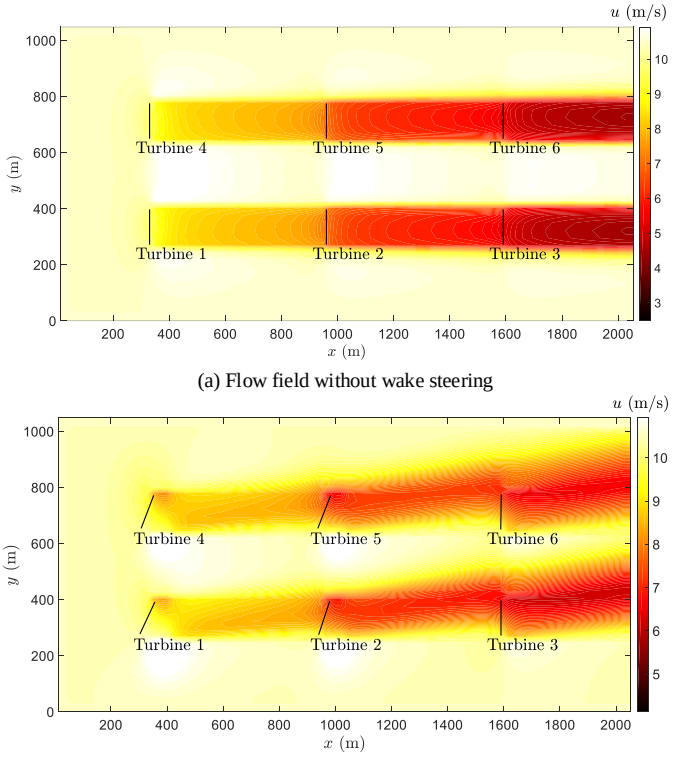


Figure 9: Normalized power productions under CER-DDPG (scenario 1).

greedy strategy. It is noteworthy that wake effects have time-delayed features. One can see that, before the wakes under new yaw settings are fully propagated (the first 300s in the simulation as shown in Fig. 9), the total power output is decreased due to yaw offsets. Then this situation is rapidly improved after 300s, and a clear increase (over 25%) for the total power production can be obtained. This phenomenon also shows a key feature of reinforcement learning - instead of pursuing a high instantaneous or short-time reward (like the greedy strategy), the reinforcement learning agent aims to maximize a long-term reward. The flow fields under both the CER-DDPG and the greedy strategy at 700s are given in Fig. 10. One can see that CER-DDPG successfully mitigates the wake effects of upstream turbines with respect to downstream turbines, rendering the increase of the total power production.

Then we test our algorithm under scenario 2. As shown in Fig. 4.b, the positions of turbines are varied at Y direction in this scenario. Due to the special layout, excessive penalties for yaw offsets will hinder wind farm control in this case. Therefore we reduce κ_f from 0.1 to 0.05 and increase η



(a) Flow field without wake steering

(b) Flow field with wake steering by CER-DDPG

Figure 10: Flow fields with and without wake steering (scenario 1, at $t = 700$s).

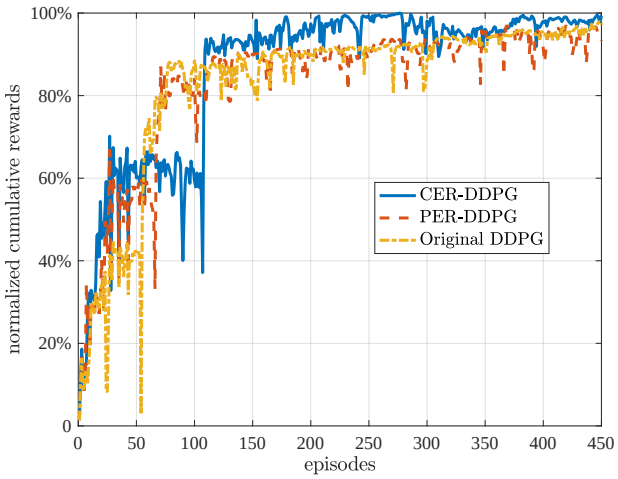


Figure 11: Normalized cumulative rewards under different algorithms (scenario 2).

from 5 to 20 to keep 'inceasing total power maximization' still as the first-priority task in our wind farm control. All the other simulation settings follow Table I. The normalized cumulative rewards under the CER-DDPG, PER-DDPG, and original DDPG are given in Fig. 11. One can see that the CER-DDPG algorithm proposed in this paper still has better learning efficiency and effectiveness - it achieves higher cumulative rewards than the other two methods after 100 episodes. Same as scenario 1, we also test our CER-DDPG under a closed-loop control problem. The results are given in Figs. 12 and 13.

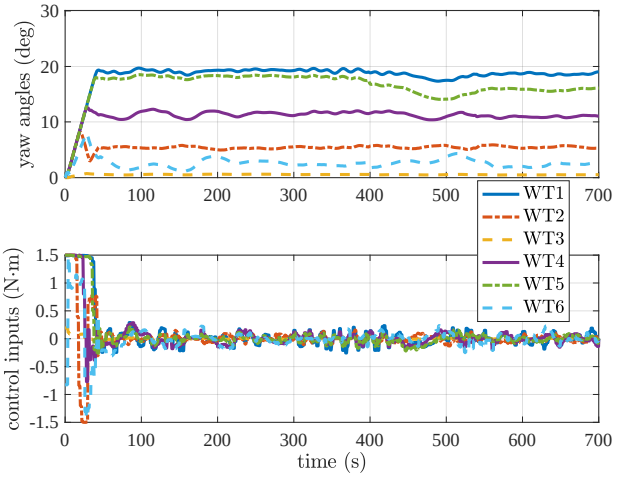


Figure 12: Simulation results of yaw angles and control inputs under CER-DDPG (scenario 2).

