Deep reinforcement genetic learning-based model reference adaptive inverse control applied on the floating wind turbine

First Author : Hadi Mohammadian KhalafAnsar
Affiliation : Faculty of Mechanical Engineering, University of Tabriz, Iran
Second Author : Jafar Keighobadi
Affiliation : Faculty of Mechanical Engineering, University of Tabriz, Iran
Third Author : Mir Mohammad Ettefagh
Affiliation : Faculty of Mechanical Engineering, University of Tabriz, Iran

Abstract
The ulterior motive of studying floating wind turbine is to decrease the harmful effects of global warming issue. Owing to the crucial changes of the Earth planet, the high potential of active structural control and the technical requirements associated with adaptation of floating wind turbine is applicable. An artificial intelligent controller involving deep reinforcement learning method of both state and action pairs is proposed as useful tool to cope with the variant environmental situations. The floating wind turbine tracks an input reference signal driving by a controller that approximates the inverse of plant model. The proposed adaptive algorithm should minimize the tracking error of the plant output with respect to the reference model output and the controller parameters are updated as well. The model reference adaptive control system is affected by sensor noise and exogenous disturbances. Modeling uncertainties and exogenous inputs are imposed in the overall control system through the proposed intelligent controller. The gathered rewards of both the identification and feed forward terms with control actions in loss function undergoes a Genetic Algorithm (GA) optimization process. Through software implementation results, removing the disturbance and noise effect on the tracking performance of the wind turbine and its stability is vivid.

Keywords: deep reinforcement learning, artificial intelligence, inverse control, model reference adaptive control, Floating Wind Turbine, genetic algorithm.

1. Introduction

This paper explores adaptive signal processing techniques to minimize plant disturbances and address challenges in dynamic system control. The approach involves optimal least-square methods for plant dynamics and disturbance control separately. Inverse control, implemented for both minimum and non-minimum phase systems, uses feed-forward compensation by driving the plant with a filter whose transfer function is the plant's inverse. To leverage intelligent control, the plant model is first identified using conventional neural networks, followed by adaptive inverse control that incorporates model reference control through an additional loss function term.

Benchmark problems are crucial for evaluating control systems, with wind turbines being a notable example due to their complexity, nonlinearity, and under-actuated nature. Research in this field aims to develop controllers that stabilize wind turbines while ensuring robustness against external disturbances [1]. Adaptive control is beneficial for noisy, unknown, or time-variant plants, as it adjusts parameters dynamically to meet varying requirements [2].

The study designs a floating wind turbine with 16 degrees of freedom and three control actions while considering wave and wind disturbances. Deep reinforcement learning (RL) is applied for control optimization. The concept of minimizing disturbance power through optimal adaptive disturbance cancellation, initially demonstrated by Widrow and Walach [1], is further extended. Model Reference Adaptive Control (MRAC) using state variable filters [3] and adaptive filtering for nonlinear systems [4] have been explored in prior work, influencing this study's approach. Adaptive filtering finds applications in antenna systems, channel equalization, spectral estimation, and speech processing.

The paper highlights how deep RL-based adaptive filtering algorithms can control unknown and time-varying systems. Traditional adaptive control involves state feedback with variable parameter networks, while signal processing methods adapt transversal filter weights using gradient methods. The study first applies adaptive filtering for direct modeling of a floating wind turbine, then repurposes the same neural network for inverse control. The plant is assumed to be controllable and observable, with an unknown input-output transfer function.

Machine learning (ML) offers an alternative approach, utilizing data-driven methods to construct models for complex systems [5]. Simulation-generated synthetic data enables diverse applications of neural networks [6], including imitation learning for motor skills [7]. However, RL is preferred for optimizing control, as it dynamically generates samples and learns from sparse reward signals [8]. Effective RL policies depend on well-defined reward functions [9] and require minimal prior knowledge [10-15]. RL methods, integrating optimal and adaptive control properties [16,17], are computationally efficient and suited for nonlinear systems.

This paper emphasizes the capability of ML and RL for mechanical engineering applications. RL-based controllers treat the mechanical system as a black-box environment, collecting input-output data without relying on explicit physical models. The key challenge is defining an appropriate reward function tailored to engineering problems. Post-training techniques further enhance control policy robustness, ensuring satisfactory performance despite environmental uncertainties. Traditional robust control struggles with physical parameter variations, whereas RL-based controllers leverage neural networks' generalization ability to maintain robust performance.

The study employs numerical experiments to evaluate an RL-based nonlinear controller for wind turbines. The contributions include robustness analysis of RL-based control, efficiency improvement in real-world applications through a parametric training policy, and testing under modified environments. The disturbance model uses a more precise wave disturbance formulation, making this study the first comprehensive application of deep RL for wind turbine control, including disturbances. A novel training method extends the agent's capability by re-training neural networks in a modified environment with disturbances.

For numerical validation, a single-frequency sinusoidal disturbance model based on linear wave theory is used. Proper computational tools are essential for capturing disturbance effects accurately, often requiring complex numerical solutions for differential equations. Engineering applications favor simplified models like stationary step disturbances for practical feasibility. This study integrates adaptive filtering with deep RL, training inverse and feed-forward neural network parameters using genetic algorithms. The optimal solution from each generation updates the system, leading to improved data efficiency and robustness in real-world implementations.

2. Research Methodology

Due to the lack of information source in the field of combination deep RL and GA, as a brief explanation, the procedure of applying these two methods should be clarified to the reader. Therefore, this section is divided to individual parts each of which is for special purpose.

2.1. Structure and Training of Deep Reinforcement Learning

RL is a data-driven approach to solving decision-making problems in the form of Markov Decision Processes. Figure 1 shows the workflow where an agent is fed with an observation s and a reward r so that it can decide an action a to take on an environment, with the scope of optimizing the cumulative reward $\sum r$ over time [18,19].

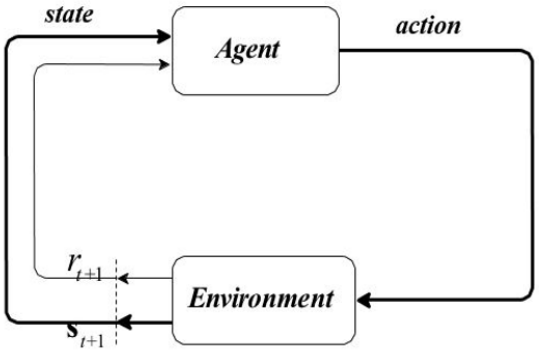


Figure 1: Workflow scheme of the RL approach.

