Reinforcement Learning Based Maximum Power Point Tracking Control of Tidal Turbines

Reinforcement Learning Based Maximum Power Point Tracking Control of Tidal Turbines

Anup Nambiar#1, Enrico Anderlini#2, Grégory S. Payne#3, David Forehand#4, Aristides Kiprakis#5, Robin Wallace#6

#*Institute for Energy Systems, School of Engineering*
The University of Edinburgh, Edinburgh, UK

1A.Nambiar@ed.ac.uk
2E.Anderlini@ed.ac.uk
3Gregory.Payne@ed.ac.uk
4D.Forehand@ed.ac.uk
5Aristides.Kiprakis@ed.ac.uk
6Robin.Wallace@ed.ac.uk

Abstract— **This paper explores the use of model free reinforcement learning (RL) in the maximum power point tracking (MPPT) control of a tidal turbine. Two RL algorithms – Q-learning and Neural Fitted Q-iteration – are used in this work to identify the optimal power curve of the turbine, which is then used to control the turbine in real time. The RL algorithms are setup to maximise the energy yield of the turbine and are tested in different tidal flow conditions. These algorithms are tested through numerical simulations of a tide-to-wire model of a direct drive Permanent Magnet Synchronous Generator based tidal turbine. The algorithms are found to converge to the optimal power curve coefficient for the different tidal current flows tested. Using RL allows the turbine control system to adapt to changes in the turbine characteristics brought about by causes like biofouling and long-term changes to the tidal flow patterns at a deployment site. RL requires no prior knowledge of the tidal turbine system, which is an advantage of the described approach.**

***Keywords*— Tidal turbine, control, maximum power point tracking, reinforcement learning.**

I. INTRODUCTION

The last few years have seen a significant growth in the tidal power industry with arrays of tidal devices being installed and planned. Generating tidal power using variable speed generators, like in the majority of wind turbines, allows for more efficient power extraction. This is because the turbine can operate at its maximum power points, on the power-speed curve, over all current conditions.

Different maximum power point tracking (MPPT) algorithms have been developed for wind turbines [1]. Since tidal turbines, structurally, are similar to wind turbines some of these control algorithms have been extended to tidal devices too [2],[3]. This paper explores the use of reinforcement learning (RL) based maximum power point tracking (MPPT) algorithms for tidal turbines using numerical simulations. RL is an on-line learning algorithm in which an agent learns from its own experience of interacting with its environment.

A similar RL based MPPT approach has been developed and tested for wind turbines in [4]. Turbulence, wakes in arrays, possible wave action and biofouling make the extension of the algorithm to tidal devices more challenging. This paper proposes an approach by which the optimal power curve of a tidal turbine is learnt through RL. The main advantages of this approach are that there is no need for prior knowledge of the system and that the control system can adapt to changes in the turbine characteristics.

Section II in the paper describes the tide-to-wire model of the permanent magnetic synchronous generator (PMSG) based turbine and the high level MPPT controller used in this work. A general background to reinforcement learning and to the two RL algorithms used here are presented in Section III. The application of the two RL algorithms in MPPT control of a single tidal turbine is reported in Section IV. Results from the simulation runs using the two RL variants and for the different tidal current flow cases considered are reported in Section V and are further discussed in Section VI. Section VII summarises the main contributions of this paper in tidal turbine control.

II. TIDAL TURBINE MODEL AND CONTROL

A. Physical model of the turbine

The turbine model used for this project is a three bladed horizontal axis machine. The rotor is 1.2 m in diameter and the blade profile was designed so that it produces a radial variation of thrust similar to that of a full scale generic design described in [5]. The model is highly instrumented with sensors measuring the streamwise bending moment at the root of each blade, the thrust and torque over the whole rotor and the absolute angular position of the rotor. Fig. 1 shows the turbine model being tested at the FloWave basin of the University of Edinburgh. More details on the turbine model and on its design and manufacturing can be found in [6].

B. BEMT based numerical model of the turbine

To simulate the turbine model performance, a standard blade element momentum theory (BEMT) code was developed. This approach relies on estimating flow momentum extraction by the rotor. The method used here is one dimensional (in polar coordinates) in that the flow energy extraction is considered uniform over the annulus elements of the rotor disk. In other words, energy extraction is considered to be affected spatially only by the radius from the rotor axis and not by azimuthal position over the rotor disk. This method yields estimates of the torque and thrust applied to the rotor and of their radial variations. The main inputs to the model are the onset flow velocity, the rotor rotational velocity and the lift and drag coefficients of the different blade elements. The foil sections used for the turbine blades are NACA 63-8XX and the lift and drag coefficients values used for the model were taken from [7]. The BEMT method used here includes Spera high axial induction factor correction and is described in detail in chapter 4 of [8].

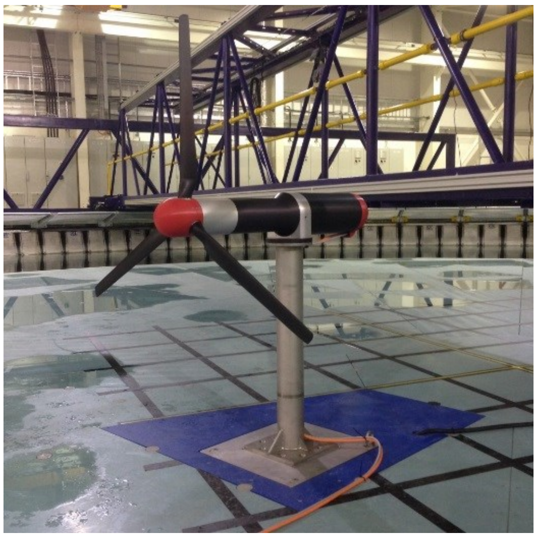

Fig. 1 Turbine model mounted on the floor of the FloWave tank

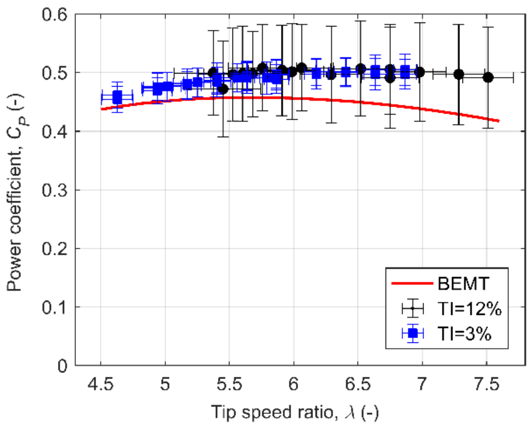

Fig. 2 Numerical and experimental power coefficients plotted against tip speed ratio. The error bars correspond to the standard deviation of the measurements

The total mechanical power extracted by a tidal turbine can be represented by:

$$P_m = \frac{1}{2}\rho A u^3 C_p(\lambda,\beta), \tag{1}$$

where ρ is the sea water density, A is the rotor swept area, u is the onset tidal current velocity and C_p is the power coefficient, which is a function of the tip speed ratio λ and the blade pitch angle β. Fig. 2 shows the power coefficients C_p plotted against tip speed ratio λ for pitch angle $\beta = 0°$. The graph includes coefficients measured experimentally at the IFREMER flow recirculating flume of Boulogne-sur-Mer (France) [9] with an onset flow of 0.815 m/s and turbulence intensity levels of 3 and 12%. It also includes Cp values predicted by the BEMT model. It can be seen that the numerical model underestimates slightly the experimental measurements. This is believed to be due to the fact that the standard BEMT approach implemented here does not account for the relatively high blockage factor (14%) of the experiment.