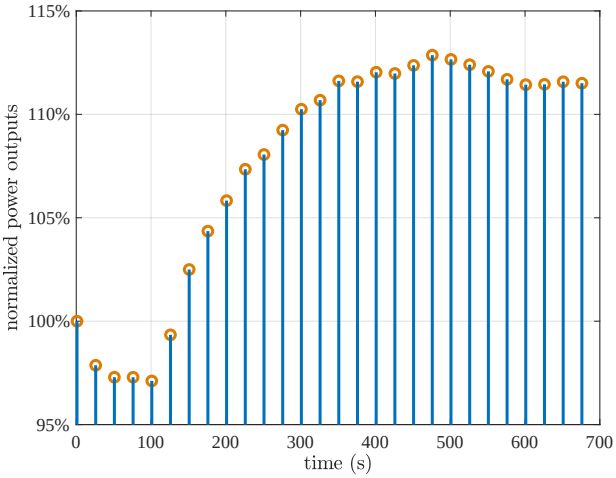
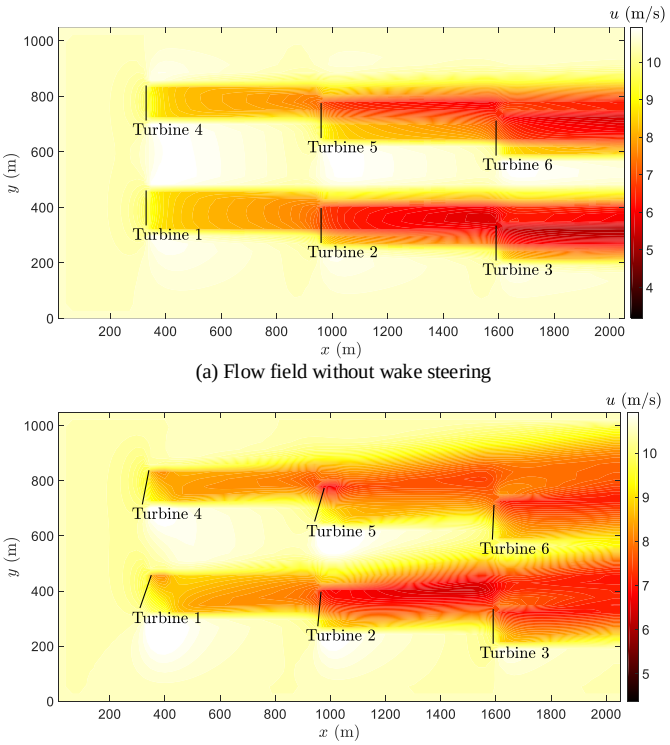


Figure 13: Normalized power productions under CER-DDPG (scenario 2).

After the wakes are fully propagated, CER-DDPG can achieve an 11% growth for the total power production, compared with the greedy strategy. The flow fields at 700s are given in Fig. 14. As shown in Fig. 14, the wind farm layout scenario 2 reduces the overlaps among turbine rotors, which helps the downstream turbines suffer less from the wake effects under the greedy strategy. This is why the deep RL algorithm leads to a relatively lower increase for the total power generation when compared with scenario 1.

C. Simulation Results with the DTU 10MW Wind Turbines

To show the adaptability of the proposed CER-DDPG, we employ a different type of wind turbines - the DTU 10MW wind turbines, to carry out simulation tests in this subsection. The wind-farm layout is set to be same with the scenario 1 in Sec. IV.B, while the rotor diameter of DTU 10MW wind turbine is $D = 178.3$m. The flow field in WFSim is set to be 2674.5m$\times$1248.1m ($15D \times 7D$), and it is discretized into a 100×50 staggered grid. All the settings of our CER-DDPG are kept unchanged from Table I.



(a) Flow field without wake steering

(b) Flow field with wake steering by CER-DDPG

Figure 14: Flow fields with and without wake steering (scenario 2, at $t = 700$s).

For comparison purposes, we also employ the adjoint-based model predictive control (AMPC) method proposed in [22] to carry out simulations and compare the performance and features of our CER-DDPG with AMPC. AMPC is a model-based method that is built upon the 2D Navier-Stokes equations in (21). In [22], AMPC is designed to maximize wind farm power generation via induction control, and we adapt it to a yaw control algorithm in this case study. The core principle of AMPC is to employ a discrete-time wind farm model deduced by Eqs. (21):

$$E(X_k)X_{k+1} = AX_k + B(X_k)\beta_k + b(X_k) \qquad (24)$$

where the state variable X_k stacks all the longitudinal and lateral flow velocities and the scaled pressure components in every cell of the staggered grid $(N_x \times N_y)$ [22], and β_k denotes the control input vector. One can refer to [22] for the detailed expression of E, A, B, b and also other design details of AMPC. It should be emphasized that Eq. (24) is actually the inherent mathematical model of WFSim. Thus there are no modelling errors and uncertainties when applying AMPC to WFSim. This feature allows AMPC to have the potentially best control performance that can be achieved with WFSim, rendering AMPC a suitable method and an excellent baseline to carry out comparison with our CER-DDPG.