In this work, the algorithm used for training the agent is the Deep Deterministic Policy Gradient (DDPG) [20], an off-policy, model-free, actor-critic structured approach. The agent is composed of four neural networks, namely the Q-function Q_ϑ,the policy $_\varphi$, and their respective target lagged copies $\boldsymbol{Q}'_{\vartheta'}$, and $\boldsymbol{\pi}'_{\varphi'}$. The Q_ϑ network approximates the Q-function of the environment, that is the function predicting the cumulative reward for a given s and a. The policy $_\varphi$ maps the state with an action. Each of the target copies contains lagged parameters of its corresponding neural network. The DDPG algorithm works through the joint learning of the optimal Q-function and the policy. The RL loop begins with the random initialization of the four neural networks. Then, the agent begins to act in the environment to save experiences in an experience buffer of dimension D. Each entrance of the experience buffer is a tuple with the configuration $(\boldsymbol{s},\ \boldsymbol{a},\ \boldsymbol{r},\ \boldsymbol{s}')$ where $\boldsymbol{s}'$ characterizes the new state of the system after taking action a. Then, so as to estimate the optimal Q-function, the Mean Squared Bellman Error (MSBE) is minimized by taking on the loss function specified by [21]:

$$L_{MSBE} = \sum_{i=1}^{d} \left(\mathbf{Q}_\vartheta(\mathbf{s}_i, \mathbf{a}_i) - y_i(\boldsymbol{s}_i,\ \boldsymbol{a}_i,\ \boldsymbol{r}_i,\ \mathbf{s}'_i)\right) \quad (1)$$

where $\mathbf{Q}_\vartheta(\mathbf{s}_i, \mathbf{a}_i)$ is the quantity of the neural network with the current approximation of the optimal Q-function, d is the size of the replay buffer of former experiences haphazardly sampled from the experience buffer of length D enclosing a subset of all the transitions in $(\boldsymbol{s},\ \boldsymbol{a},\ \boldsymbol{r},\ \boldsymbol{s}')$ tuples, $\boldsymbol{s}'$ is the observation of the subsequent state, and y_i is the target value function defined as:

$$y_i = \begin{cases} r_i + \gamma \mathbf{Q}'_{\vartheta'}\left(\mathbf{s}'_i, \pi'_{\phi'}(\mathbf{s}'_i)\right), t < t_{end} \\ r_i, \qquad t = t_{end} \end{cases} \quad (2)$$

where t is the time of the observation $\boldsymbol{s}'$, t_{end} is the total duration of the incidence of training, and γ is the discount factor. In Eq. (2), the target lagged copies $\pi'_{\phi'}(\mathbf{s}'_i)$ and $\mathbf{Q}'_{\vartheta'}\left(\mathbf{s}'_i, \pi'_{\phi'}(\mathbf{s}'_i)\right)$ are utilized to avoid the instability in the training procedure derived from the recursive approach to approximate the Q-function. Then, the neural network $\mathbf{Q}_\vartheta$ is updated through the use of GA to optimize MSBE.

Eventually, the parameters ϑ' and φ' of the target neural networks $\mathbf{Q}'_{\vartheta'}$ and $\pi'_{\phi'}$, are updated by Polyak averaging with the parameter τ as:

$$\vartheta' = \tau\vartheta + (1-\vartheta)\vartheta', \qquad \varphi' = \tau\varphi + (1-\varphi)\varphi' \quad (3)$$

This procedure is reiterated until the optimal policy is found.

2.2. Structure and Basis of Genetic Algorithm

In implementation of a neural network algorithm, we try to recreate the working procedure of neurons in the human brain. GAs are another class of algorithms which use the concept of Darwin's theory of evolution. Based on this theory the existence of all living things is relevant to the rule of "survival of the fittest". Darwin also postulated that three main processes in which new breeds of living things come into existence involve reproduction, crossover, and mutation among existing organisms. As solutions to problems in a more natural way, these concepts in the theory of evolution have been translated into algorithms to search . First, variant possible solutions to a problem are guessed. These solutions are then tested for their performance i.e., how good a solution they provide. Among all possible solutions, a rate of the good solutions is selected, and the others are removed i.e., survival of the fittest. The chosen solutions undergo the processes of reproduction, crossover, and mutation to make a new generation of possible solutions which are expected to do better than the previous generation. The process of producing a new generation and its evaluation is reiterated until there is convergence within a generation [22]. The advantage of this technique is to search for a solution from a wide spectrum of possible solutions, rather than limit the search to a narrow range where the results would be normally expected. The concept above mentioned is shown in Figure 2.

In a GA, the parameter set of the problem is coded as a finite string of bits. For example, given a set of data for amounts of weights, we want to fit a curve through the data. To get a better fit, we encode the parameter set for MSBE in Eq. (1) by creating independent bit strings for the unknown parameters and then concatenate the strings. The bit

strings are combinations of zeros and ones, which represent the value of a number in binary form. An n-bit string can accommodate all integers up to the value $2^n - 1$. As an example, the number 7 requires a 3-bit string, that is, $2^3 - 1 = 7$, and the bit string would look like " 111 ," where the first unit digit is in the 2^2 place $(= 4)$, the second unit digit is in the 2^1 place $(= 2)$, and the last unit digit is in the 2^0 place $(= 1)$; hence, $4 + 2 + 1 = 7$. The number 10 would look like "1010," that is, $2^3 + 2^1 = 10$, from a 4-bit string. This bit string may be mapped to the value of a parameter, say $C_i, i = 1, \dots, n$, by the mapping [23]

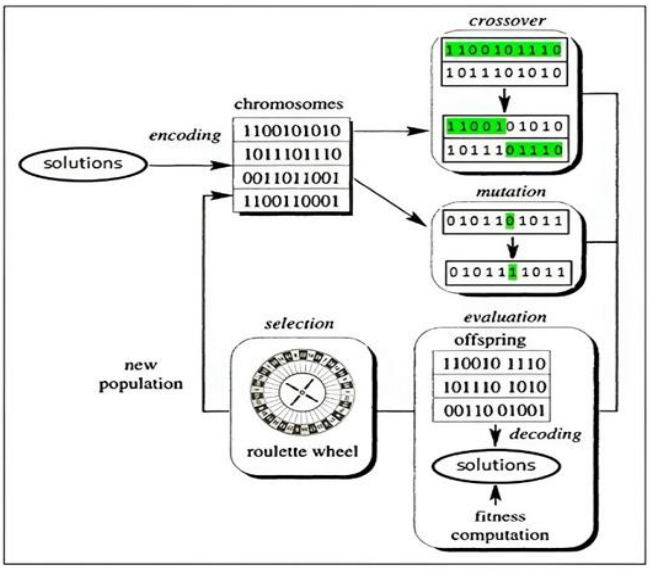


Figure 2: All Processes needed to solve optimal solution with GA

$$W_i = W_{\min} + \frac{b}{2^L - 1}\left(W_{\max_i} - W_{\min_i}\right) \quad (4)$$

where b is the number in decimal form that is being represented in binary form, for example, 152 may be represented in binary form as 10011000, L stands for the length of the bit string L is the length of the bit string, in our case is equal to 6, $W_{\max}$ and $W_{\min}$ are user-defined constants between which amounts of weights vary linearly. The length of the bit strings is based on the handling capacity of the computer being used, that is, on how long a string, strings of each parameter are concatenated to make one long string representing the whole parameter set, the computer can manipulate at an optimum speed.

All genetic algorithms contain three basic operators: reproduction, crossover, and mutation, where all three are analogous to their namesakes in genetics. First, an initial population of n strings of length L is created. The strings are created in a random fashion. Each of the strings is decoded into a set of parameters that it represents. This set of parameters is passed through a numerical model of the problem space. The numerical model gives out a solution based on the input set of parameters. On the basis of the quality of this solution, the string is assigned a fitness value. The fitness values are determined for each string in the entire population of strings. With these fitness values, the three genetic operators are employed to create a new generation of strings, which is expected to perform better than the previous generations. The new set of strings is again decoded and evaluated, and a new generation is created using the three basic genetic operators. This process is continued until convergence is achieved within a population. Among the three genetic operators, reproduction is the process by which we try to ensure that better solutions persist and contribute to better offspring during successive generations. This is a way of ensuring the "survival of the fittest" strings. Because the total number of strings in each generation is kept a constant, strings with lower fitness values are eliminated.