As in wind energy systems, operating the tidal turbine at the optimal tip speed ratio λ_{opt} maximises the power extraction from the tidal current flow. At λ_{opt} the power extracted by the turbine is maximum and can be shown to be:

$$P_{\max} = \frac{1}{2}\rho A\left(R^3 C_{p\max}\big/\lambda_{opt}^3\right)\omega_m^3 , \tag{2}$$

where R is the radius of the turbine and ω_m its rotational speed. Equation (2) can be re-written as $\omega_m = K_{opt}\sqrt[3]{P_{\max}}$, where K_{opt} is an optimal parameter that defines the optimal power curve of the turbine. Fig. 3 shows the turbine power as a function of the turbine speed over a range of tidal current velocities and also the optimal power curve of the turbine. Note that $P_{\max}$ is not the maximum power the turbine can generate but is the maximum power it can generate at a particular tidal current velocity.

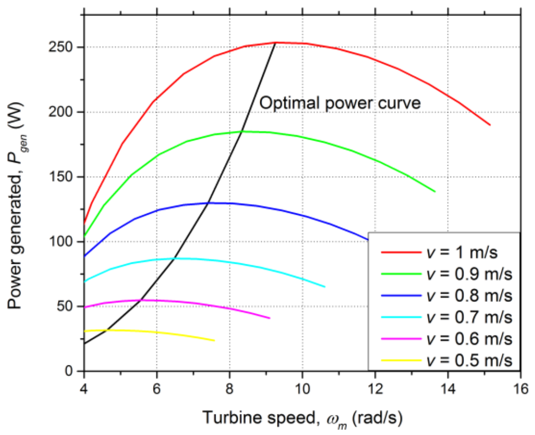

Fig. 3 Turbine power as a function of the turbine speed for a range of tidal current velocities

A simplified numerical model of the tidal turbine was developed based on the C_p-λ curve, shown in Fig. 2, and was used in this work. Fig. 4 shows the schematic of the tidal turbine model used in this work.

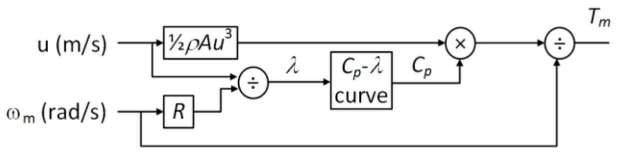

Fig. 4 Schematic of the tidal turbine model

C. Numerical model and control of a Permanent Magnet Synchronous Generator (PMSG)

In this work, a dynamic model of a direct-drive, non-salient permanent magnet synchronous generator (PMSG) coupled to the tidal turbine was used. The direct drive system allows the use of low-speed generators and eliminates the use of the gearbox and normally requires a generator with a higher number of poles. Direct drive systems are competitive for offshore applications [10].

The PMSG was modelled using its state equations [10],[11]:

$$v_{sd} = -R_s i_{sd} + \frac{d}{dt}\lambda_{sd} - \omega_{me}\lambda_{sq}, \quad (3)$$

$$v_{sq} = -R_s i_{sq} + \frac{d}{dt}\lambda_{sq} + \omega_{me}\lambda_{sd}, \quad (4)$$

where $[k_{sqd0}]^{\mathrm{T}} = [k_{sq}\ k_{sd}\ k_{s0}]$ (k represents current, voltage or flux), subscripts d and q refer respectively to the direct and quadrature axis components of voltage (v), current (i) and flux (λ), subscript s is used for stator quantities, R is the resistance and ω_{me} is the electrical angular speed of the rotor. The PM rotor circuit is modelled as an equivalent current source I_f.

The stator d and q winding flux linkages can be expressed as [10],[11]:

$$\lambda_{sd} = -(L_{ls} + L_{md})i_{sd} + \lambda_r \text{ and} \quad (5)$$

$$\lambda_{sq} = -(L_{ls} + L_{mq})i_{sq}, \quad (6)$$

where L_{ls} is the stator leakage inductance, L_{md} and L_{mq} are the magnetising inductance of the d and q axis and λ_r is the rotor flux linked to the d winding of the stator, which is equal to L_{md} I_f.

The electromagnetic torque T_{em} of the synchronous machine is given by:

$$T_{em} = \frac{3p}{2}(\lambda_{sd} i_{sq} - \lambda_{sq} i_{sd}), \quad (7)$$

where p is the number of pole pairs in the machine. The rotor speed can be obtained by solving:

$$J\frac{d\omega_m}{dt} = T_{em} - T_m, \quad (8)$$

where J is the moment of inertia of the rotor and T_m is the load torque.

Speed control of the PMSG is introduced by vector control of the generator side converter. In this work, the zero d-axis current control, as detailed in [10], was used for the generator side control. In this control, i_{ds} is controlled to be zero, which means that the stator current is equal to the q-axis component. The electromagnetic torque T_{em}, substituting $i_{sd}=0$ in Equations (5) and (7), becomes:

$$T_{em} = \frac{3p}{2}(\lambda_r i_{sq}), \quad (9)$$

which indicates that the generator torque is proportional to the stator current. The schematic of the generator side converter controller used in this work is shown in Fig. 5 and constitutes the low-level controller. For the purposes of this work, the grid side converter was not modelled and it was assumed that the DC link voltage of the AC/DC/AC converter was constant.

In this work, a modified form of the Power Signal Feedback (PSF) control, as described in [1], was used for MPPT control of the tidal turbine (high-level controller). The controller requires the optimal power curve of the tidal turbine and uses it to provide the speed reference signal to the generator side converter. Fig. 6 shows the optimal power curve, plotted in Fig. 3, used in a closed loop control scheme of the tidal turbine. The power generated by the tidal turbine is fed into the high-level controller, which then uses the optimal power curve to determine the speed reference for the turbine. The implemented controller is, in effect, a modified version of the optimal tip speed ratio controller, which aims to maintain the optimal tip speed ratio λ_{opt} for all flow conditions..