We set the state & control input constraints of AMPC to be the same with CER-DDPG. We test the performance of our CER-DDPG (after 200 training episodes) and AMPC with the DTU 10MW wind turbines. Simulation results of yaw angles are given in Fig. 15. It is interesting to see that these

two controllers converge to similar yaw offsets at the end of simulations, and this result is also validated by the flow fields (at $t = 700$s) as shown in Figs. 16. To be specific, after 500s, the turbine yaw angles under CER-DDPG are around $\{26.34, 18.08, 0.00, 25.24, 25.25, 9.09\}$ deg, and those under AMPC are $\{26.49, 19.11, 0.10, 25.74, 25.62, 12.94\}$ deg. The normalized power outputs (with respect to the greedy strategy) under CER-DDPG and AMPC are illustrated in Fig. 17. One can see that both control methods lead to clear power generation increases (over 24%) with respect to the greedy strategy.

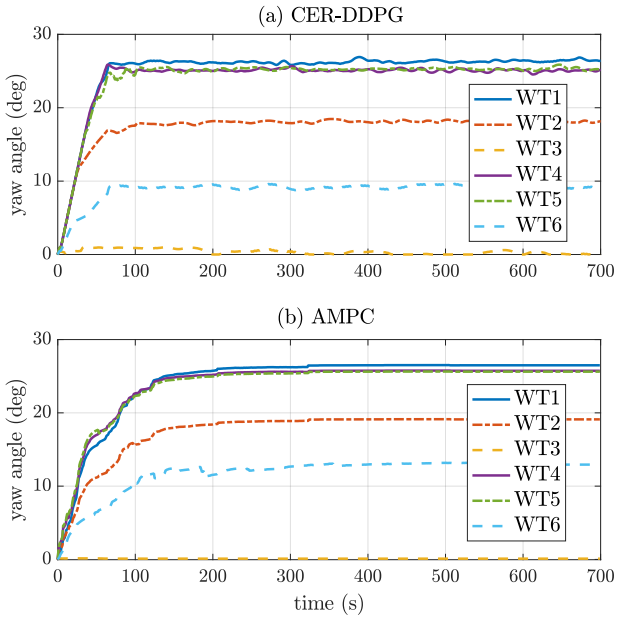


Figure 15: Yaw angle changes under CER-DDPG and AMPC with DTU 10MW wind turbines.

As discussed above, since AMPC directly employs the inherent mathematical model of WFSim, it is not influenced by modelling errors & uncertainties and can potentially achieve the best control performance with WFSim. Simulation results indicate that our CER-DDPG achieves similar performance to AMPC - showing the effectiveness and the optimizing ability of our CER-DDPG.

Though both CER-DDPG and AMPC can successfully achieve wake steering and increase the farm's total power generations, they have different features with each other, as summarized in Table II. As a model-based method, AMPC employs the wind conditions in every cell of the staggered grid at every time step to predict future states, allowing AMPC to capture the whole-flow-field information and rendering relatively smoother yaw-change curves with fewer fluctuations when compared with CER-DDPG. However, it is not feasible to accurately measure the wind conditions in every cell of a pre-determined staggered grid for a flow field (e.g., a 100×50 grid is employed for a 2675.5m$\times$1248.1m flow field in this case study) in practice. Moreover, such full-state-style information of the flow field also leads to a large number of system states (over ten thousand states are employed in AMPC in this case study), resulting in a high computational complexity

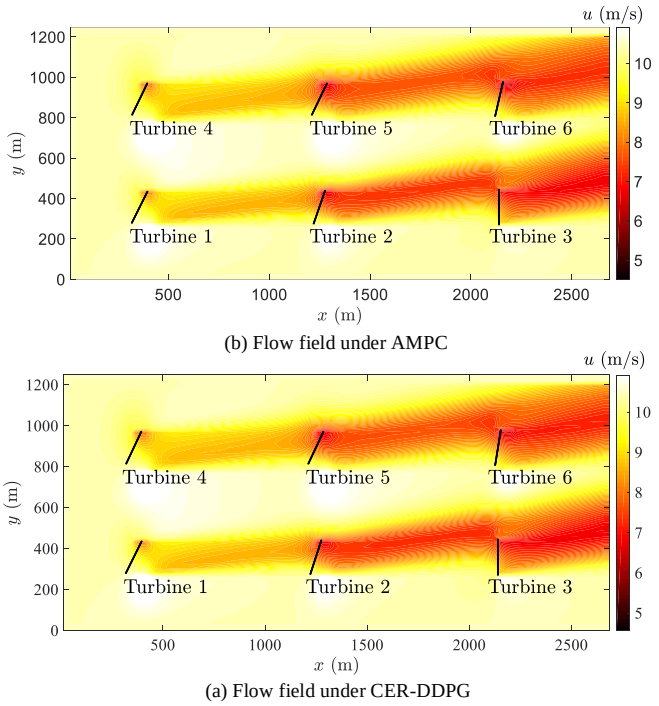


Figure 16: Flow fields under CER-DDPG and AMPC with DTU 10MW wind turbines (at $t = 700$s).

in real-time closed-loop control. In contrast, CER-DDPG is model-free and only requires turbines' power outputs to carry out learning, avoiding the use of impractical measurements and leading to enhanced applicability and generality. But similar to other data-driven RL algorithms [9], [10], [32], CER-DDPG still requires a relatively large amount of data for offline training purposes, which usually leads to a high offline computational complexity. In contrast, MPC has no offline cost but needs to solve a complex optimal control task at every time step in real-time control, resulting in a high computational cost online.