The second operator, crossover, is the process in which the strings are able to mix and match their desirable qualities in a random fashion. After reproduction, crossover proceeds in three simple steps. First, two new strings are selected at random shown in Figure 3a. Second, a random location in both strings is selected illustrated in Figure 3b. Third, the portions of the strings to the right of the randomly selected location in the two strings are exchanged demonstrated in Figure 3c. In this way information is exchanged between strings, and portions of high-quality solutions are exchanged and combined. Reproduction and crossover together give genetic algorithms most of their searching power. The third genetic operator, mutation, helps to increase the searching power. In order to understand the need for mutation, let us consider the case where reproduction or crossover may not be able to find an optimum solution to a problem. During the creation of a generation it is possible that the entire population of strings is missing a vital bit of information, e.g., none of the strings has a one at the fourth location that is important for determining the correct or the most nearly optimum solution. Future generations that would be created using reproduction and crossover would not be able to alleviate this problem. Here mutation becomes important. Occasionally, the value at a certain string location is changed, that is, if there is a one originally at a location in the bit string, it is changed to a zero, or vice versa. Mutation thus ensures that the vital bit of information is introduced into the generation. Mutation, as it does in

nature, takes place very rarely, on the order of once in a thousand bit string locations. The process of generation of strings and their evaluation is continued until we get a convergence to the solution i.e., final values for weights within a generation.

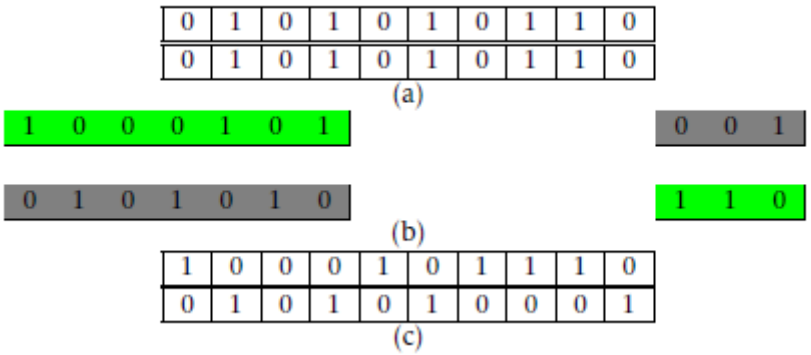


Figure 3: Crossover in strings. (a) Two strings are selected at random to be mated; (b) a random location in the strings is located i.e., here the location is before the last three bit locations; and (c) the string portions following the selected location are exchanged.

2.3. Adaptive Inverse Control Implementation

One of the suitable answers to resist against variant dynamic or conditions of any system is adaptive methodology. It tries to tune parameters of controller with adaptation rules which can be obtained by Lyapunov's stability. Also, adaptive inverse control uses feedforward filter whose transfer function is the same as the one which obtains the inverse of the main plant backward direction [24]. The difference between two input signals, one obtained by inverse transfer function and the other obtained by feedforward filter, is back propagated to regulate the parameters of controller. However, in our case the adaptive filter is substituted with deep reinforcement learning network and instead of backpropagation of differences in errors, genetic algorithm tries to find the optimum and the fittest weight. This procedure is shown in Figure 4. The combination of structures of adaptive inverse and deep RL makes the error MSBE in Eq. (1) be added to difference between inputs. Therefore, the GA efforts are constrained to optimize the following loss function:

$$\begin{aligned} Loss\ \ Function = \ & \tfrac{1}{d}\textstyle\sum_{i=1}^{d} \ \left(\mathbf{Q}_{\vartheta}(\mathbf{s}_i,\mathbf{a}_i) - y_i(\mathbf{s}_i,\mathbf{a}_i,r_i,\mathbf{s}_i')\right)^2 \ |_{id} - \\ & \tfrac{1}{d}\textstyle\sum_{i=1}^{d} \ \left(\mathbf{Q}_{\vartheta}(\mathbf{s}_i,\mathbf{a}_i) - y_i(\mathbf{s}_i,\mathbf{a}_i,r_i,\mathbf{s}_i')\right)^2 |_{ff} \\ & + \tfrac{1}{d}(u-u_e)^2 \end{aligned} \tag{5}$$

Where subscript 'id' stands for identification part in which the first actor and critic of deep RL produces u_e and 'ff' shows the feedforward part in which the second copy of actor and critic of deep RL produces u. Our purpose is to find the minimum of the Eq. (5) that arises the update of networks and controls the floating wind turbine affected by disturbance.

2.4. Dynamic Model Description and Numerical Experiments' Setup

Lately, a plethora of benchmark problems has been used to evaluate the performance of RL algorithms. For instance, the Arcade Learning Environment [25,26] for discrete action space problems or OpenAI Gym [27], which considers benchmark control problems. Among the latter, there is the floating wind turbine [28], a multi-variable, innately unstable. When it comes to RL, the adequate level of complexity of the turbine system results in being beneficial due to its potential generalization to different domains, as opposed to other RL benchmarks such as the locomotion task [29,30], which requires highly specific algorithmic adaptations. In this paper, the mentioned system is chosen as a case study because its topological straightforwardness allows for execution with a detailed study of the sensitivity of the developed controller in the presence of disturbance forces. Figure 5 shows the scheme of the system, consisting of $\vec{F}_A$ the aerodynamic force, $\vec{F}_B$ buoyancy force, $\vec{F}_C$ catenary line forces and $\vec{F}_D$ hydrodynamic drag/inertial force. For each of these forces, there is an associated torque ($\vec{T}_A$, $\vec{T}_B$, $\vec{T}_C$ and $\vec{T}_D$), however these are not shown for lucidity. With the states x, control inputs u and disturbances v and w, we can derive the equations of motion. The nonlinear function f can be written as [31]:

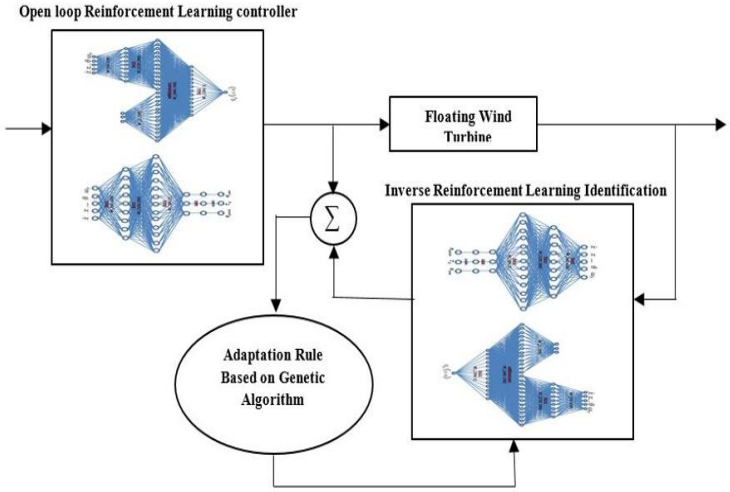


Figure 4: Adaptive inverse control with deep reinforcement learning

$$f(\mathrm{x},\mathbf{u},\mathbf{v},\mathbf{w}) = \begin{bmatrix} \overleftarrow{\dot{x}}_g \\ \overleftarrow{\dot{\theta}}_g \\ \begin{pmatrix}\omega_r \\ \omega_g\end{pmatrix}_{\text{or}} \left(\omega_r - \frac{1}{N_{GR}}\omega_g\right) \\ \overleftarrow{f}_F(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w}) \\ \overleftarrow{f}_T(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w}) \\ f_Q(\mathrm{x},\mathrm{u},\mathrm{v}) \end{bmatrix} \tag{6}$$

