# Controlling Mixed-Mode Fatigue Crack Growth using Deep Reinforcement Learning

Yuteng Jin, M.S.
Harold Vance Department of Petroleum Engineering, College of Engineering, Texas A&M University, College Station, Texas, USA

Prof. Siddharth Misra, Ph.D.
Harold Vance Department of Petroleum Engineering, College of Engineering, Texas A&M University, College Station, Texas, USA
Department of Geology and Geophysics, College of Geosciences, Texas A&M University, College Station, Texas, USA



## Abstract

Mechanical discontinuity embedded in a material plays an essential role in determining the bulk mechanical, physical, and chemical properties. This paper is a proof-of-concept development and deployment of a reinforcement learning framework to control both the direction and rate of the growth of fatigue crack. The reinforcement learning framework is coupled with an OpenAI-Gym-based environment that implements the mechanistic equations governing the fatigue crack growth. Learning agent does not explicitly know about the underlying physics; nonetheless, the learning agent can infer the control strategy by continuously interacting the numerical environment. The Markov decision process, which includes state, action and reward, is carefully designed to obtain a good control policy. The deep deterministic policy gradient algorithm is implemented for learning the continuous actions required to control the fatigue crack growth. An adaptive reward function involving reward shaping improves the training. The reward is mostly positive to encourage the learning agent to keep accumulating the reward rather than terminate early to avoid receiving high accumulated penalties. An additional high reward is given to the learning agent when the crack tip reaches close enough to the goal point within specific training iterations to encourage the agent to reach the goal points as quick as possible rather than slowly approaching the goal point to accumulate the positive reward. The reinforcement learning framework can successfully control the fatigue crack propagation in a material despite the complexity of the propagation pathway determined by multiple goal points.

**Keywords**: Reinforcement learning; DDPG; Discontinuity; Fatigue; Propagation; Control

# 1. Introduction

## 1.1.Study on the mixed-mode fatigue crack growth

Under cyclic loading, fatigue in a material leads to the initiation and propagation of cracks. In general, when a fatigue occurs after about 10000 cycles, it is called a high cycle fatigue (stress-based) that can be characterized by elastic deformation, whereas a low cycle fatigue (strain-based) can be characterized by plastic deformation in each cycle. In linear elastic fracture mechanics, the fatigue crack growth can be divided into two periods, namely initiation and propagation **(McBagonluri and Soboyejo, 2005)**. In the initiation period, the microcracks form with preference in regions of high stress concentration. The propagation period can be further divided into two stages, namely stage I and II. The stage I fatigue crack growth is characterized by low stress intensity factors, whereas the fatigue crack growth in stage II is characterized by large stress intensity factor range and long crack length. In stage II, the fatigue crack growth rate has a power law relationship with respect to the stress intensity range, which is generally described by the Paris–Erdogan formulation.

Mixed-mode fatigue crack growth is widely studied. Various criteria for the prediction of mixed-mode fatigue crack growth direction and rate have been proposed. For example, maximum tangential stress criterion, minimum strain energy density criterion, J-criterion for the prediction of fatigue crack growth direction, while the effective stress intensity factors, strain energy density factors, and J-integral approach for the prediction of fatigue crack growth rate **(Qian and Fatemi, 1996)**. **Sander and Richard (2006)** conducted both experimental and numerical investigations to understand the influence of loading direction on the mixed-mode fatigue crack growth in the Compact Tension Shear specimen. Their experimental results show that the retardation effect decreases with increasing Mode II overload. The fatigue crack growth rate decreases and the direction changes due to the block loading. Their numerical simulation in ABAQUS suggests plastic deformations occur and the stress distribution changes due to mixed-mode overloads. **Alegre et al. (2007)** performed numerical simulation of mixed-mode fatigue crack growth in an anti-return valve using ANSYS. The simulation result agrees well with experimental observation, which validates the use of the maximum circumferential stress criterion for predicting the fatigue crack growth direction. **Ding et al. (2007)** proposed a method based on the detailed elastic–plastic

stress analysis to predict both fatigue crack growth rate and direction in 1070 steel. The predictions match the experiment result.

Fatigue crack also exists in quasi-brittle materials. **Le et al. (2014)** conducted a set of experiments on fatigue crack kinetics for Berea sandstone. They developed a model that accounts for the size effect of the specimens based on the experimental observations. They concluded that the model is also applicable to other quasi-brittle materials including concrete and ceramics. **Ray and Kishen (2011)** proposed an analytical dimensionally homogeneous model for fatigue crack growth in concrete using dimensional analysis. Their model considers the effect of structural size, loading ratio, initial crack length, tensile strength, and fracture toughness.

Being able to control the growth of crack demonstrates the human understanding and capability of manipulating the bulk mechanical, chemical and physical properties. In this paper, we focus on the stage II fatigue crack growth for high cycle fatigue with crack length higher than 10μm, such that the fatigue crack propagation honors the Paris–Erdogan law. We propose a reinforcement learning technique that is capable of controlling both the fatigue crack growth rate and direction.