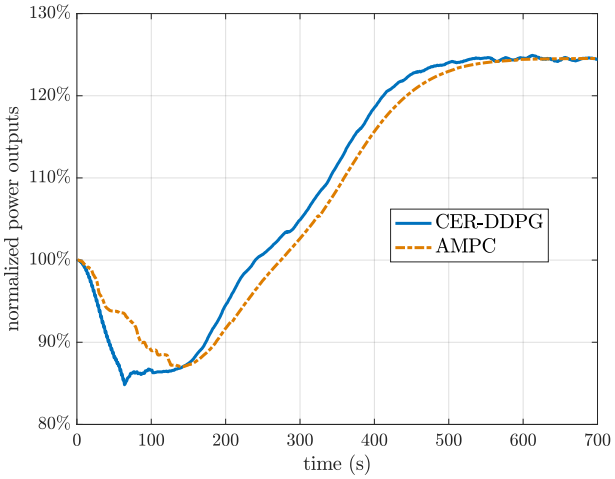


Figure 17: Normalized power outputs under CER-DDPG and AMPC with DTU 10MW wind turbines.

Table II: Comparison of AMPC and CER-DDPG.

	AMPC in [22]	CER-DDPG
Optimizing Ability	√	√
Model dependence	model-based	model-free
Online Complexity	high	low
Offline Complexity	low	high
Requirement of full-flow states	√	×

D. Simulations under Different Environmental Conditions

In this subsection, additional simulation results under different environmental conditions are provided to show the robustness of the CER-DDPG method proposed in this paper. In the following case studies, The wind farm specifications (with DTU 10MW wind turbines) and the settings of CER-DDPG are all kept same with Table I and Sec. IV.C.

1) *Simulations under Time-Varying Wind Speeds.* As discussed in Sec. III, a reward regularization module is employed in CER-DDPG to adapt to different wind speeds during the training process. To test its performance under time-varying wind speeds in real-time control, we consider a wind profile as illustrated in Fig. 18. The normalized power outputs (with respect to the greedy mode under the initial wind speed at $t = 0$s) are illustrated in Fig. 19. The CER-DDPG method proposed in this paper leads to clear power generation increases (23.1% on average after 300s) when compared with the greedy strategy, showing its effectiveness and robustness to time-varying wind speeds. The flow fields under the greedy strategy and CER-DDPG are illustrated in Fig. 20.

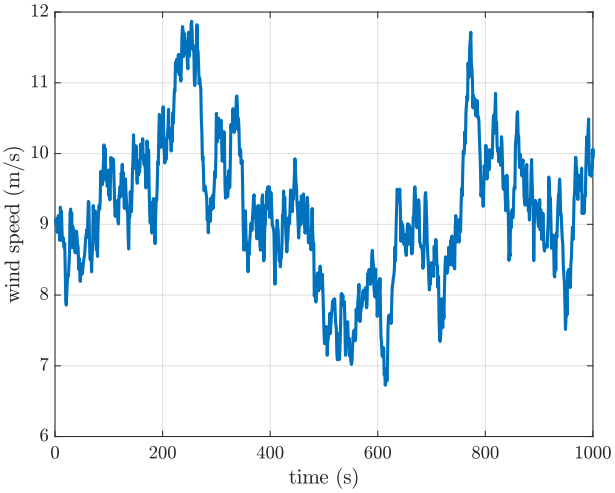


Figure 18: Wind speed changes.

2) *Simulations under Time-Varying Wind Directions & Speeds.* In this paper, CER-DDPG is trained under a main wind direction, which is a common practice for wind farm yaw control [28], [29], [30]. Here we test the robustness of CER-DDPG to fluctuating wind directions around the main wind direction, while the changes of wind speed are shown in Fig. 18. We follow the settings given in [22] - the wind direction varies from the main wind direction (i.e. x-direction) in the interval of $\pm 8^{o}$ per every 100s, as shown in Fig. 21. The

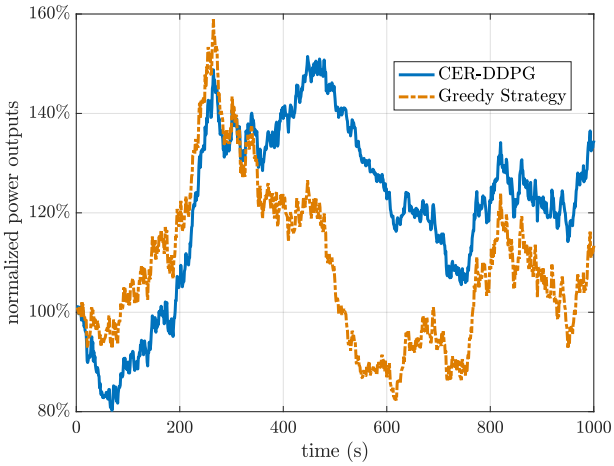
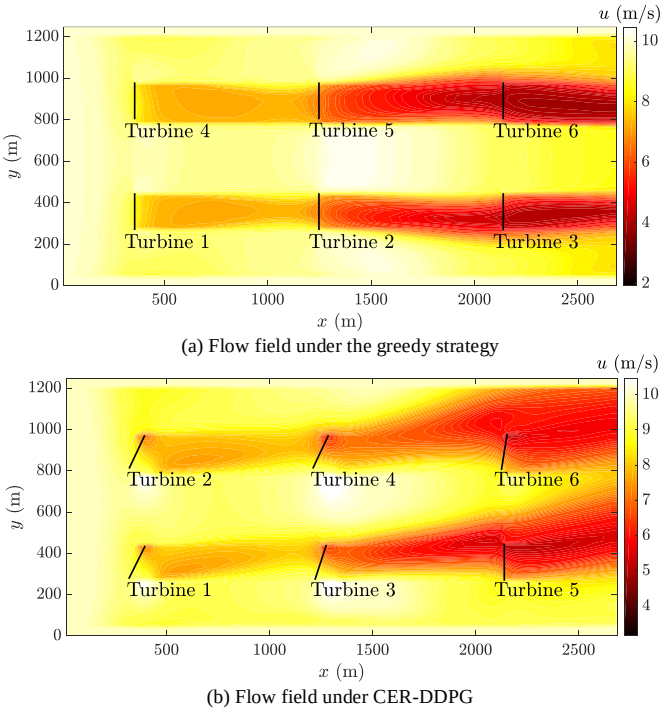


Figure 19: Power outputs under time-varying wind speeds.