The force equation will be employed to obtain the accelerations. That is, in Eq. (6), $\overleftarrow{f}_F(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w})$ will be the sum of all the forces acting on the structure:

$$\overleftarrow{f}_F(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w}) = \left(m_g I_{3\times3} + diag[\overleftarrow{m}_a]\right)^{-1} \sum_j \quad \overleftarrow{F}_j(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w}) \tag{7}$$

where m_g is the total weight of the structure, $I_{3\times3}$ is the identity matrix, $\overleftarrow{m}_a$ is the mass, and $\overleftarrow{F}_j(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w})$ will contain all applied forces.

The torque equation will be to obtain angular accelerations, i.e. in Eq. (6) $\overleftarrow{f}_T(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w})$ will be the sum of all the torques resulting from the applied forces on the structure [31]:

$$\overleftarrow{f}_T(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w}) = \left(\mathrm{R}\underline{I}_g^{-1}\mathrm{R}^T\right)\sum_j \quad \overleftarrow{T}_j(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w}) \tag{8}$$

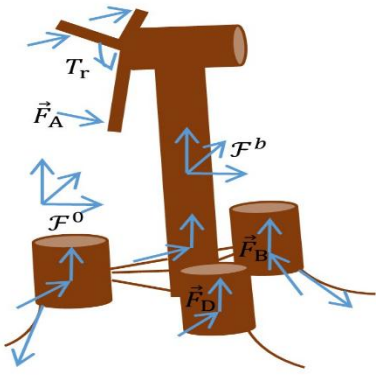


Figure 5: Overall Force Diagram of the Non-Linear Model

where I_g is the inertial tensor around the vertical axis, R is the transformation matrix, and $\overleftarrow{T}_j(\mathrm{x},\mathrm{u},\mathrm{v},\mathrm{w})$ includes all of the torques by the forces acting on the structure.

$f_Q(\mathrm{x},\mathrm{u},\mathrm{v})$ is also found by:

$$f_Q(\mathrm{x},\mathbf{u},\mathrm{v}) = \begin{bmatrix} \sum_{k_r} & \frac{1}{J_r} Q_{k_r}(\mathrm{x},\mathrm{u},\mathrm{v}) \\ \sum_{k_g} & \frac{1}{J_g} c(\mathrm{x},\mathbf{u},\mathrm{v}) \end{bmatrix} \tag{9}$$

where J_r and J_g represent the inertia about the rotor-side shaft and generator-side shaft, respectively, and Q_{k_r} and Q_{k_g} represent the ${k_r}^{th}$ and ${k_g}^{th}$ torque about each respective shaft. The aim of simulation is to determine the displacement components along the "X_t,Y_t,Z_t" directions, as "surge, sway and heave" respectively, and the angle components around body axes "x,y,z", as "roll, yaw, pitch", respectively. To become more familiar with the components of the problem, refer to the Figure 6a.

The buoyancy force acting on a floating object is identical to the weight of the fluid being moved by object based on Archimedes' principle. The operative buoyancy force on the ${i}^{th}$ fluctuating cylinder is:

$$\tilde{\vec{F}}_{B,i}(x) = \rho_\omega g A_i l_i \hat{e}_3 \tag{10}$$

Where $\rho_\omega$is the density of water, $g$the gravity constant, $A_i$the cross section of the cylinder and $l_i$length of cylinder.

The drag force is a squandering force that resists the relative motion between the body and the fluid. For a transversely immersed cylinder, the Morrison equation provides a simple approximation for the surface drag force associated with the flow direction [31]:

$$\tilde{\vec{F}}_{Dt} = K_{d,i}\left\|\tilde{\vec{v}}_{t,i}\right\| + K_{a,i}\tilde{\vec{a}}_{t,i} \tag{11}$$

where $K_{d,i}$ is the drag constant and $K_{a,i}$ the inertia constant of the Morrison equation, and $\vec{v}_{t,i}$ and $\vec{a}_{t,i}$ are the transverse velocities and accelerations.

The thrust force is the wind force in a direction parallel to the axis of rotor rotation, while the drag force is the wind force in the direction of motion of a point on the blade. Indeed, these forces are constantly acting on the entire blade. However, for simplicity, the net thrust force for all three blades is placed in where is called the thrust center.

The net thrust force is estimated as:

$$\tilde{\vec{F}}_A = \frac{1}{2}\rho A_r C_t(\lambda,\beta)\|\tilde{\vec{v}}_n\|\tilde{\vec{v}}_n \tag{12}$$

where C_t is the drift coefficient, a function of the tip speed ratio (TSR) and the blade pitch angle. The ρ and A_r show the air density and the swept area of the rotor, respectively.

With N_{gr} as gear ratio, we reckon its torque, the same transformation matrices will reappear in the equation and the generator torque term will be added:

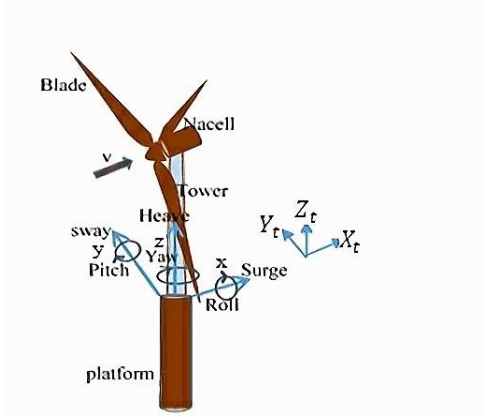
$$T = R \times \tilde{\vec{F}}_A(x,\mathrm{u},v) + T_g R\frac{J_r+N_{gr}J_g}{J_r+N_{gr}^2 J_g}\hat{e}_1 \tag{13}$$

If the aerodynamic power is written:

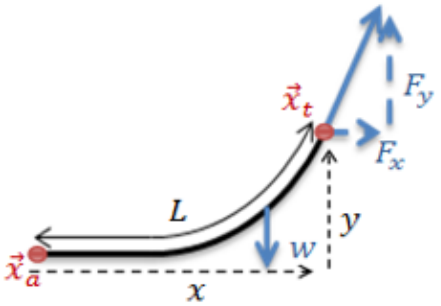
$$P = \frac{1}{2}\rho A_r C_p(\lambda,\beta)\|\tilde{\vec{v}}_n\|^3 \tag{14}$$

The torque balance about the rotor axis and the generator axis leads to [31]:

$$\begin{aligned}\tilde{\dot{\omega}}_r &= \frac{1}{J_r}\left(\frac{P}{\omega_r} - k\left(\theta_r - \frac{1}{N_{gr}}\theta_g\right) - b\left(\omega_r - \frac{1}{N_{gr}}\omega_g\right)\right)\\ \tilde{\dot{\omega}}_g &= \frac{1}{J_g}\left(-T_g + \frac{k}{N_{gr}} + \frac{b}{N_{gr}}\left(\omega_r - \frac{1}{N_{gr}}\omega_g\right)\right)\end{aligned} \tag{15}$$



a)



b)

Figure 6: a) Angular and displacement components of system. b) Drag force of cables.