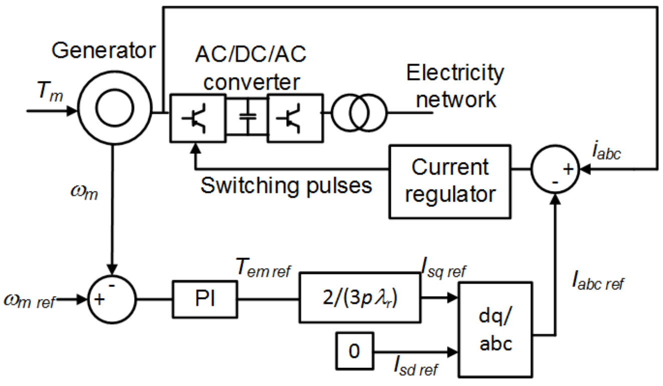

Fig. 5 Schematic of the zero d-axis generator side converter controller

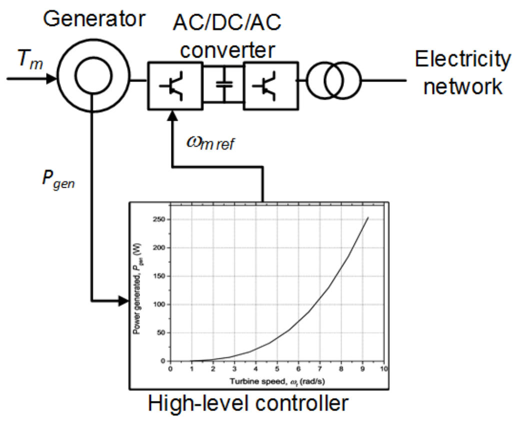

Fig. 6 Power signal feedback controller based MPPT control of the tidal turbine

III. Reinforcement Learning

RL is a class of unsupervised learning algorithms, which has recently been the focus of many studies by the robotics and computer science industries [12]. Within this framework [13], an *agent* (in this case the controller) learns an optimal *policy*, or behaviour, for the maximization of a specified *reward* from direct interactions with its environment.

As shown in Fig. 7, at each step, the agent, which is in a particular *state s*, interacts with the surrounding environment by taking an *action a*. The agent then moves to a new state, *s'*, and the action is followed by a reward, *r*, depending on its outcome. The action selection process is modelled as a Markov decision process based on the *value function*, which expresses the estimate of the future reward. The agent is expected to learn an optimal policy over time for the maximization of the total reward.

RL methods can be divided into three main categories: dynamic programming, temporal difference and Monte-Carlo methods [13]. Of these, temporal difference strategies will be considered here, since they present a real-time implementation. Additionally, in order to limit modelling errors and to pick up changes in the device behaviour over time, model-free techniques are of interest, which use the state-action value function $Q(s,a)$. The state-value function is a measure of the expected return following the selection of action a in state s. Hence, the aim of the RL agent is to select a policy that maximize its value. This is described in Section III-A.

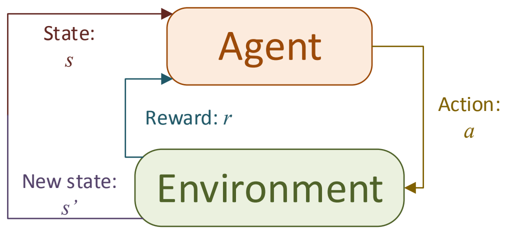

Fig. 7 Block diagram of RL (adapted from [13])

Temporal-difference methods can be divided into on- and off-line schemes depending on how the state-action value function is updated [14]. On-line strategies update the Q-value at every step, whereas off-line techniques wait for a batch of samples in the form (s,a,r,s'). Furthermore, function approximation can be used to improve the performance of RL algorithms for the treatment of large state and action spaces [14]. In Section III-B, an on-line strategy with discrete states is described, namely Q-learning. In Section III-C an off-line scheme with function approximation for the state space is analysed, namely Neural Fitted Q-iteration (NFQ).

A. Exploration Strategy

The action that maximizes the state-action value, and thus the expected future reward, is referred to as the *greedy* action [13]. If the agent selects purely the greedy action, i.e. an exploitative action, it will never visit states other than the usual ones. In fact, other states may result in higher total reward; however, the agent cannot learn this unless it visits them. This is known as the issue of exploration versus exploitation [13]. Hence, it is still beneficial to adopt an approach that ensures some exploration at the expense of exploitation, particularly for the initial stages. Once the simulation has been initialized, the balance may be shifted towards exploitation. This is achieved here with the adoption of an ε-greedy exploration strategy. At each step, with an ε-greedy policy the agent in state s selects the following action [13]:

$$a = \begin{cases} \arg\max_{a' \in A} Q(s,a') & \text{with probability } 1-\epsilon, \\ \text{random action} & \text{with probability } \epsilon, \end{cases} \tag{10}$$

where ε is the exploration rate and A the action space (i.e. all possible actions). The exploration rate is obtained as

$$\epsilon = \begin{cases} \epsilon_0 & \text{if } N \le 0, \\ \epsilon_0/\sqrt{N} & \text{otherwise}, \end{cases} \tag{11}$$

where $N = \sum_{i=1}^{n_a} \boldsymbol{N}(s,a_i) - N_s$, with $\boldsymbol{N}(s,a)$ indicating the total number of visits to the state-action pair (s,a), n_a the number of actions, N_ε the minimum number of visits to a state for random exploration and ε_0 the initial exploration rate. Equation (11) ensures a sufficient level of exploration at the start of RL control, with the focus being shifted to the exploitative action as learning progresses.

The learning rate determines the proportion of new and old knowledge that is retained during learning and is calculated as

$$\alpha = \begin{cases} \alpha_0 & \text{if } \boldsymbol{N}(s,a) \le N_\alpha, \\ \alpha_0/\boldsymbol{N}(s,a) & \text{otherwise}, \end{cases} \tag{12}$$

where N_α is a predefined parameter. Equation (12) ensures sufficient learning when each state-action pair is visited for the first few times. As learning progresses, older knowledge is given greater importance to limit the impact of sensor noise.

B. Q-learning with discrete states

Q-learning is an on-line temporal difference scheme that is very popular with the robotics industry [12]. Originally proposed by Watkins [15], [16], Q-learning updates the value function using the optimal known policy, which may not be the policy being followed due to exploration. For this reason, it is labelled as an off-policy strategy [13]. Using discrete states and actions, the state-action value update is expressed as [13]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)\right], \tag{13}$$

where γ is the discount factor, which is used to discount future rewards, and α is the learning rate, which regulates how much of the previous learning is retained in the update of the action-value table.

Q-learning is guaranteed to converge for discrete actions and states, a bounded reward variance, the use of a discount factor and a properly decaying learning rate [17].

Fig. 8 shows the algorithm of discrete Q-learning that was used in the learning process.