### 1.2. Application of the DDPG reinforcement learning algorithm

Reinforcement learning (RL) is a machine learning technique developed specifically for solving the sequential decision-making and control problems. Unlike the supervised and unsupervised learning, reinforcement learning learns from the experience obtained through dynamically interacting with the environment by taking actions, observing the state of the environment, and receiving rewards based on the action and state. The supervised and unsupervised learning aim to learn certain representations by exploiting the existing dataset, whereas reinforcement learning agent learns to develop the optimal control policy for a particular problem by focusing on the maximization of the expected cumulative long-term reward and balancing exploration and exploitation **(Li and Misra, 2021)**. The reinforcement learning models the continuous interaction with environment as a Markov decision process (MDP), where the next state of the environment depends only on the current state of the environment and the action taken by the agent. The MDP assumption is useful for modeling many real-world sequential decision-making problems because it eliminates the need for storing the entire history of states and actions,

which tremendously reduce the memory and computation requirement. However, the MDP assumption is not necessarily true for all systems and processes.

In off-policy reinforcement learning, the RL agent evaluates a target policy based on the actions generated by following a behavior policy that is different from the target policy. Deep deterministic policy gradient (DDPG) algorithm is an off-policy reinforcement learning algorithm introduced by **Lillicrap et al. (2015)**. This algorithm combines the ideas from deterministic policy gradient method, deep Q-learning, and actor-critic techniques. By implementing deep neural networks in both actor and critic models, DDPG algorithm is suitable for control problems with continuous state and action space. **Kendall et al. (2019)** proposed the first application of DDPG algorithm in self-driving. By taking a single image as input, their model learned a policy for lane following task in tens of training episodes. **Qiu et al. (2019)** proposed an energy management algorithm based on DDPG in energy harvesting wireless networks. The algorithm achieves better performance compared with other algorithms in terms of long-term average net bit rate. **Ma et al. (2019)** applied several deep reinforcement learning algorithms on the waterflooding optimization problem to find the optimal water injection rate under geological uncertainties by maximizing the net present value. They found the DDPG algorithm requires the least training episodes and converges the fastest. But it can stuck into the local optimum solution due to inferior exploration-exploitation implementation. **Li and Misra (2021)** applied the DDPG algorithm to develop an automated history matching technique that improves the forecast of hydrocarbon production. The technique works well for data generated using a multi-stencil fast marching reservoir simulator. **Jin and Misra (2021)** demonstrated a successful control of crack propagation in material under biaxial stress.

### 1.3. Novelty and significance of the work

The paper is a proof-of-concept development and deployment of a DDPG reinforcement learning framework for controlling both the fatigue crack growth rate and direction. To ensure computational tractability of this proof-of-concept development, we assumed a 2D material containing an initial linear crack. The RL agent is trained by interacting with a numerical environment, which implements mechanistic equations for 2D mixed-mode fatigue crack growth under sinusoidal biaxial stress. The RL agent does not explicitly know about the underlying physics of fatigue crack growth; nonetheless, it can infer the control policy by continuously

interacting the numerical simulation within a training environment that is developed based on the OpenAI Gym environment. The following scientific questions are investigated and answered in this paper:

- Is the deep deterministic policy gradient (DDPG) reinforcement learning algorithm capable of controlling both the fatigue crack growth rate and direction?
- How to design generalizable reward function that facilitate the training of the RL agents on the desired control problem?
- How do the design of reward function and the size of neural network affect the training outcome and efficiency of the reinforcement-learning-based control of discontinuity?