The mooring system consists of a series of cables connecting the wind turbine into the seabed. The force cables provide a recovery force in response to structural shifts caused by wind, wave's disturbances. The Gaussian static model of a cable is described in two dimensions and entails two nonlinear coupling equations relating the horizontal and vertical distances between the ends of the cable to the two-dimensional force at the wind turbine connection point. It is important to note that the equations change depending on whether a part of the rope is on the seabed or is in direct contact with seabed. Figure 6b illustrates cable's force.

The vector $\vec{x}_{t,i}$ is the point of connection to the structure given by:

$$\tilde{\vec{x}}_{t,i} = \tilde{\vec{x}}_{a,i} - \tilde{\vec{x}}_g - \underline{R}\tilde{\vec{r}}_{gci}^{\,b} \tag{16}$$

The vector of $\vec{x}_{t,i}$must be decomposed into its components along the axes. It is operated as:

$$\tilde{x}_{th,i} = \frac{\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}\left(\tilde{x}_{a,i}-\tilde{x}_g-\underline{R}\tilde{r}^b_{gci}\right)}{\|\tilde{x}_{t,i}\|}$$
$$y_t = \begin{pmatrix}0\\0\\1\end{pmatrix}\left(\tilde{x}_{a,i}-\tilde{x}_g-\underline{R}\tilde{r}^b_{gci}\right) \quad (17)$$

the forces in the x and y directions are obtained:

$$F_x = \frac{W_c}{\left(1\ \|\tilde{x}_{t,i}\|\ \|\tilde{x}_{t,i}\|^2\ \|\tilde{x}_{t,i}\|^3\ \|\tilde{x}_{t,i}\|^4\ \|\tilde{x}_{t,i}\|^5\right)P_c\begin{bmatrix}1\\y_t\end{bmatrix}}\tilde{x}_{th,i}$$
$$F_y = W_c\sqrt{\left(\frac{2}{\left(1\ \|\tilde{x}_{t,i}\|\ \|\tilde{x}_{t,i}\|^2\ \|\tilde{x}_{t,i}\|^3\ \|\tilde{x}_{t,i}\|^4\ \|\tilde{x}_{t,i}\|^5\right)P_c\begin{bmatrix}1\\y_t\end{bmatrix}}+y_t\right)y_t}\begin{bmatrix}0\\0\\-1\end{bmatrix} \quad (18)$$

3. Implementation of New Approach

The instant reward function for the environment is defined as follows:

$$r = \begin{cases} r_1, & y \le y_{lim} \\ r_2, & y > y_{lim} \end{cases} \quad (19)$$

where y stands for output of the plant, y_{lim} is the maximum desirable amounts allowed for the plant during the oscillation task, while r_1 and r_2 are defined as:

$$r_1 = (A_r|x|^n + B_r|\theta|^n + C_r|u|^n)D_r \quad (20)$$

and:

$$r_2 = r_1 + E_r \quad (21)$$

where the constant parameter s are respectively set as $A_r = 10^{-2}$, $B_r = 10^{-1}$, $C_r = 5$, $D_r = -10^{-2}$, $E_r = -10^{2}$, and n = 2. The shape of the reward function allows for penalizing the magnitude of x, , and u. Therefore, the optimal policy will be that in which the system manages to perform the oscillation with the lowest input possible, and the tower is in the initial position at the end of the task. On the other hand, the actor π_φ and critic Q_ϑ neural networks are multilayer perceptrons with architectures as respectively shown in (a) and (b) of Figure 7.

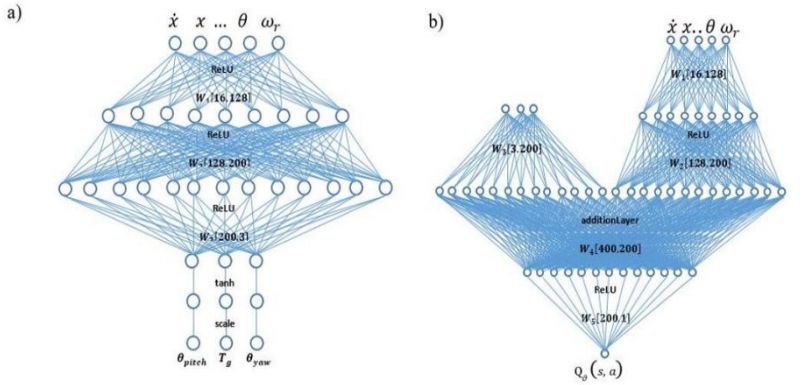


Figure 7: (a) Actor neural network π_φ architecture. (b) Critic neural network Q_ϑ architecture.

The training process of the agent is done by employing the DDPG algorithm under the following restrictions:

$$u \le u_{max}, \quad |x| \le x_{lim,} \quad t \le t_f \quad (22)$$

where u_{max} is the maximum amount of input allowed to the controller, set to [18 rad, 4×10^4 N.m, 20 rad] for roll angle, generator torque and pitch angle, respectively and $x_{lim,}$ is the maximum lateral output of the plan, and t_f is the maximum duration of the task, set to 25 sec. Then, the sensitivity analysis of the controller consists of evaluating its performance under modified environments featuring variations of the physical properties of the system. Finally, to expand the optimal behavior space for the agent, a post-training of its structural parameters is performed in a modified environment.

3.1. Stability analysis

Consider the plant model (6) and Equation (20) for capturing reward, estimation errors of both the state vector and the weighting parameters model have been guaranteed.

Proof of Theorem 2. Consider the following Lyapunov candidate

$$V = \frac{1}{2}r^2 + \frac{1}{2}(A_r^2 + \beta_r^2 + C_r^2) \quad (23)$$

Taking the time derivative of V leads to:

$$\dot{V} = r\dot{r} + \left(A_r\dot{A}_r + B_r\dot{B}_r + C_r\dot{C}_r\right) \quad (24)$$

Replacing Eq. (20), results in:

$$\dot{V} = A_r\left(\dot{r}|x|^2 + \dot{A}_r\right) + B_r\left(\dot{r}|\theta|^2 + \dot{\beta}_r\right) + C_r(\dot{r}|u|^2)^2 + \dot{C}_r\big) \quad (25)$$

Assuming Eq. (2), yields to:

$$\dot{V} = A_r\left(\dot{y}|x|^2 - \gamma\dot{Q}'|x|^2 + \dot{A}_r\right) + B_r\left(\dot{y}|\theta|^2 - \gamma\dot{Q}'|\theta|^2 + \dot{B}_r\right) + C_r\left(\dot{y}|u|^2 - \gamma\dot{Q}'|u|^2 + \dot{C}_r\right)^2 \quad (26)$$