C. Neural Fitted Q-Iteration

Function approximation can be used to treat the state-action value as a continuum and improve the performance of RL algorithms [14]. A large number of discrete states can result in an excessive learning time, since the agent may have to experience each state before convergence. Function approximation can significantly decrease learning time by presenting a smaller number of features instead, which allow the controller to generalize for unseen states [14]. Although linear features have resulted in the development of successful RL algorithms [13],[14], neural networks (NNs) represent a more powerful, non-linear tool that allows global approximation also for non-linear problems [18]. Their main advantage is the capacity to generalize for unseen situations [19].

```
Input: $\varepsilon_0, \gamma, N_\varepsilon, N_\alpha, S, A$

Initialize $\boldsymbol{Q} \leftarrow 0$;
Initialize $\boldsymbol{N} \leftarrow 0$;
Get number of actions $n_a$;
Initialize $a$;

For each step until end:
    Observe current continuous state $s$;
    Get corresponding discrete state $s_d \in S$;
    Update $N(s_d, a) \leftarrow N(s_d, a) + 1$;
    Update exploration rate $\varepsilon$ with (11);
    Select an $\varepsilon$-greedy action $a$ with (10);
    Apply action $a$;
    Observe reward $r$;
    Observe new continuous state $s'$;
    Get corresponding discrete state $s'_d \in S$;
    Update $Q(s_d, a)$ with (13);
End
```

Fig. 8 Algorithm of discrete Q-learning adapted from [13]

NNs are machine learning algorithms that can be used to learn the non-linear mapping between specified input and output data [20]. They are inspired from biological brains and are thus composed of neurons arranged in layers. Here, a feedforward multi-layer perceptron with a single hidden layer with m neurons is considered, as shown in Fig. 9. Additionally, the input and hidden layer present a bias term, which is required to find the intercept of the function fitted by the NN. Each layer l has input and output variables, denoted as $\boldsymbol{x}_l$ and $\boldsymbol{y}_l$ respectively. The input to the NN corresponds to $\boldsymbol{y}_1$, while the output to $\boldsymbol{y}_3$. The weight matrices between each two layers are defined as $\boldsymbol{W}_j$, with j=1,2, and the bias matrices as $\boldsymbol{b}_i$. Using forward propagation [20], the NN returns an estimate for the output value given input data by propagating the signal forward in the network. The input and output vectors for each layer l=2,3 are thus computed as [20]:

$$x_l = \boldsymbol{W}_{l-1}\boldsymbol{y}_{l-1} + \boldsymbol{b}_{l-1}, \tag{14}$$

$$\boldsymbol{y}_l = e_l(\boldsymbol{x}_l).s \tag{15}$$

In Equation (14), e_l denotes the activation function for the neurons in layer l. As shown in Fig. 9, the hidden layer uses the *tanh* activation function, which is the preferred choice for a small number of layers [21], while the output layer uses a linear activation function.

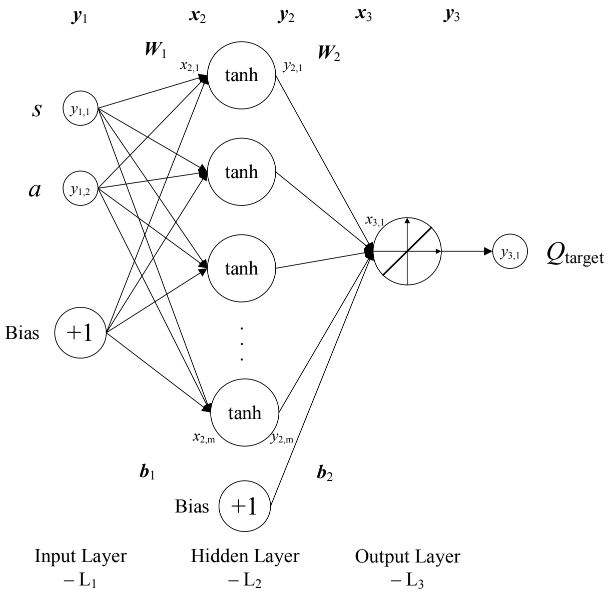

Fig. 9 Schematic diagram of the feedforward NN used in NFQ

In a NN, learning occurs by tuning the weight matrices so that the network provides an accurate mapping between the provided input and output data. To simplify the notation, here we denote the mapping provided by the NN between input $\boldsymbol{i}$ and output $\boldsymbol{o}$ as

$$\boldsymbol{o} = f(\boldsymbol{i}). \tag{16}$$

Training occurs through a process known as backward propagation, where the error signal is propagated backwards from the output to the input layer. The reader is referred to [20] for more information. Additionally, much more efficient techniques have been developed to train NNs using batch of samples rather than an individual sample. Here, we use the efficient Levenberg-Marquardt algorithm for training of the NN in batch mode [22], even though the Rprop algorithm was originally proposed for use with NFQ [23].

NNs present a major disadvantage when used to fit the state-action value in RL, in addition to their much greater computational cost as compared with linear features. If the weights of the NNs are adjusted for a particular state-action pair, then unpredictable changes also occur at other places in the state-action space according to (11) and (12). These problems have been alleviated through the use of off-line, batch learning in NFQ, originally developed by [18]. As shown in Fig. 10, with this procedure, the algorithm is run with an ε–greedy policy using the estimate of the state-action value provided by the NN. At each step, the current state, action, reward and new state are stored as samples of the form (s,a,r,s'). After a predefined number of samples is collected, the NN weights are updated using the complete set of transition experiences (i.e. all past samples) and the Levenberg-Marquardt algorithm. In particular, this procedure is repeated for a specified number of epochs, $k_{\max}$. This approach has been found to work well, and is more computationally efficient than setting a limit for the error [19]. In Fig. 10, the notation $\boldsymbol{S}(:,j)$ indicates the jth column vector of list $\boldsymbol{S}$.

```
Input: $\varepsilon_0, \gamma, N_\varepsilon, N_\alpha, A$

Initialize $W$ (NN weights) randomly;
Initialize $S \leftarrow []$ (empty samples list);
Get number of actions $n_a$;

For each step until end:
    Observe current continuous state $s$;
    Get corresponding discrete state $s_d$;
    Update exploration rate ε with (11);
    For each action $a'$ in $A$:
        $Q(s,a') \approx f(s,a')$;
    End
    Select an ε-greedy action $a$ with (10);
    Apply action $a$;
    Observe reward $r$;
    Observe new continuous state $s'$;
    Store $\boldsymbol{S} \leftarrow [\boldsymbol{S}; (s,a,r,s')]$;
    If condition met for off-line NN update:
        Get the number of samples $n_S$;
        Initialize $Q_{\text{target}}$=0 with size $(n_S, n_a)$;
        For $k$=1:$k_{\max}$
            For each sample $i$ in S
                For each action $a'$ in $A$:
                    $Q(s',a') \approx f(\boldsymbol{S}(i,4),a')$;
                End
                $Q_{\text{target}}(i) = S(i,3) + \gamma \max_{a' \in A} Q(s',a')$;
            End
            Train NN as in [22] with input $S(:,1)$ and $S(:,2)$ and output $Q_{\text{target}}$;
        End
    End
End
```

Fig. 10 Algorithm of Neural Fitted Q-iteration, adapted from [18]

IV. Reinforcement Learning in MPPT Control of Tidal Turbines

In this work, the two variants of RL, described in the previous section, are used to determine the optimal coefficient K_{opt} of the optimal power curve for the tidal turbine. According to Fig. 7, at every RL time step, the RL controller (the agent), observes the current power curve coefficient K_{opt} (the state s) of the environment and chooses a change in the K_{opt} value (the action a) with the aim of maximising the reward r accrued over time. Fig. 11 shows the schematic of the RL implementation. The high-level controller, as seen in Fig. 6, learns the optimal power curve coefficient through interacting with its environment and learning from the observations made. To account for possible wave action and turbulence in tidal flows, the instantaneous power generated needs to be averaged over a time period T before it can be used to calculate the reward r. During this time period T the state s and the action a are kept constant. The state space, the action space and the reward function used in this work are described in detail in this section.