(a) Flow field under the greedy strategy

(b) Flow field under CER-DDPG

Figure 20: Flow fields under time-varying wind speeds (at $t = 1000$s).

flow fields at $t = 1000$s under the greedy strategy and CER-DDPG are illustrated in Fig. 22. One can see that, wakes have irregular features in this simulation scenario. The normalized power generations (with respect to the power generation at $t = 0$s under the greedy mode) are given in Fig. 23. It shows that our CER-DDPG approach still leads to higher generations than the greedy strategy.

Although simulation results indicate that our CER-DDPG has a certain level of robustness to wind direction fluctuations, it should be emphasized that such robustness is validated only when the wind direction is fluctuating around the main wind direction within a limited range. The algorithm requires to be re-trained if the main wind direction changes drastically. To demonstrate that CER-DDPG can adapt to different main

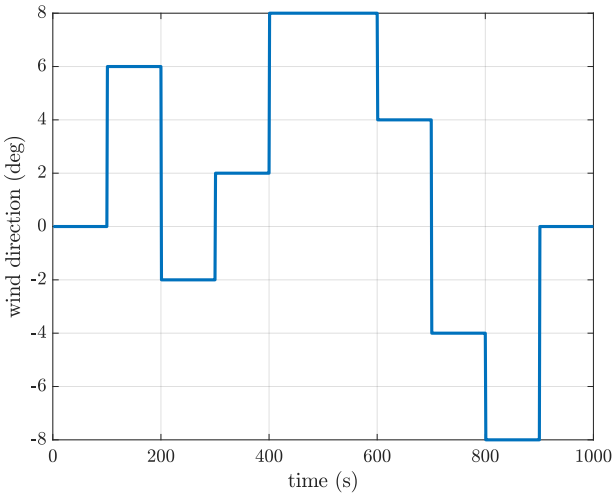


Figure 21: Wind direction changes.

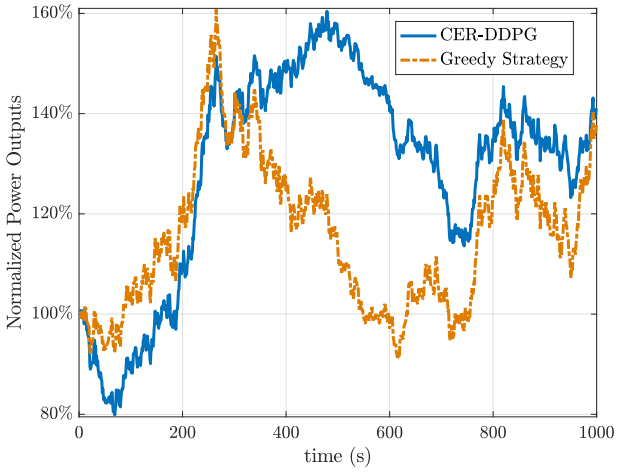
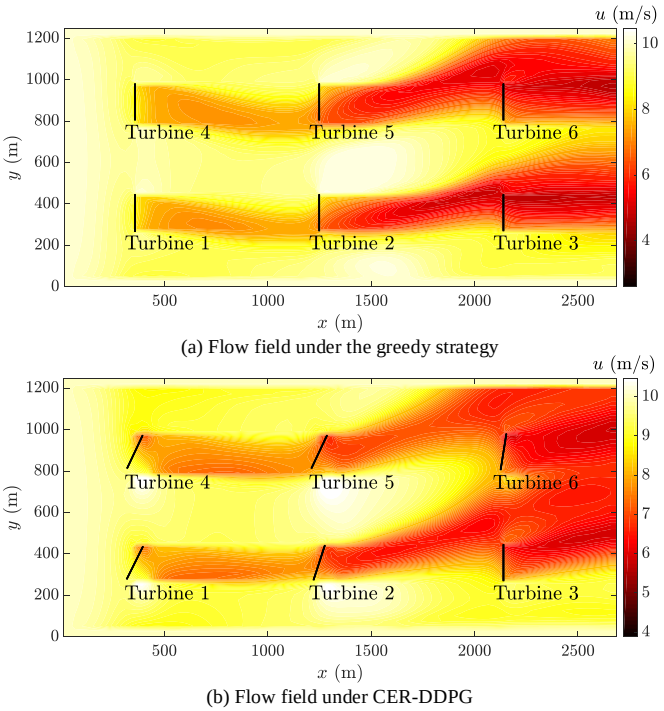


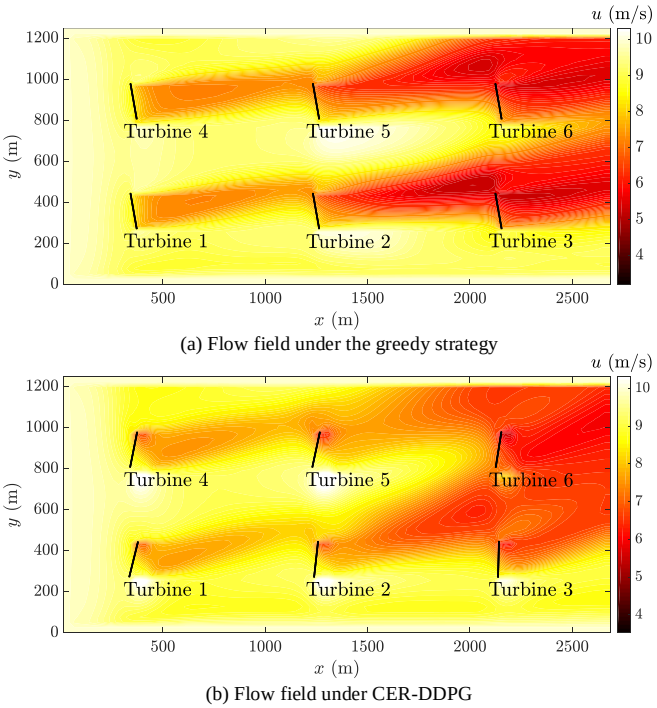
Figure 23: Power outputs under time-varying wind directions & speeds.