Factorizing and rewriting terms multiplied by $\dot{y}$ lead to:

$$\dot{V} = \dot{y}(\underbrace{A_r|x|^2 + \beta_r|\theta|^2 + C_r|u|^2}_{r}) + A_r\left(-\gamma\dot{Q}'|x|^2 + A_r\right) + B_r\left(-\gamma\dot{Q}'|\theta|^2 + \dot{B}_r\right) + C_r\left(-\gamma\dot{Q}'|u|^2 + \dot{C}_r\right) \quad (27)$$

If we choose quantity of second, third and fourth parenthesis in a way making them zeros, three adaptive laws for featuring amounts of coefficients in reward equation will be achieved as following:

$$\dot{A}_r = \gamma\dot{Q}'|x|^2 \dot{B}_r = \gamma\dot{Q}'|\theta|^2 \dot{C}_r = \gamma\dot{Q}'|u|^2 \quad (28)$$

Only left term of Lyapunov's function derivative is:

$$\dot{V} = \dot{y}r \leq |\dot{y}||\mathrm{r}| \leq \left(|\dot{r}| - \gamma\left|\dot{Q}'\right|\right)|\mathrm{r}| \quad (29)$$

Outside of the impact set $S = \{s||\dot{r}| \geq \gamma\dot{Q}'\}$ derivative of Lyapunov function is negative and proves system's error will be ultimately bounded. Furthermore, the fact that the derivative of Lyapunov's function is negative implies that the entire system is stable and the network weights are convergent. Based on Eq. (29) and the range of its trueness, r is bounded, causing y to be bounded in Equation (2), allowing the cost function in Equation (1) to be optimized.

4. Numerical Results and Discussion

The starting point of this paper is the formulation of a multibody model of the floating wind turbine system easily manipulated by controlling an external torque applied to the tower. Simulink diagram of the model and controller is illustrated in Figure 8. This dynamic model is complex to exhibit a nonlinear behavior during the oscillation task and, at the same time, sufficiently simple to be used for a parametric study of the robustness and sensitivity of a nonlinear control law devised by using a deep RL method. Furthermore, the presence of disturbance, often inaccurately neglected in applications leading to the origination of unwanted dynamic phenomena, is taken into account in this investigation. As expected, the presence of disturbance has a mere impact on the performance of the controller devised by employing the deep RL approach. Implementing the modified environment and performing a post-training procedure for the agent resulting from the original design require only a small set of thirty-nine additional episodes to generate the optimal control policy for environments with or without the perturbation presence. For instance, the plant can indeed do the oscillation in an environment with disturbance, as shown in Figures 9 through 11. Thereby, this approach provides a promising path for further developments. As can be seen in Figures 9 through 11, our purpose, that is, controlling and stabilizing floating wind turbine has been fulfilled and after a short oscillating, the system reaches to final position.

4.1. Discussion and Final Remarks

Based on the impact of the modified environments on the performance of the floating wind turbine devised in this work, the robustness of the RL-based control system is concluded. Therefore, the numerical experiments performed in this investigation prove the transfer learning approach to be promising for the reality-gap problem found in literature especially in the robotics field. Nonlinear model-based control systems, such as [43-46], to name a few, do not consider disturbance since the complexity of the resulting model is restrictive. In contrast, a model-free approach like the one in this work, based on deep RL, allows us to consider such a phenomenon. Additionally, given the generalization capability of neural networks, the robustness of the system to the uncertainty in the physical parameters is high, as shown in figures. This, to the best of the authors' knowledge, has not been previously done in the literature. Furthermore, supposed robustness can be increased with the post-training proposed and numerically tested in the present work. The developed control system has restrictions related to the simplifications used in the disturbance model present in the environment during training. Furthermore, the floating wind turbine capacity can be extended by running training epochs in a subsequent session in an experimental setting. Assumed implementation is feasible given the sampling time used during training and the performance required by an actuator, both within commercial hardware margins.

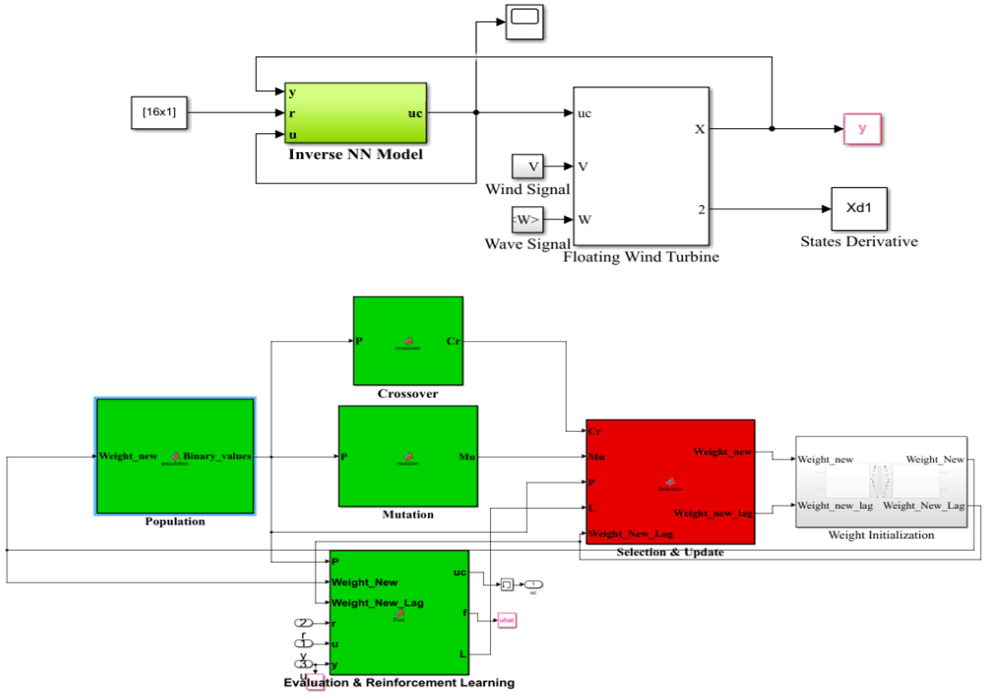


Figure 8: Simulink diagram of floating wind turbine (upper figure), and Adaptive inverse with using deep RL (underneath figure)

4.2. Summary and Conclusions

The authors' research focuses on model designing, and controlling mechanical systems subjected to complex force fields. Therefore, the mutual interactions between multibody system dynamics, applied system identification, and nonlinear control fall within the scope of the authors' research domain. In fact, the benchmark problems mentioned before are closer to practical systems employed in engineering applications, and the development of a robust controller for such systems seems quite promising.

In this paper, also, the sensitivity analysis of the deep RL controller applied to the floating wind turbine problem is performed. Through extensive numerical experiments, the effectiveness of the proposed controller is analyzed in the case of the presence of disturbance forces. The uniform ultimate boundedness of estimation errors of both the state vector and the weighting parameters model have been guaranteed by Lyapunov's direct method. Subsequent works will focus on studying environments characterized by randomized parameters during the training procedure to further improve the robustness of the resulting control system. In summary, the current study focused on the computational elements of the floating wind turbine system's oscillation problem. The goal of this work is to create a nonlinear controller using deep RL approaches that are simulated in a virtual environment. The numerical results of using the controller were then simulated in Matlab, demonstrating key characteristics of floating wind turbines. Future expansions of the current study will include the creation of an in-depth examination of the deep RL-based sensitivity and robustness of the control system, as well as its experimental application employing sophisticated disturbance models.