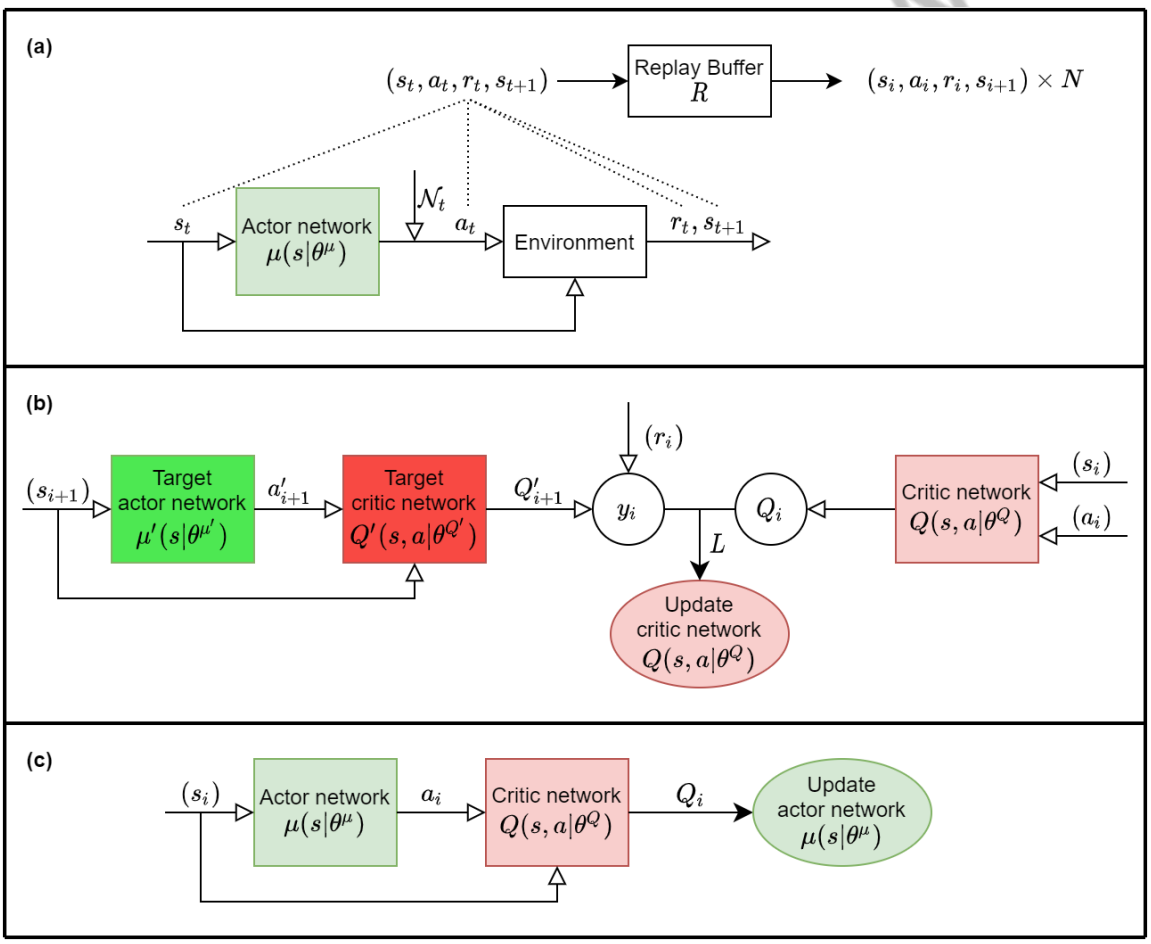


**Figure 1**. Illustration of the DDPG algorithm developed by **Lillicrap et al. (2015)** that is adapted and modified for the proposed reinforcement-learning-based control of fatigue crack growth. (a) The interaction between the RL agent and environment to generate transitions and the use of replay buffer. (b) The procedure for updating the critic network that approximates the expected return. (c) The procedure for updating the actor network that maximizes the expected return. In the subplots (b) and (c), the parameter in the parentheses indicates that it is sampled from the replay buffer.

# 2. Methodology

## 2.1. DDPG reinforcement learning algorithm

In this paper, the DDPG algorithm developed by **Lillicrap et al. (2015)** is implemented for the optimal control task. DDPG is a model-free, off-policy algorithm that combines the ideas from deterministic policy gradient method, deep Q-learning and actor-critic techniques. It aims to maximize the action-value (Q-value) function that evaluates the performance of the control policy. Q-value is the expected return (also known as the expected cumulative discounted rewards) after taking a specific action for a specific state and thereafter following that policy. By implementing deep neural networks as function approximator in both actor and critic models, DDPG algorithm is able to handle control problems with high-dimensional, continuous state and action space, thus suitable for the fatigue crack growth control problem investigated in this paper. The DDPG algorithm is illustrated in **Figure 1** and explained in the following paragraphs. A more in-depth discussion is available in author's open-source documentation archived on ESSOAr website (**Jin and Misra, 2021**).

As shown in **Figure 1(a)**, for each time step $t$ within the same episode, the actor network $\mu(s|\theta^{\mu})$ predicts an action $a_t$ that should be taken given the current state $s_t$, where the parameter $\theta$ represents the weights in the deep neural network and the superscript $\mu$ denotes the actor network. The systematic learning of the optimal control policy includes an update in the weigths of the connections in the actor network $\theta^{\mu}$. The action is chosen by adding an Ornstein-Uhlenbeck noise $\mathcal{N}_t$ **(Uhlenbeck and Ornstein, 1930)** to the prediction. The random noise $\mathcal{N}$ is reinitialized at the beginning of each episode to control the balance between exploration and exploitation. After the RL agent is trained, the noise will be removed during the deployment stage. Given $s_t$ and $a_t$, the RL environment returns the reward value $r_t$ and the next state $s_{t+1}$. The transition $(s_t, a_t, r_t, s_{t+1})$ is then stored in a replay buffer $R$. For each time step, a minibatch of $N$ random transitions $(s_i, a_i, r_i, s_{i+1})$ are sampled from $R$ to update the weights of the connections in the critic and actor network, where the subscript $i$ denotes each transition in the minibatch selected from the replay buffer. The assumption of independently and identically distributed samples in reinforcement learning no longer holds if the samples are generated from exploring sequentially in the environment. The use of replay buffer breaks the temporal correlations between the samples

and make sure that rare experiences are used more than once, which significantly enhances the performance of reinforcement learning.