(a) Flow field under the greedy strategy

(b) Flow field under CER-DDPG

Figure 22: Flow fields under time-varying wind directions & speeds (at $t = 1000$s).



(a) Flow field under the greedy strategy

(b) Flow field under CER-DDPG

Figure 24: Flow fields under the changed main wind direction with time-varying wind speeds (at $t = 1000$s).

wind directions, we employ a new scenario in which the main wind direction varies 10 degrees counterclockwise from the x-direction, and the changes of wind speed still follow Fig. 18. The flow fields under the greedy method and CER-DDPG with this new main wind direction are provided in Fig. 24. After 500s, the turbine yaw offsets (with respect to the new main wind direction) under CER-DDPG are around {24.02, 21.27, 16.29, 22.09, 11.43, 19.06} deg. The normalized power outputs with respect to the greedy strategy are given in Fig. 25, which illustrate that an average increase of 26.3% is achieved after the wake changes are fully propagated, showing that CER-DDPG can adapt to different main wind directions.

3) *Simulations under Different Turbulence Levels.* In WFSim, three adjustable parameters: d, d' and l_s, are embedded in the turbulence model to match different turbulence intensities. The detailed definitions and physical meanings of these parameters are given in [31], [41]. In this case study, we carry out Monte Carlo simulations to comprehensively test the CER-DDPG's robustness to these turbulence-related parameters. Following the guide of WFSim and the default settings in WFSim for DTU 10MW wind turbines ($d = 700$m, $d' = 50$m, $l_s = 0.05$), we set the changing ranges of d and d' in Monte Carlo simulations to be $d \in [200, 800]$m and $d' \in [20, 180]$m. We consider three batches of simulations with l_s to be 0.02, 0.05 and 0.08, respectively. Each batch contains

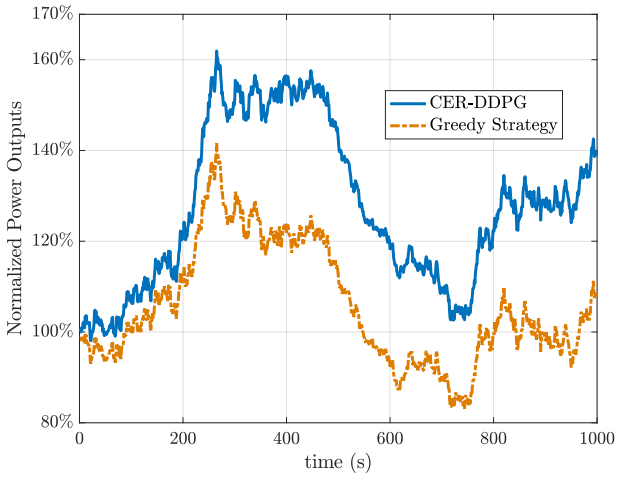


Figure 25: Power outputs under the changed main wind direction with time-varying wind speeds.

1000 separated 1000-second simulations. In these simulations, d and d' are randomly selected from the corresponding ranges, and the changes of wind speed follow Fig. 18. In every simulation, the averaged power generation over the last 500 seconds under the CER-DDPG method is calculated. Its ratio to the corresponding greedy strategy (under the same simulation conditions, including all turbulence-related parameters) is shown in Fig. 26. This figure indicates that CER-DDPG has robustness to different turbulence levels (corresponding to different turbulence-related parameters). A 30.18% power generation increase is achieved on average of all Monte Carlo simulations. Even in the most extreme cases, CER-DDPG can still increase the power generations by approximately 15%.

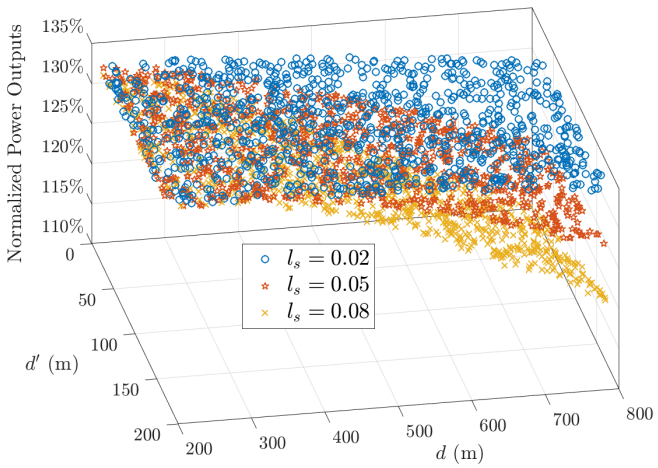


Figure 26: Monte Carlo simulation results under different turbulence settings.