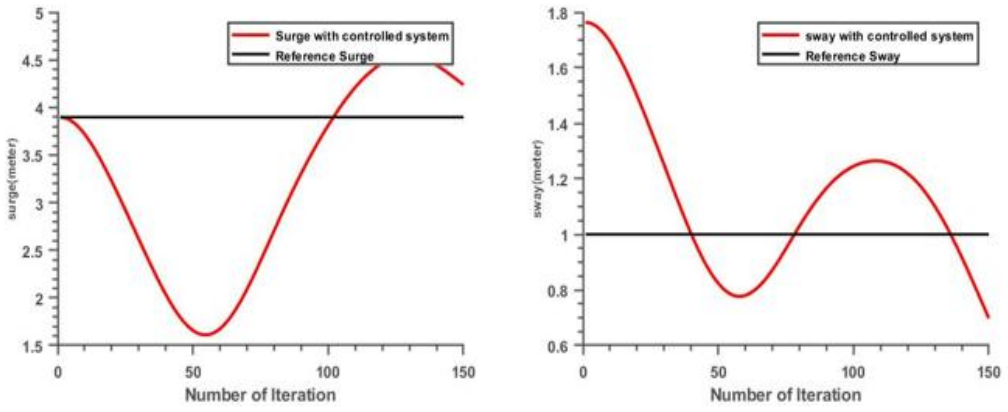


Figure 9: Surge and Sway with deep reinforcement learning-based model reference adaptive inverse control

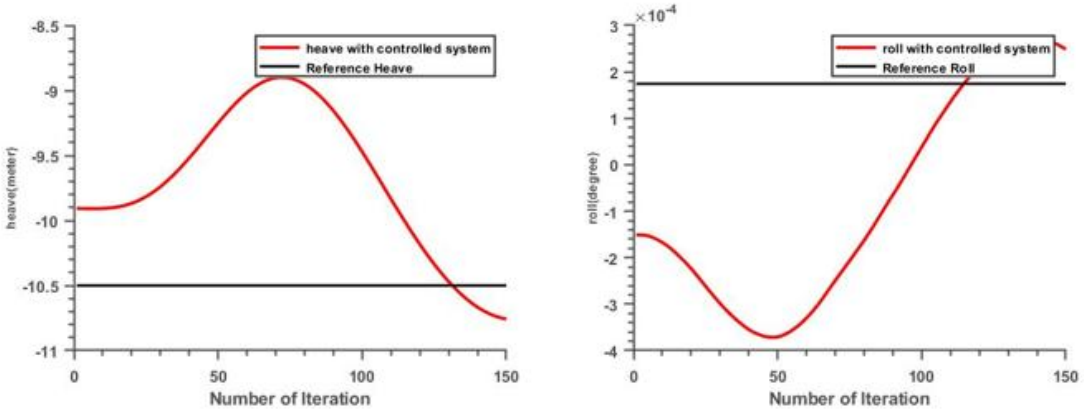


Figure 10: Heave and Roll with Deep reinforcement learning-based model reference adaptive inverse control

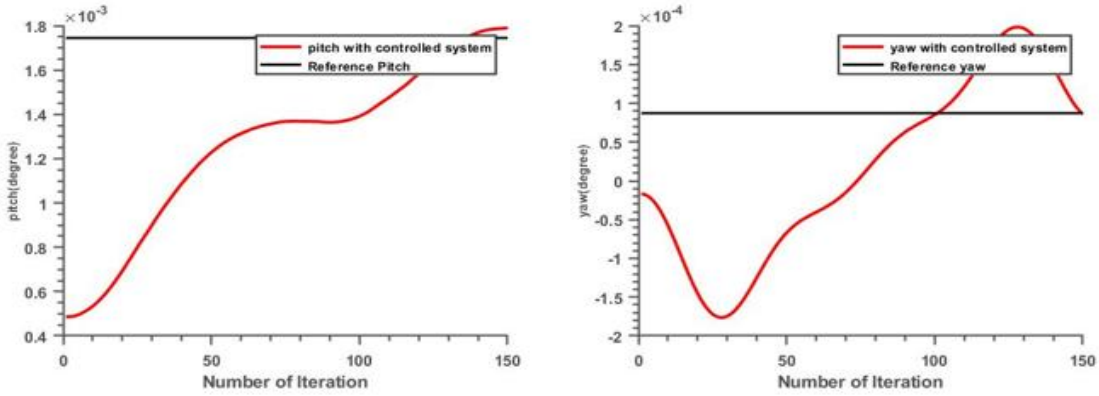


Figure 11: Pitch and Yaw with Deep reinforcement learning-based model reference adaptive inverse control

References

[1] Widrow, B; Duvall, K.M.; Gooch, R.P.; Newman, W.C. Signal cancellation phenomena in adaptive antennas: Cases and cures, IEEE Trans. Antennas Propag., Vol. AP-30, May 1982, pp.469-478.
[2] Jim, C.W. A comparison of two LMS constrained optimal array structures, Proc IEEE, Vol 65, December 1977, pp. 1730-1731.
[3] Griffiths, L.J.; Jim, C.W. An alternative approach to linearly constrained adaptive beamforming, IEEE Trans. Antenna Propag., Vol. AP-30, January 1982, pp. 27-34.
[4] Gooch, R.P. Adaptive pole-zero array processing, in Proc. 16th Asilomar Conf. Circuits Syst. Comput. Santa Clara, CA, November 1982.
[5] Hesse, M.; Timmermann, J.; Hullermeier, E.; Trachtler, A. A Reinforcement Learning Strategy for the Swing-Up of the Double Pendulum on a Cart. Procedia Manuf. 2018, 24, 15.
[6] Manrique, C.; Pappalardo, C.M.; Guida, D. A Model Validating Technique for the Kinematic Study of Two-Wheeled Vehicles. Leet. Notes Mech. Eng. 2019, 549-558._56.
[7] Pappalardo, C.M.; De Simone, M.C.; Guida, D. Multibody modeling and nonlinear control of the pantograph/catenary system. Arch. Appl. Mech. 2019, 89, 1589-1626.
[8] Pappalardo, C.; Guida, D. Forward and Inverse Dynamics of a Unicycle-Like Mobile Robot. Machines 2019, 7, 5.
[9] Villecco, F.; Pellegrino, A. Evaluation of Uncertainties in the Design Process of Complex Mechanical Systems. Entropy 2017, 19, 475.
[10] Villecco, F.; Pellegrino, A. Entropic Measure of Epistemic Uncertainties in Multibody System Models by Axiomatic Design. Entropy 2017, 19, 291.
[11] Hu, D.; Pei, Z.; Tang, Z. Single-Parameter-Tuned Attitude Control for Quadrotor with Unknown Disturbance. Appl. Sci. 2020, 10, 5564.
[12] Talamini, J.; Bartoli, A.; De Lorenzo, A.D.; Medvet, E. On the Impact of the Rules on Autonomous Drive Learning. Appl. Sci. 2020, 10, 2394.
[13] Sharifzadeh, S.; Chiotellis, I.; Triebel, R.; Cremers, D. Learning to drive using inverse reinforcement learning and deep q-networks. ArXiv 2016, arXiv:1612.03653.
[14] Cho, N.J.; Lee, S.H.; Kim, J.B.; Suh, I.H. Learning, Improving, and Generalizing Motor Skills for the Peg-in-Hole Tasks Based on Imitation Learning and Self-Learning. Appl. Sci. 2020, 10, 2719.
[15] Zhang, H.; Qu, C.; Zhang, J.; Li, J. Self-Adaptive Priority Correction for Prioritized Experience Replay. Appl. Sci. 2020, 10, 6925.
[16] Hong, D.; Kim, M.; Park, S. Study on Reinforcement Learning-Based Missile Guidance Law. Appl. Sci. 2020, 10, 6567.