A. State Space

As mentioned earlier, the coefficient of the power curve used in the high-level controller K_{opt} was considered to be the state variable in this work. The RL state space thus becomes:

$$S = \{s \mid s_k = (K_{opt}), k \in [1,2,\ldots,N]\}\,. \tag{17}$$

N in Equation (17) is the number of equally divided segments in the entire range of K_{opt} considered in this work. The value of N significantly affects the learning process, with a larger N increasing the learning period, while smaller N reducing the learning accuracy. Thus, N needs to be appropriately chosen to balance these two opposing requirements.

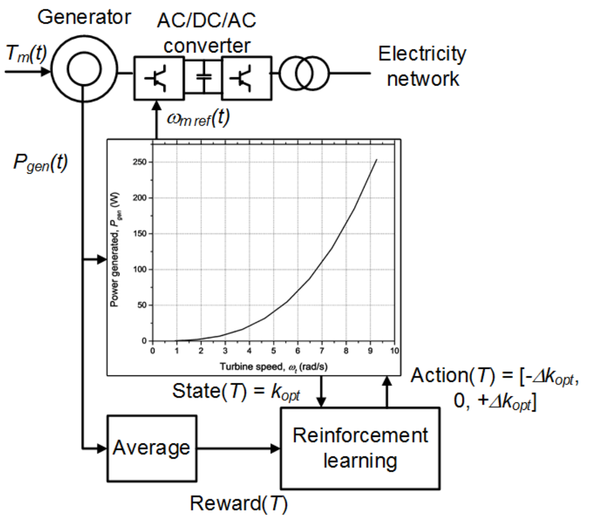

Fig. 11 Reinforcement learning applied in MPPT control of tidal turbine

B. Action Space

From the current state s, the action taken by the agent involves changing the value K_{opt} by a small value ΔK_{opt}. The action space for the RL controller thus becomes:

$$A = \{a \mid -\Delta K_{opt}, 0, +\Delta K_{opt}\} \tag{18}$$

The states corresponding to the maximum and the minimum value of K_{opt} have limits on the actions that can be taken to ensure K_{opt} remains within the state space boundary.

C. Reward

After an action a is taken from state s, the agent receives a reward r. The aim of the RL algorithm is to maximise the total future rewards instead of the just the immediate reward r. In the case of MPPT control of tidal turbines, the main objective of the controller is to maximise the average power output of the turbine by selecting the optimal power curve coefficient. Considering that the average power generated by the tidal turbine depends on the tidal current velocity and the presence of any turbulence and wave action, offline simulations were run with different current velocities, turbulence intensities and wave actions, to determine the maximum average power max($P_{gen\ avg}$) seen over the averaging horizon T_{avg}. The average power over the period T_{avg}, normalised with respect to the maximum $P_{gen\ avg}$ (from the offline simulations) was used as the reward function. Additionally, the magnitude of ΔK_{opt} selected can mean that the difference in the average power between two neighbouring states may not be very high. To magnify the effects of a change in K_{opt} and for the algorithm to converge faster, the reward function was then raised to a power value m. The reward function used thus became:

$$r = \left\{ \left[\frac{P_{genavg}}{\max(P_{genavg})} \right]^m \right. . \tag{19}$$

E. Learning Process

Applying RL in tidal turbine MPPT control consists of two distinct parts. The first one is the learning process itself, through which the agent learns what best action to take at every state. This process involves updating the Q-table or Q-function based on the learning experience.

Fig. 8 and Fig. 10 show the algorithms of discrete Q-learning and NFQ that were used in the learning process.

The learning process is stopped after a specified number of iterations are complete.

F. Application Process

Once the optimal power curve coefficient K_{opt} has been identified by the RL algorithm, the learning process is stopped and the high-level controller works independently with the power curve as shown in Fig. 6. This constitutes the application process and is continued until another learning process is required. This could be necessitated by changes in turbine characteristics caused by changes in the long term flow conditions and/or other causes like biofouling.

V. Simulation Results

Simulations using both the RL variants were completed for two different tidal current velocities, without any turbulence and wave action. Three other cases were then assessed to study the performance of the algorithms in the presence of turbulence and wave action. Since only a single tidal turbine has been considered here, the RL algorithms would converge to the same K_{opt} value for all the cases considered. The performance of the two RL algorithms tested will then be compared for the simulated cases.

A. RL parameters and simulation setup

The state space, or the range of K_{opt}, used in this work was between 1 and 3 with $\Delta K_{opt} = 0.1$ ($n_s = 21$). The state space was decided from prior experience of the K_{opt} of the tidal turbine obtained through tank tests and offline simulations. Tank tests of a physical model of the turbine showed that the turbine stalls at a tip speed ratio of around 3.53 (or turbine rotational speed of around 45 rpm) for a current velocity of 0.8 m/s. The range of K_{opt} was chosen keeping this in mind, such that even at the lowest K_{opt} value, the turbine would not stall.

The number of actions n_a was set to 3, as explained in Section IV B. The Q-table, thus, has a dimension of 21x3. For NFQ, a neural network with one hidden layer having 5 nodes was chosen. The simulation, for every RL iteration, was run for 15 s (T_{avg}), out of which the power generated by the turbine over the last 10 s was used to calculate $P_{gen\ avg}$. If any action a (change in K_{opt}) was required in the current iteration, the change was made 1 s after the simulation starts. The simulation time of 15 s, per RL iteration, was chosen because waves with period between 1 s and 3 s were tested with the scaled turbine model.

For the RL algorithms, the initial learning rate, exploration rate, and discount factor were set to 0.5, 0.6 and 0.95 respectively. The number of RL iterations was set to 1000 for both the discrete Q-learning and the neural-fitted Q-iteration algorithms. The elements of the Q-table were initialised to zero and the algorithms were set to start at the 1st state every time.

B. *Learning with constant current velocity*

As the first test, the tidal turbine was exposed to a constant tidal current velocity over the duration of the learning process. Two current velocities of 0.6 m/s and 0.8 m/s were used and both the RL algorithms were tested.

Fig. 12 shows the Q-learning algorithm converging towards the optimum K_{opt} over the 1000 time steps simulated. It takes the algorithm around 350 time steps to converge. Note that even after converging, the algorithm explores the neighbouring K_{opt} values in the time steps after 350. This would identify any significant change in the turbine characteristics or flow conditions and update the RL learning process accordingly. Fig. 12 also shows the mean power generated by the turbine at each RL time step, which finally converges to its maximum value. The convergence of Q-learning with the 0.6 m/s tidal current velocity is shown in Fig. 13. With the new tidal current velocity, the algorithm takes almost the same number of steps to converge. Since it is the same single turbine being considered, the Q-learning algorithm converges to the same K_{opt} value. The convergence of the NFQ algorithm is shown in Fig. 14 for the tidal current velocity of 0.8 m/s. The algorithm converges to the optimal K_{opt} in about 90 RL time steps.