**Figure 1(b)** shows the procedure for updating the weights of connections $\theta^Q$ in the critic network $Q(s, a|\theta^Q)$, where the superscript $Q$ denotes the critic network. The critic network aims to predict the action-value (Q-value) $Q_i$, which represents the expected value of an action $a_i$ taken at state $s_i$. The critic network is updated by minimizing the mean squared loss $L$, so that its prediction $Q_i$ is close to the moving target $y_i$. The difference between the prediction $Q_i$ and the moving target $y_i$ is also called temporal difference (TD) error and the moving target $y_i$ is also known as TD target. The loss $L$ is defined as

$$L = \frac{1}{N}\sum_{i}^{N}\left(y_i - Q(s_i, a_i|\theta^Q)\right)^2 \quad (1)$$

The moving target $y_i$ is the expected cumulative reward as indicated by the target critic network $Q'\left(s, a|\theta^{Q'}\right)$ calculated as

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right) \quad (2)$$

where $Q'$ denotes the target critic network and $\mu'$ denotes the target actor network. The target critic network predicts the Q-value $Q'_{i+1}$ with respect to the next state $s_{i+1}$ and the action $a'_{i+1}$, which is predicted by the target actor network $\mu'\left(s|\theta^{\mu'}\right)$ given the next state $s_{i+1}$. $\gamma$ is a discount factor for future rewards. The use of target networks enhances the learning stability in the cost of learning speed. If the episode ends at the current state $s_i$, the TD target $y_i$ should instead be calculated as

$$y_i = r_i \quad (3)$$

**Figure 1(c)** shows the procedure for updating the actor network. The actor network represents the current deterministic policy by mapping the state to the specific action deterministically. The actor network is updated using the policy gradient theorem, so that the action it predicts given the current state $s_i$ leads to a high Q-value $Q_i$ as predicted by the critic network. The following equation describes the sampled policy gradient (implies that the gradient is calculated based on the samples drawn from the replay buffer) to update the actor network:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_{i}^{N}\left(\left(\nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i|\theta^\mu)}\right)\left(\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_i}\right)\right) \quad (4)$$

where $J$ denotes the expected return from the start distribution/state. In this paper, the weight updates for critic and actor networks are accomplished in the Tensorflow with Adam optimizer. The initial weights for the target critic/actor networks are the same as the critic/actor networks. For each time step, both the target networks are updated slowly (also referred to as "soft" update) according to rate $\tau$, where $\tau \ll 1$:

$$\theta^{Q'} = \tau\theta^Q + (1-\tau)\theta^{Q'} \quad (5)$$

$$\theta^{\mu'} = \tau\theta^\mu + (1-\tau)\theta^{\mu'} \quad (6)$$

It is observed that Q-learning with neural networks tends to be unstable. The use of soft update ensures stability in learning by making the moving target $y_i$ change slowly. Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state.



### 2.2. Architectures of the actor and critic networks

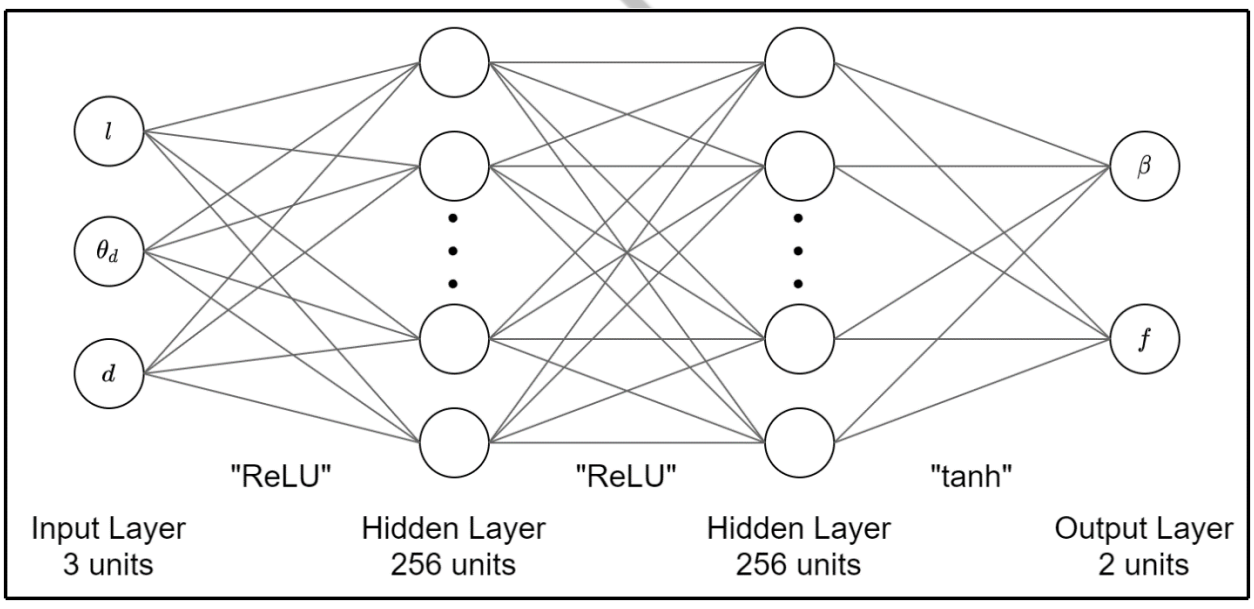


**Figure 2**. Architecture of the actor network that predicts an action $a_t$ that should be taken given the current state $s_t$. Bias units are not shown in this architecture. The inputs to the network are the crack half-length $l$, the direction of the current goal point $\theta_d$, and the distance between crack tip and the current goal point $d$. The network generates the action that includes the stress angle $\beta$ and stress frequency $f$.