[17] Rivera, Z.B.; De Simone, M.C.; Guida, D. Unmanned Ground Vehicle Modelling in Gazebo/ROS-Based Environments. Machines 2019, 7, 42.
[18] De Simone, M.C.; Guida, D. Control design for an under-actuated UAV model. FM E Trans. 2018, 46, 443-452.
[19] Murray, R.; Palladino, M. A model for system uncertainty in reinforcement learning. Syst. Control Lett. 2018, 122, 24-31.
[20] Sutton, R.S.; Barto, A.G.; Williams, R.J. Reinforcement learning is direct adaptive optimal control. IEEE Control Syst. 1992, 12, 19-22.
[21] Cheng, Q.; Wang, X.; Niu, Y.; Shen, L. Reusing Source Task Knowledge via Transfer Approximator in Reinforcement Transfer Learning. Symmetry 2018, 11, 25.
[22] Nichols, B.D. Continuous Action-Space Reinforcement Learning Methods Applied to the Minimum-Time Swing-Up of the Acrobat. In Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics, Hong Kong, China, 9-12 October 2015; pp. 2084-2089.
[23] Forrest, S. Genetic algorithms: principles of natural selection applied to computation. Science, 1993, 261, 872-878.
[24] Goldberg, D. Genetic algorithms, Addison-Wesley, New York, 1989.
[25] Lesort, T.; Diaz-Rodriguez, N.; Goudou, J.-F.; Filliat, D. State representation learning for control: An overview. Neural Net. 2018, 108, 379-392.
[26] Oh, E. Reinforcement-Learning-Based Virtual Energy Storage System Operation Strategy for Wind Power Forecast Uncertainty Management. Appl. Sci. 2020, 10, 6420.
[27] Phan, B.C.; Lai, Y.-C. Control Strategy of a Hybrid Renewable Energy System Based on Reinforcement Learning Approach for an Isolated Microgrid. Appl. Sci. 2019, 9, 4001.
[28] Kovari, B.; Hegedus, F.; Becsi, T. Design of a Reinforcement Learning-Based Lane Keeping Planning Agent for Automated Vehicles. Appl. Sci. 2020, 10, 7171.
[29] Tran, D.Q.; Bae, S.-H. Proximal Policy Optimization Through a Deep Reinforcement Learning Framework for Multiple Autonomous Vehicles at a Non-Signalized Intersection. Appl. Sci. 2020, 10, 5722.
[30] Sattarov, O.; Muminov, A.; Lee, C.W.; Kang, H.K.; Oh, R.; Ahn, J.; Oh, H.J.; Jeon, H.S. Recommending Cryptocurrency Trading Points with Deep Reinforcement Learning Approach. Appl. Sci. 2020, 10, 1506.
[31] Homer, J.R. Physics-based control-oriented modelling for floating offshore floating wind turbine, M.S. thesis, Univ. British Columbia, Vancouver, BC, Canada, 2015.
[32] Chen, H.; Liu, Y.; Zhou, Z.; Zhang, M. A2C: Attention-Augmented Contrastive Learning for State Representation Extraction. Appl. Sci. 2020, 10, 5902.
[33] Xiang, G.; Su, J. Task-Oriented Deep Reinforcement Learning for Robotic Skill Acquisition and Control. IEEE Trans. Cybern. 2019, 1-14.
[34] Wawrzyftski, P. Control Policy with Autocorrelated Noise in Reinforcement Learning for Robotics. Int. J. Mach. Learn. Comput. 2015, 5, 91-95.
[35] Beltran-Hernandez, C.C.; Petit, D.; Ramirez-Alpizar, LG.; Harada, K. Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep-Reinforcement-Learning Approach. Appl. Sci. 2020, 10, 6923.
[36] Moreira, I.; Rivas, J.; Cruz, F.; Dazeley, R.; Ayala, A.; Fernandes, B. Deep Reinforcement Learning with Interactive Feedback in a Human-Robot Environment. Appl. Sci. 2020, 10, 5574.
[37] Williams, G.; Wagener, N.; Goldfain, B.; Drews, P.; Rehg, J.M.; Boots, B.; Theodorou, E.A. Information theoretic MPC for model-based reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May-3 June 2017; pp. 1714-1721.
[38] Anderson, C.W.; Lee, M.; Elliott, D.L. Faster reinforcement learning after pretraining deep networks to predict state dynamics. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12-16 July 2015; Volume 2015, pp. 1-7.
[39] Lee, M.; Anderson, C.W. Convergent reinforcement learning control with neural networks and continuous action search. In Proceedings of the 2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Orlando, FL, USA, 9-12 December 2014; pp. 1-8.
[40] Maei, H.R.; Szepesvari, C.; Bhatnagar, S.; Precup, D.; Silver, D.; Sutton, R.S. Convergent temporal-difference learning with arbitrary smooth function approximation. In Proceedings of the 22nd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 7-10 December 2009; pp. 1204-1212.
[41] Morimoto, J.; Doya, K. Robust Reinforcement Learning. Neural Comput. 2005, 17, 335-359.
[42] Frarn; ois-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An Introduction to Deep Reinforcement Learning. Found. Trends Mach. Learn. 2018, 11, 219-354.
[43] Yang, Y.; Li, X.; Zhang, L. Task-specific pre-learning to improve the convergence of reinforcement learning based on a deep neural network. In Proceedings of the 2016 12th World Congress on Intelligent Control and Automation (WCICA), Guilin, China, 12-15 June 2016; Volume 2016, pp. 2209-2214.
[44] Zagal, J.C.; Ruiz-del-Solar, J.; Vallejos, P. Back to reality: Crossing the reality gap in evolutionary robotics. IFAC Proc. Vol. 2004, 37, 834-839.
[45] Bekar, C.; Yuksek, B.; Inalhan, G. High Fidelity Progressive Reinforcement Learning for Agile Maneuvering UAVs. In Proceedings of the AIAA Scitech 2020 Forum; American Institute of Aeronautics and Astronautics, Orlando, FL, USA, 6-10 January 2020; pp. 1-12.
[46] Al-Araji, A.S. An adaptive swing-up sliding mode controller design for a real inverted pendulum system based on Culture-Bees algorithm. Eur. J. Control 2019, 45, 45-56.