C. *Learning with constant current velocity and turbulence*

As the next test, the two RL algorithms were employed with turbulence added to the constant tidal current velocity. The effect of turbulence was modelled using random noise superimposed on the constant tidal current velocity. Fig. 15 and Fig. 16 show the convergence of the two RL variants over the RL time steps simulated. Discrete Q-learning still takes approximately 350 time steps to converge, while NFQ takes approximately 90. The optimal power curve coefficient K_{opt} output by the learning process is still the same as seen in the case without any turbulence. This is as expected and proves that the RL algorithms are identifying the correct K_{opt}.

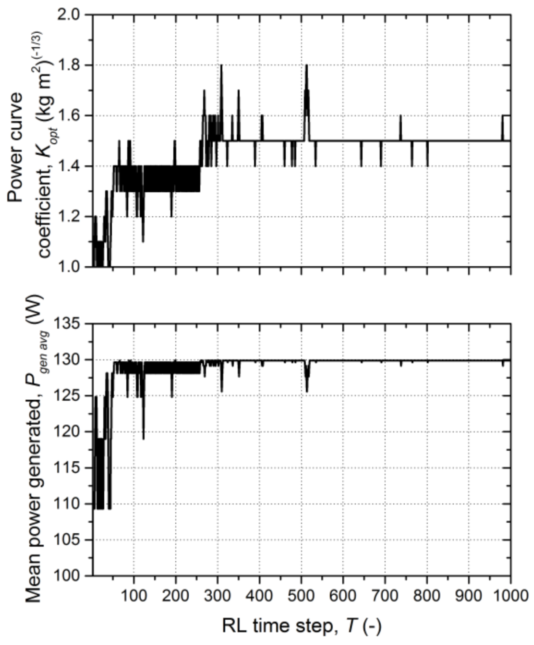

Fig. 12 Q-learning results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s

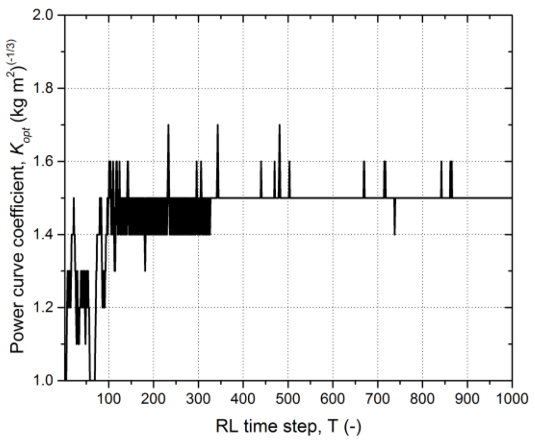

Fig. 13 Q-learning results with the tidal turbine simulated in a constant tidal current velocity of 0.6 m/s

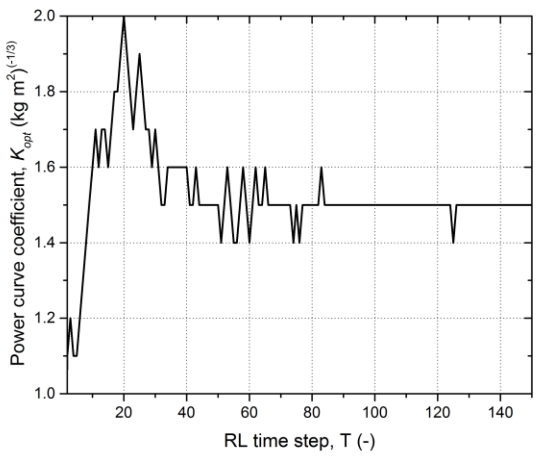

Fig. 14 NFQ results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s

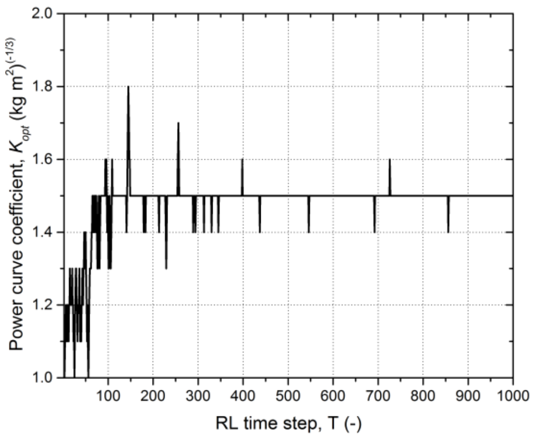

Fig. 15 Q-learning results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s with turbulence

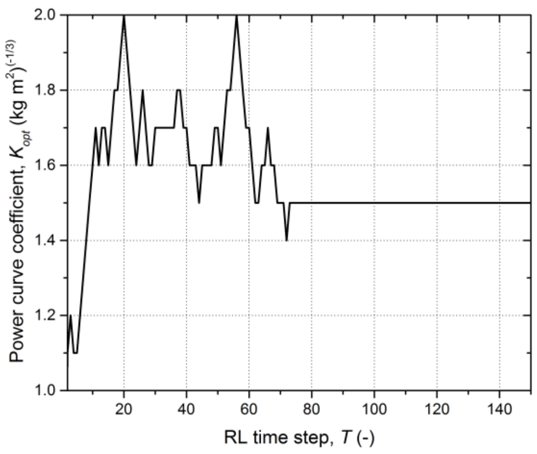

Fig. 16 NFQ-iteration results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s with turbulence

D. Learning with constant current velocity and wave action

This section examines the learning process when sinusoidal waves of different amplitudes and frequencies are superimposed on the tidal current velocity. Wave action on the turbine was modelled by adding a sinusoidal velocity signal to the constant tidal current velocity. Two wave induced velocity amplitudes were tested – 0.1 m/s amplitude, 2 s period and 0.15 m/s amplitude, 3 s period. Fig. 17 and Fig. 18 show the two RL algorithms' converge to the optimal K_{opt} value for the second wave tested. As has been seen with the earlier results, the two algorithms converge to $K_{opt} = 1.5$. Q-learning took around 400 time steps to converge, while NFQ took around 80 time steps.

E. Learning with constant current velocity, wave action and turbulence

This test was arranged to make the learning process most challenging, with both turbulence and wave action, as setup in the previous simulation runs, superimposed on the constant tidal current velocity. Even in this learning environment, both the RL algorithms converged to the optimal K_{opt} value in approximately 400 (for Q-learning) and 90 (for NFQ) time steps. The convergence of these two algorithms to the optimal K_{opt} value are shown in Fig. 19 and Fig. 20.