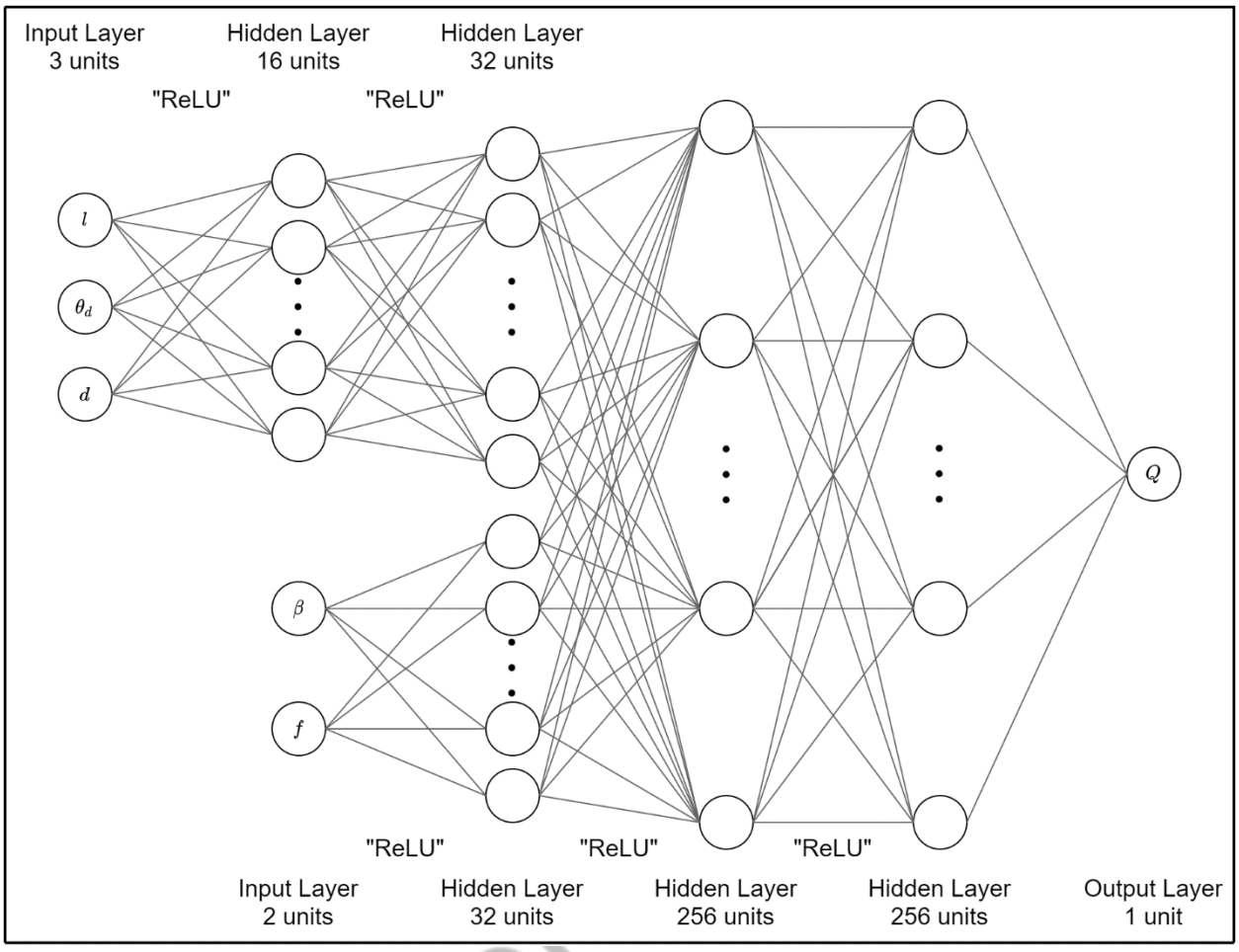


**Figure 3**. Architecture of the critic network that predicts to predict the action-value (Q-value) $Q_i$, which represents the expected return after taking action $a_i$ at state $s_i$, where $i$ represents a transition from the minibatch of $N$ transitions selected from the replay buffer. Bias units are not shown in this architecture. The critic network takes all the actions and states as input.

Four deep neural networks are used as the learning agents in the reinforcement learning framework: actor network $\mu$, target actor network $\mu'$, function approximator for the Q-value function, referred as the critic network $Q$, and target critic network $Q'$. In this paper, the neural networks were built in the Keras platform. **Figure 2** and **Figure 3** depicts the architectures of the actor and critic networks, respectively. The impact of the size of neural networks on the training results will be discussed in the appendix. The target critic/actor networks have the same architecture as the critic/actor networks with the same initial weights. The weights of target networks are updated as shown in equations 5 and 6. The actor network takes state $(l, \theta_d, d)$ as input and then deterministically computes the specific action $(\beta, f)$, while the critic network takes both the state $(l, \theta_d, d)$ and action $(\beta, f)$ as inputs and then computes a scalar Q-value. The "tanh"

activation is used in the output layer in the actor network to bound the action to ensure the action is within a proper range, no activation is used in the output layer in the critic network, whereas all the other layers in both networks use the "ReLU" activation.