Remark 1: WFSim [31] has been employed to carry out case studies in this section, which is a control-oriented wind farm simulator built upon the Navier-Stokes equations as shown in Eqs. (21) and (24). WFSim makes a trade-off between computational complexity and modelling accuracy. On the one hand, the inherent mathematical model in WFSim is significantly less costly than high-fidelity large-eddy simulations. On the one hand, as a dynamic simulator, WFSim has a much higher accuracy than commonly-used steady-state models/simulators (e.g. FLORIS [28] and Jensen [42]). These features make WFSim an ideal choice for testing wind farm control methods. But the downside is that WFSim can only carry out two-dimension flow field simulations with a degraded accuracy when compared with the full-dimension large-eddy simulations, leading to inevitable simulation errors in comparison with real wind farms.

Remark 2: Similar to many other mode-free deep RL algorithms [9], [10], [32], the CER-DDPG method designed in this paper is independent of the training/testing environment. Its key features (e.g. data-driven and optimizing abilities) are always valid no matter in simulations or in practical use. It should be mentioned that training and validating in the same simulation environment (e.g. the WFSim environment employed in this paper) may result in sub-optimal solutions in practice use. In addition, it should be emphasized that all simulators inevitably have simulation errors in comparison with real wind farms. Thus an online fine-tuning process will be carried out for our CER-DDPG in practical use. Particularly, CER-DDPG can be first pre-trained by a proper simulator or a sufficient set of actual wind farm data to make an initial guess for the optimal control policy. Then, after it is applied to the real wind farm, CER-DDPG can be fine-tuned by the online wind farm data to potentially improve its performance, and the composite experience replay strategy proposed in this paper can accelerate this online fine-tuning process.

Remark 3: Simulations with WFSim verify the effectiveness of the deep RL algorithm proposed in this paper. As analyzed in Introduction and shown in Sec. IV. B, directly employing PER in DDPG does not gain much in the case studies. In contrast, by balancing the two sets of priorities (i.e. TD errors and rewards) and sampling transitions based on the composite priorities, our deep RL method leads to superior performance and better training efficiency than the standard DDPG algorithm and the PER-based DDPG algorithm. Moreover, the whole algorithm design is generic, which has the ability to achieve data-driven model-free control for a large class of complex systems. In addition, the composite experience replay module designed in this paper has a flexible structure. It has the potential to be combined with other strategies, such as the asynchronous advantage actor-critic (A3C) [43] strategy to further improve efficiency via the asynchronous learning process - this will be our future work in this direction.

V. Conclusions

A novel deep RL-based control approach has been proposed in this paper to address wind farm control problems. Specifically, we enhanced the learning efficiency and effectiveness of DDPG through a novel experience replay strategy, i.e. the CER strategy. In CER, we employed both TD-errors and rewards to measure the priorities of transitions, then concurrently sampled transitions based on both sets of priorities. In addition, the modified importance-sampling weights were employed to

anneal the biases introduced by CER. To utilize CER-DDPG to handle the wind farm control problems, we also designed a reward regularization module, which brought the essential adapting ability to the algorithm with respect to time-varying wind speeds. A dynamic wind farm simulator, WFSim, was employed for case studies. Simulation results showed that our algorithm achieved superior learning performance compared with the original and PER-based DDPG. It can increase the farm-level power generation under different wind-farm specifications.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[3] Y. Jiang and Z.-P. Jiang, *Robust adaptive dynamic programming*. John Wiley & Sons, 2017.

[4] B. Luo, Y. Yang, and D. Liu, "Policy iteration Q-learning for data-based two-player zero-sum game of linear discrete-time systems," *IEEE Transactions on Cybernetics*, 2020, Early Access.

[5] ——, "Adaptive Q-learning for data-based optimal output regulation with experience replay," *IEEE transactions on Cybernetics*, vol. 48, no. 12, pp. 3337–3348, 2018.

[6] Q. Wei, L. Wang, Y. Liu, and M. M. Polycarpou, "Optimal elevator group control via deep asynchronous actor-critic learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020, Early Access.

[7] D. Wang, C. Mu, D. Liu, and H. Ma, "On mixed data and event driven design for adaptive-critic-based nonlinear H_∞ control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 993–1005, 2017.

[8] C. Mu, Z. Ni, C. Sun, and H. He, "Air-breathing hypersonic vehicle tracking control based on adaptive dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 584–598, 2016.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations, San Juan, Puerto Rico, May 2-4*, 2016.

[11] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *International Conference on Machine Learning*, 2016.

[12] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*, 2016.

[13] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *International Conference on Machine Learning*, 2018.

[14] X. Liang, X. Du, G. Wang, and Z. Han, "A deep q learning network for traffic lights' cycle control in vehicular networks," *IEEE Transactions on Vehicular Technology*, 2019, Early Access.

[15] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel ddpg method with prioritized experience replay," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2017, pp. 316–321.

[16] Y. Ye, D. Qiu, M. Sun, D. Papadaskalopoulos, and G. Strbac, "Deep reinforcement learning for strategic bidding in electricity markets," *IEEE Transactions on Smart Grid*, 2019.

[17] L. Vermeer, J. N. Sørensen, and A. Crespo, "Wind turbine wake aerodynamics," *Progress in Aerospace Sciences*, vol. 39, no. 6-7, pp. 467–510, 2003.

[18] M. Adaramola and P.-Å. Krogstad, "Experimental investigation of wake effects on wind turbine performance," *Renewable Energy*, vol. 36, no. 8, pp. 2078–2086, 2011.

[19] N. Marathe, A. Swift, B. Hirth, R. Walker, and J. Schroeder, "Characterizing power performance and wake of a wind turbine under yaw and blade pitch," *Wind Energy*, vol. 19, no. 5, pp. 963–978, 2016.

[20] R. J. Barthelmie, K. Hansen, S. T. Frandsen, O. Rathmann, J. Schepers, W. Schlez, J. Phillips, K. Rados, A. Zervos, E. Politis *et al.*, "Modelling and measuring flow and wind turbine wakes in large wind farms offshore," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 12, no. 5, pp. 431–444, 2009.