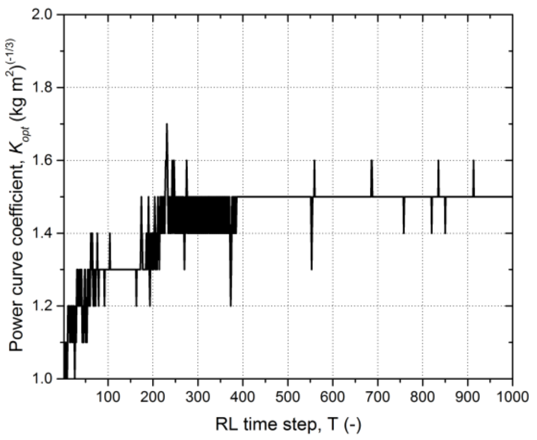

Fig. 17 Q-learning results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s with a 0.15 m/s amplitude, 3 s sinusoidal velocity superimposed

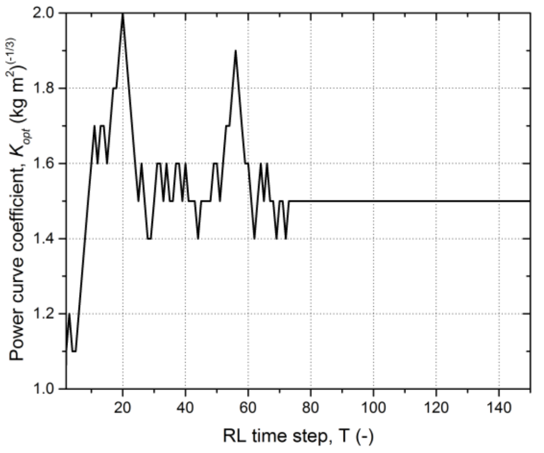

Fig. 18 NFQ results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s with a 0.15 m/s amplitude, 3 s sinusoidal velocity superimposed

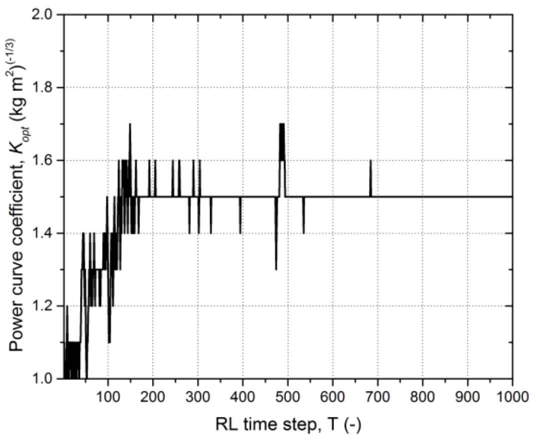

Fig. 19 Q-learning results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s with a 0.1 m/s amplitude, 2 s sinusoidal velocity and turbulence superimposed

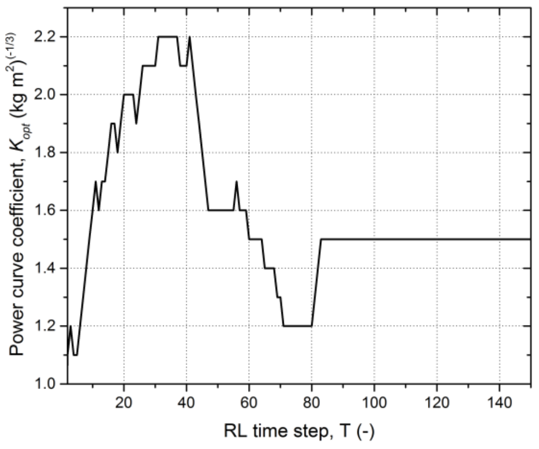

Fig. 20 NFQ results with the tidal turbine simulated in a constant tidal current velocity of 0.8 m/s with a 0.1 m/s amplitude, 2 s sinusoidal velocity and turbulence superimposed

F. Adaptive learning with RL

In this section, the RL algorithms' performance in the identification of K_{opt} when the turbine characteristics change is examined. As was mentioned earlier, one of the advantages of the RL approach is that the algorithm can respond to changes in the turbine characteristics brought about by modifications in the long term flow conditions or by biofouling.

Biofouling affects the roughness of the turbine blades and modifies its C_p-λ curve. Based on the work reported in [24] the modified C_p-λ curve of the turbine was assumed to be the one shown in Fig. 21. As shown in [24] through scaled physical testing and modelling work, the value of C_p, decreased for the whole range of λ, with the peak of the curve shifting towards a smaller λ. This new C_p-λ curve of the turbine modified the optimal power curve and the corresponding K_{opt}, which was found, from offline simulations, to be 1.36.

The Q-learning simulation with the modified turbine was initialised with the final Q-matrix and state obtained from the simulation described in Section V C done with the original turbine. This simulates the RL process working continuously over the two periods between which the turbine C_p-λ curve had changed. The turbine was simulated in a tidal current flow of 0.8 m/s with turbulence. The learning rate and the exploration rate for the algorithm were increased to 0.75 and 0.75 respectively to make the learning and update process faster. Fig. 22 shows the Q-learning algorithm converge to the optimal K_{opt} value in approximately 410 time steps. The excursions beyond this time step is because of the higher exploration rate used.

Fig. 23 shows the NFQ algorithm, initialised with the same neural network weights as from the simulation described in Section V C, converging to the new optimal K_{opt} value in 82 time steps. This demonstrates how the algorithm can deal with changes in the turbine characteristics.

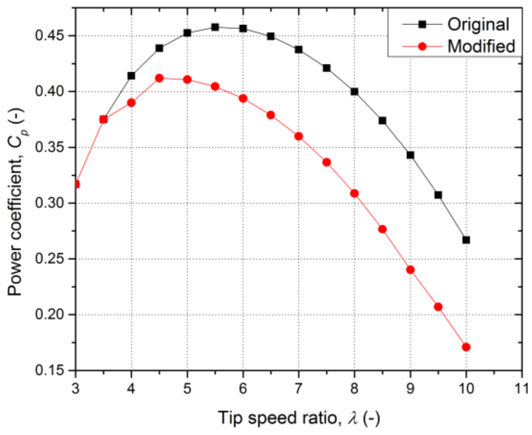

Fig. 21 The original and the modified C_p-λ curve of the turbine.

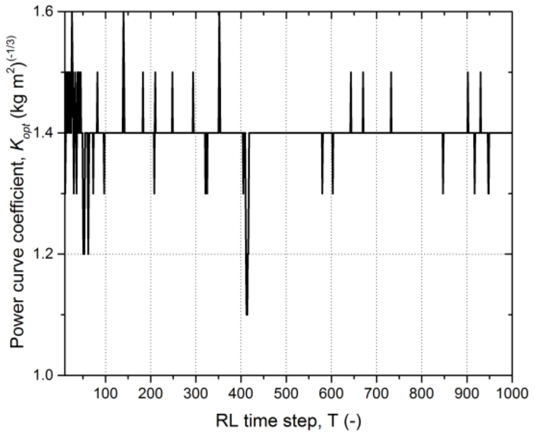

Fig. 22 Q-learning results with the tidal turbine, with the modified C_p-λ curve, simulated in a constant tidal current velocity of 0.8 m/s and turbulence superimposed

VI. Discussion

The results presented and discussed in the preceding section showed the learning process with the two RL variants. In all the cases discussed, for the single turbine, the K_{opt} value was identical, as was expected, which proves the applicability of the methods used. In this work, it was assumed that the flow was uniform over the rotor. With non-uniform flow, the RL algorithms will take longer to converge.

During the learning process, the controller moves through various K_{opt} values before converging on the optimal one. Since the learning process only occurs for an extremely short period of time relative to the life time of a tidal turbine, the higher loads that may be encountered during the learning process will not impact the fatigue life of the turbine significantly.