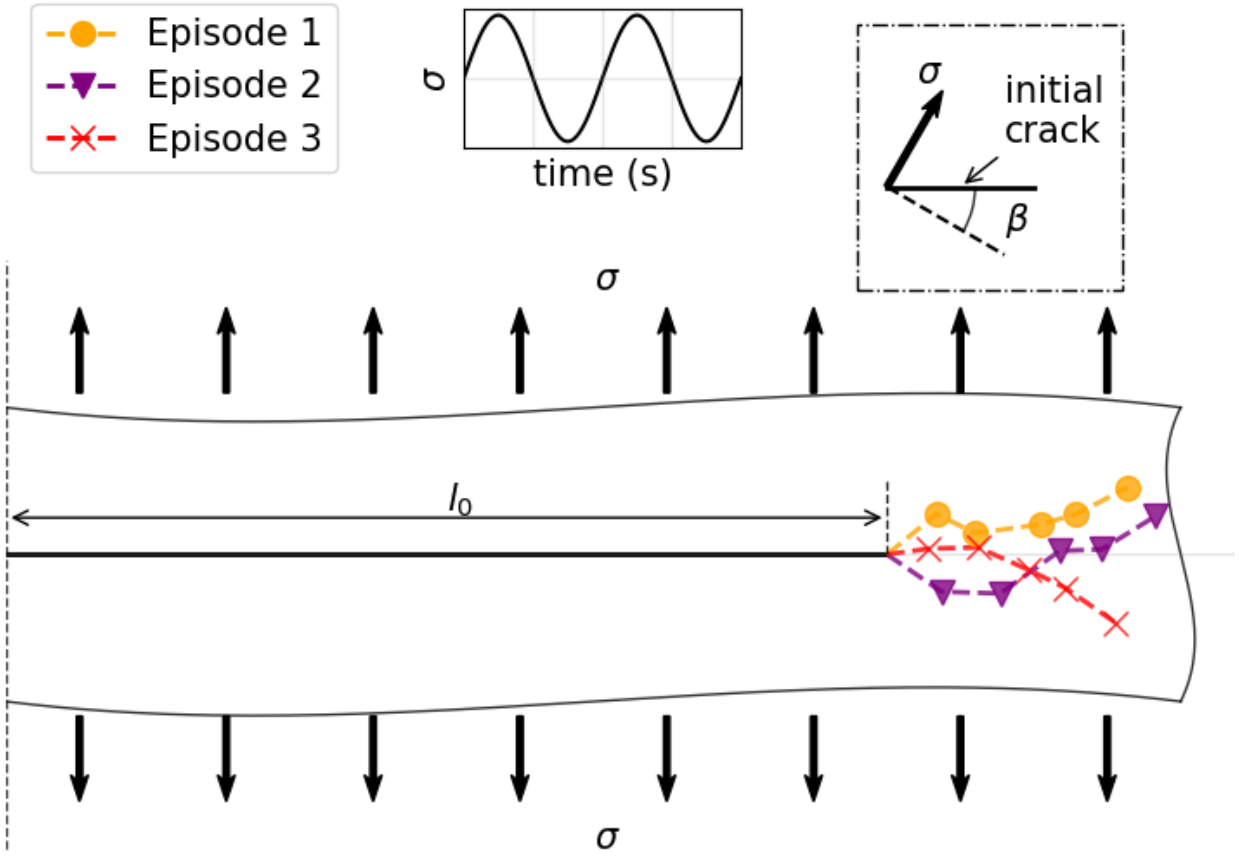


**Figure 4**. Three distinct and independent fatigue crack growth paths in a 2D infinite material for 3 different training episodes. Only half of the material is shown assuming centrosymmetric about the center of the initial crack. The reinforcement learning scheme will train the learning agent to control the mixed-mode fatigue crack growth to track each of the 3 paths during each training episode. The growth of mixed-mode fatigue crack is under the influence of a sinusoidal uniaxial stress field with a maximum magnitude of $\sigma$. The initial crack with length $l_0$ is located at the center. The RL agent needs to learn to control the stress angle $\beta$ and the stress frequency $f$, so that the crack growth on the right half can track the 5 discrete goal points.

### 2.3. Numerical environment used by the learning agents

The training environment was specifically developed for the growth of mixed-mode fatigue crack in an infinite 2D material under sinusoidal biaxial stress field based on the OpenAI Gym environment. In these trainings, the crack growth starts from the middle of an infinite plate containing an initial crack in a sinusoidal biaxial stress field, as shown in **Figure 4**. The initial crack half-length is $l_0$. The maximum magnitude of the sinusoidal stress field is $\sigma$. The angle

between initial crack and the direction that is perpendicular to the stress is $\beta$. The crack environment is centrosymmetric at the center of the initial crack. For the purpose of simplicity, we'll focus on the right half of the crack growth in the following discussion. The maximum stress magnitude $\sigma$ for the sinusoidal stress is kept constant as 100MPa throughout the simulation. The initial locations of the crack tips are $(6.0, 0.0)$ and $(-6.0, 0.0)$. The reinforcement learning agent will learn to control the stress angle $\beta$ and the stress frequency $f$, so that the crack growth can track any 5 pre-defined discrete goal points. For each episode in the training period, first a random desired crack growth path comprising 5 randomly selected goal points is predefined prior to learning. The RL agent needs to learn to track all the 5 goal points in minimum steps (preferably one step for each goal point) in each training episode.