[21] D. Song, J. Yang, X. Fan, Y. Liu, A. Liu, G. Chen, and Y. H. Joo, "Maximum power extraction for wind turbines through a novel yaw control solution using predicted wind directions," *Energy Conversion and Management*, vol. 157, pp. 587–599, 2018.

[22] M. Vali, V. Petrović, S. Boersma, J.-W. van Wingerden, L. Y. Pao, and M. Kühn, "Adjoint-based model predictive control for optimal energy extraction in waked wind farms," *Control Engineering Practice*, vol. 84, pp. 48–62, 2019.

[23] A. C. Kheirabadi and R. Nagamune, "Modeling and power optimization of floating offshore wind farms with yaw and induction-based turbine repositioning," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 5458–5463.

[24] J. Quick, J. Annoni, R. King, K. Dykes, P. Fleming, and A. Ning, "Optimization under uncertainty for wake steering strategies," in *Journal of physics: Conference series*, vol. 854, no. 1, 2017, Paper ID: 012036.

[25] P. A. Fleming, A. Ning, P. M. Gebraad, and K. Dykes, "Wind plant system engineering through optimization of layout and yaw control," *Wind Energy*, vol. 19, no. 2, pp. 329–344, 2016.

[26] M. Vali, V. Petrović, S. Boersma, J.-W. van Wingerden, and M. Kühn, "Adjoint-based model predictive control of wind farms: Beyond the quasi steady-state power maximization," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4510–4515, 2017.

[27] F. Ebrahimi, A. Khayatiyan, and E. Farjah, "A novel optimizing power control strategy for centralized wind farm control system," *Renewable Energy*, vol. 86, pp. 399–408, 2016.

[28] P. M. O. Gebraad, F. Teeuwisse, J. Van Wingerden, P. A. Fleming, S. Ruben, J. Marden, and L. Pao, "Wind plant power optimization through yaw control using a parametric model for wake effects - a CFD simulation study," *Wind Energy*, vol. 19, no. 1, pp. 95–114, 2016.

[29] J. Park and K. H. Law, "Bayesian ascent: A data-driven optimization scheme for real-time control with application to wind farm power maximization," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1655–1668, 2016.

[30] ——, "A data-driven, cooperative wind farm control to maximize the total power production," *Applied Energy*, vol. 165, pp. 151–165, 2016.

[31] S. Boersma, B. Doekemeijer, M. Vali, J. Meyers, and J.-W. van Wingerden, "A control-oriented dynamic wind farm model: WFSim," *Wind Energy Science*, vol. 3, no. 1, pp. 75–95, 2018.

[32] Z. Zhang, J. Chen, Z. Chen, and W. Li, "Asynchronous episodic deep deterministic policy gradient: Toward continuous control in computationally complex environments," *IEEE Transactions on Cybernetics*, 2019, Early Access.

[33] A. M. Do, A. V. Rupert, and G. Wolford, "Evaluations of pleasurable experiences: The peak-end rule," *Psychonomic Bulletin & Review*, vol. 15, no. 1, pp. 96–98, 2008.

[34] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems*, 2014, pp. 3014–3022.

[35] J. R. Marden, S. D. Ruben, and L. Y. Pao, "A model-free approach to wind farm control using game theoretic methods," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1207–1214, 2013.

[36] D. Medici, "Experimental studies of wind turbine wakes: power optimisation and meandering," Ph.D. dissertation, KTH, 2005.

[37] https://github.com/TUDelft DataDrivenControl/WFSim.

[38] S. Boersma, M. Vali, M. Kühn, and J.-W. van Wingerden, "Quasi linear parameter varying modeling for wind farm control using the 2D Navier-Stokes equations," in *American Control Conference*, 2016, IEEE, pp. 4409–4414.

[39] M. Vali, J.-W. van Wingerden, S. Boersma, V. Petrović, and M. Kühn, "A predictive control framework for optimal energy extraction of wind farms," in *Journal of Physics: Conference Series*, Paper ID: 052013, 2016, IOP Publishing.

[40] J. M. Jonkman, M. L. Buhl Jr *et al.*, "FAST user's guide," National Renewable Energy Lab (NREL), Golden, CO, US, Tech. Rep., 2005.

[41] B. M. Doekemeijer, S. Boersma, L. Y. Pao, T. Knudsen, and J.-W. v. Wingerden, "Online model calibration for a simplified les model in pursuit of real-time closed-loop wind farm control," *Wind Energy Science*, vol. 3, no. 2, pp. 749–765, 2018.

[42] N. O. Jensen, "A note on wind generator interaction," 1983.

[43] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a GPU," in *International Conference on Learning Representations*, 2017.

Hongyang Dong is currently a Research Fellow in Machine Learning and Intelligent Control at the School of Engineering, University of Warwick, Coventry, UK. He obtained his Ph.D. degree in Control Science and Engineering from Harbin Institute of Technology, Harbin, China, in 2018. His current research interests include reinforcement learning, deep learning, intelligent control and adaptive control, with their applications in complex systems.

Xiaowei Zhao is Professor of Control Engineering and an EPSRC Fellow at the School of Engineering, University of Warwick, Coventry, U.K. He obtained the PhD degree in Control Theory from Imperial College London, London, U.K., in 2010. After that he worked as a postdoctoral researcher at the University of Oxford, Oxford, U.K., for three years before joining Warwick in 2013. His main research areas are control theory and machine learning with applications in offshore renewable energy systems, local smart energy systems, and autonomous systems.