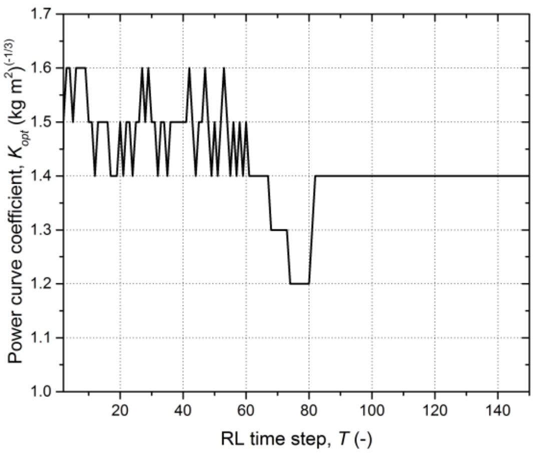

Fig. 23 NFQ results with the tidal turbine, with the modified C_p-λ curve, simulated in a constant tidal current velocity of 0.8 m/s and turbulence superimposed

The real interest in the presented algorithms is in learning the optimal power curve coefficients within arrays of tidal devices. In arrays with rows of tidal devices, the optimal power curves for the individual generators may need to be different to maximise the energy yield across the array. Through on-line learning algorithms, like the ones presented in this paper, the best optimal power curves for each tidal device can be determined. In theory, identifying the optimal power curves using RL in the arrays will take longer to converge when compared to the single turbine case.

Here, two distinct reinforcement learning algorithms have been investigated. In all cases, NFQ has been shown to have superior convergence properties. This is mainly due to its reliance on function approximation, with the NNs helping the controller generalize for unseen situations. Additionally, the greater the number of states and actions, the greater the benefits of function approximation are expected to be [14].

The presented learning algorithms can also be used to optimise turbine performance due to changes in the device characteristics due to biofouling or due to long term changes in the flow patterns. In such situations when the operator feels a change in the optimal power curve is required, the application process is stopped after which another learning process is initiated. An example of such a situation and how the two RL algorithms responded to a change in the turbine's C_p-λ curve was also shown in the paper.

VII. CONCLUSIONS

This paper discussed two, model-free RL algorithms – Q-learning and NFQ - to identify the optimal power curve for single tidal turbines. The performance of the algorithms in MPPT control of the single turbine was assessed in different tidal current flow conditions. The two algorithms converged to the optimal power curve coefficient even for the cases with wave action and turbulence added to the different constant tidal current velocities, which proves its applicability to the control of tidal turbines. The proposed algorithms were tested on a tide-to-wire model of a single turbine, which was also described in the paper. The advantages of the RL based approach are that there is no need for any prior knowledge of the system and that the control system can adapt to changes in the turbine characteristics brought about by biofouling, non-critical failures, or long term changes in the flow characteristics. Whether these algorithms can be efficiently extended to learn the optimal power curves of tidal turbines in arrays will be studied in further work.

ACKNOWLEDGMENT

The authors gratefully acknowledge that this research was supported by the Engineering and Physical Sciences Research Council (EPSRC) through the SuperGen UK Centre for Marine Energy Research (grant EP/M014738/1). Mr Anderlini's project was funded by the Energy Technologies Institute, the EPSRC (grant EP/J500847/) and Wave Energy Scotland.

REFERENCES

[1] M. A. Abdullah, A. H. M. Yatim, C. W. Tan, and R. Saidur, "A review of maximum power point tracking algorithms for wind energy systems," *Renewable and Sustainable Energy Reviews*, vol. 16, pp. 3220–3227, 2012.

[2] E. Muljadi, A. Wright, V. Gevorgian, J. Donegan, C. Marnagh, and J. McEntee, "Turbine control of a tidal and river power generator," in *Proc. of the IEEE North American Power Symposium, Colorado*, 2016.

[3] S. Benelghali, M. Benbouzid, and J. F. Charpentier, "Modeling and control of a marine current turbine driven doubly-fed induction generator," *IET Renewable Power Generation*, vol. 4, pp. 1–11, 2010.

[4] C. Wei, Z. Zhang, W. Qiao, and L. Qu, "Reinforcement-learning-based intelligent maximum power point tracking control for wind energy conversion systems," *IEEE Transactions on Industrial Electronics*, vol. 62, p. 6360-6370, 2015.

[5] T. Stallard, T. Feng, and P. K. Stansby, "Experimental study of the mean wake of a tidal stream rotor in a shallow turbulent flow," *J. Fluids Struct.*, vol. 54, pp. 235–246, 2015.

[6] G. S. Payne, T. Stallard, and R. Martinez, "Design and manufacture of a bed supported tidal turbine model for blade and shaft load measurement in turbulent flow and waves," *Renew. Energy*, vol. 107, pp. 312–326, 2017.

[7] A. S. Bahaj, W. M. J. Batten, and G. McCann, "Experimental verifications of numerical predictions for the hydrodynamic performance of horizontal axis marine current turbines," *Renew. Energy*, vol. 32, no. 15, pp. 2479–2490, 2007.

[8] M. O. L. Hansen, *Aerodynamics of Wind Turbines*. Earthscan, 2008.

[9] G. Germain, "Marine current energy converter tank testing practices," in *Proc. of the 2nd International Conference on Ocean Energy, Brest*, 2008.

[10] B. Wu, Y. Lang, N. Zargari, and S. Kouro, *Power Conversion and Control of Wind Energy Systems*. Wiley-IEEE Press, 2011.

[11] N. Mohan, *Advanced Electric Drives: Analysis, Control, and Modeling Using MATLAB/Simulink*. Wiley, 2014.

[12] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish, “Reinforcement learning and optimal adaptive control: An overview and implementation examples,” *Annu. Rev. Control*, vol. 36, no. 1, pp. 42–59, 2012.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. MIT Press, 1998.

[14] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010.

[15] C. J.Watkins, “Models of delayed reinforcement learning,” Ph.D. dissertation, Cambridge University, 1989.

[16] C. J. C. H. Watkins and P. Dayan, “Technical note: Q-learning,” vol. 8, no. 3, p. 279-292, 1992.

[17] T. Jaakkola, M. I. Jordan, and S. P. Singh, “On the convergence of stochastic iterative dynamic programming algorithms,” *Neural Comput*, vol. 6, no. 6, 1994.

[18] M. Riedmiller, *Lect. Notes Comput. Sci.*, 2005, vol. 3720LNAI, ch. Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method, pp. 317–328.

[19] ——, *Lect. Notes Comput. Sci.*, 2012, vol. 7700 LECTU, ch. 10 Steps and some tricks to set up neural reinforcement controllers, pp. 735–757.

[20] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesus, *Neural Network Design*. PWS Publishing, 1996.

[21] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the Marquardt algorithm,” *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[23] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: The rprop algorithm,” in *Proc. of IEEE Int. Conf. Neural Networks*, vol. 1993-Janua, 1993, pp. 586–591.

[24] J. M. Walker, K. A. Flack, E. E. Lust, M. P. Schultz, and L. Luznik, “Experimental and numerical studies of blade roughness and fouling on marine current turbine performance,” *Renewable Energy*, vol. 66, pp. 257 – 267, 2014.