As shown in the crack growth paths for the 3 random training episodes in **Figure 4**, the desired crack paths are centrosymmetric with respect to the location of the initial crack center (i.e. the origin). Initially, as the RL agent performs an action by selecting optimal values of stress angle $\beta$ and the stress frequency $f$ based on current state, the crack propagates toward the first goal point. Once the first goal point is reached within a certain error margin, the crack will propagate toward the second goal point. The learning episode ends when all the goal points are reached with a certain error margin or when the crack propagates towards the wrong direction. The former terminal condition is referred as positive terminal, where the RL agent has successfully completed the task. The latter terminal condition is referred as negative terminal, where the episode is terminated early to avoid wasting time on exploring the less meaningful regions in the state space. The current state of the environment that is accessible to the RL agent is $(l, \theta_d, d)$, where $l$ is the crack half-length that is measured along the crack path, $\theta_d$ is the direction of the current goal point with respect to the horizontal direction, and $d$ is the distance between crack tip and the current goal point.

Each episode of learning focuses on controlling crack growth along a randomly chosen crack path, as predefined by the 5 discrete points for each half as shown in **Figure 4**. Each episode poses a different crack path to the learning agent. The state space $\mathcal{S}$ of $l$ is $[6.0, 8.0]$ mm, that of $\theta_d$ is $[-40°, 40°]$, and that of $d$ is $[0.2, 0.5]$ mm. The reinforcement learning agent will learn to control the crack to propagate along the desired path by manipulating the stress angle $\beta$ and the stress frequency $f$. The action space $\mathcal{A}$ of $\beta$ is $[-30°, 30°]$ and that of $f$ is $[10, 100]$ Hz. The mechanistic equation governing the fatigue crack growth is coupled within the environment. RL

agent interacts with the environment by checking the current state, taking an action, and receiving a reward. All this is designed to mimic a real-world scenario where the learning agent learns by interacting with a solid material containing a single embedded crack.

In this paper, the maximum circumferential stress criterion proposed by **Erdogan and Sih (1963)** is used to determine the direction of fatigue crack growth, where the crack propagates toward a direction which is perpendicular to the direction of greatest tension. Assuming a quasi-static crack growth where the dynamic effects like wave propagation does not affect the crack propagation, the crack propagation angle $\theta_p$ can be represented as **Patricio and Mattheij (2007)**

$$\theta_p = 2\tan^{-1}\left(\frac{K_I - \sqrt{K_I^2 + 8K_{II}^2}}{4K_{II}}\right) \tag{7}$$

where the mode I and II stress intensity factors $K_I$ and $K_{II}$ are calculated as **(Sih et al., 1962)**

$$K_I = \sigma\sqrt{\pi l}\cos^2\beta \tag{8}$$

$$K_{II} = \sigma\sqrt{\pi l}\sin\beta\cos\beta \tag{9}$$

The most commonly used equation to characterize the mixed-mode fatigue crack growth is the Paris–Erdogan equation **(Paris and Erdogan, 1963; Tanaka, 1974)**:

$$\frac{dl}{dN} = C\left(\Delta K_{eq}\right)^m \tag{10}$$

where $N$ represents the number of loading cycles, $m$ is Paris' exponent, $C$ is Paris' constant, $\Delta K_{eq}$ is the stress intensity range. In this paper we assume $m = 3.77$ and $C = 8.77 \times 10^{-12}\,\frac{m/cycle}{(MPa\cdot m^{0.5})^m}$. If the length increment $\Delta l$ for each step is small, it can be estimated as **(Sajith et al., 2019)**

$$\Delta l = C\left(\Delta K_{eq}\right)^m \Delta N = C\left(\Delta K_{eq}\right)^m \Delta t f \tag{11}$$

where $\Delta N$ is the number of loading cycles for each simulation step, $\Delta t$ is the time for each simulation step. We assume $\Delta t = 100s$ during the simulation. By controlling the stress frequency $f$, we control the crack propagation length for each time step; thus, the learning agent controls the rate of crack propagation. Many forms of stress intensity range are available in the literature. In this paper, we use the expression that is proposed by **Irwin (1957)**:

$$\Delta K_{eq} = \sqrt{\Delta K_I^2 + \Delta K_{II}^2} \tag{12}$$

We consider the goal point is reached when the new distance $d_{new}$ between crack tip and the goal point is less than 10% of the original distance $d$. By controlling both the $\beta$ and $f$ such that the crack propagates at various propagation angle $\theta_p$ and length $\Delta l$ at each time step, the RL agent try to reach each of the desired goal points in one single time step.

### 2.4. Tuning parameters to optimize the reinforcement-learning based control

**Table 1** summarizes the tuning parameters used during the training stage. To help the agent better explore the state space, we add an Ornstein-Uhlenbeck noise $\mathcal{N}_t$ **(Uhlenbeck and Ornstein, 1930)** to the output of the action network. Ornstein-Uhlenbeck process can be considered as a random walk process or Brownian motion. The main feature of such process is that it tends to converge to the mean value over time. In the simulation, the Ornstein-Uhlenbeck noise can be represented as

$$\mathcal{N}_t = \mathcal{N}_{t-1} + \vartheta(\mu - \mathcal{N}_{t-1})dt + \sigma dW_t \tag{13}$$

where $\mu$ is the long term mean level, which is 0 in our paper; $\vartheta$ is the speed of reversion, which characterizes the velocity at which the trajectory will regroup around the mean $\mu$; and $\sigma$ is the instantaneous volatility that measures instant by instant the amplitude of randomness. A higher $\sigma$ implies more randomness. $dt$ represents the time interval. $W_t$ denotes the Wiener process. In simulation, $dW_t$ can be modeled as

$$dW_t = Z\sqrt{dt} \tag{14}$$

where $Z$ is an independent standard normal variable. In this paper, we make the $\sigma$ of the noise subject to exponential decay during the training period, such that the RL agent is encouraged to explore at the early training period and exploit at the late training period:

$$\sigma = \sigma_0 e^{-kr_0} \tag{15}$$

where $\sigma_0$ is initial instantaneous volatility; $k$ is the current index of training episode; $r_0$ is decay rate, which is expressed as

$$r_0 = \frac{5}{M} \tag{16}$$

where $M$ is the total number of training episodes. During the deployment stage, the noise will be removed by setting $\sigma$ to 0 for the best control behavior. The actor/critic network learning rate and target networks update rate are chosen to achieve a stable training. The discount factor represents the importance of future rewards. A higher discount factor indicates the future reward is more important. The capacity of replay buffer defines the maximum number of transitions stored. A larger capacity will typically result in a wider range of experience which benefits the stability of training, but it takes longer time to be refreshed with good/new control policy. The minibatch size defines the number of transitions used to update the networks at each step taken by the RL agent.

**Table 1**. Tuning parameters of the deep neural networks during the training stage.

| Parameter | Value |
|---|---|
| Actor network learning rate | 0.00025 |
| Critic network learning rate | 0.005 |
| Target networks update rate $\tau$ | 0.005 |
| Discount factor $\gamma$ | 0.99 |
| Capacity of replay buffer $R$ | 10000 |
| Minibatch size $N$ | 64 |
| Total training episodes $M$ | 30000 |
| Speed of reversion $\vartheta$ | 0.15 |
| Time interval in exploration noise $dt$ | 0.01 |
| Initial instantaneous volatility $\sigma_0$ | 0.2 |



## 2.5. Reward Function

The design of the reward function is the most important factor for a successful and stable reinforcement learning result. Through reward shaping, the RL agent learns in a more informative environment **(Laud, 2004)**. Ideally, the reward function provides all the necessary information for the RL agent to learn the optimal control policy while being as simple as possible. A reward that adapts during the learning steps through reward shaping helps the agent to quickly and correctly converge to the desired policy. Such adaptive rewards are better as compared to binary reward that only marks success or failure **(Ng et al., 1999)**. The impact of the reward function on the training results will be discussed in the appendix. In this paper, we design the reward function as

$$r = \left[1 - \left(\frac{3d_{new}}{d}\right)^{0.2}\right] + \left[1 - \left(\frac{|\theta_d - \theta_p|}{5}\right)^{0.2}\right] + [10n\ (if\ d_{new} < 0.1d)] \qquad (17)$$

where $n$ is the index of current goal point. In the above equation, the first part indicates that the closer the crack tip is to the goal point, the more reward it receives, while the second part indicates that the closer the crack propagates towards the right direction, the more reward it receives. This kind of shaped reward function biases exploration towards the states that are closer to the goal point, which guides the RL agent to converge to a good control policy. The gradient of the reward function becomes higher as it approaches the optimum states (i.e. $d_{new} = 0$ and $\theta_d = \theta_p$), which encourages the RL agent to get as close as possible to the optimum states. We make sure that the reward is positive most of the time so we can use negative terminals, where the episode terminates early as long as the crack propagates to the wrong direction to avoid wasting time on exploring the less meaningful regions in the state space. Positive rewards encourage the RL agent to keep going to accumulate the reward rather than terminate early to avoid receiving high accumulated penalties. However, it also encourage the RL agent to avoid the terminals, unless reaching the terminal state yields high reward. This will be shown in the appendix. To deal with this problem, we have the third part in the reward function, which is an additional reward if the crack tip is close enough to the goal point. In order to encourage the RL agent to reach the goal points as quick as possible, rather than slowly approach the goal points to accumulate the reward, we make this single-step additional reward higher than the discounted expected cumulative reward gained from slowly approaching the goal point. The single-step additional reward becomes higher as the crack approaches the last goal point.

## 3. Results and Discussions

This section shows the RL control of fatigue crack growth direction and rate. In each learning episode, the learning agent will try to maximize the total reward by sequentially reaching the 5 goal points as close as possible within the minimum steps (preferably one step for each goal point). **Figure 5** shows some of the results in training period. As a reminder, each episode poses a different crack path to the learning agent. We can see the learning agent progressively learns to control the crack propagation by comparing episodes 10000, 15000 and 20000. As the number of training episodes increase to 20000, the propagation of crack tip reaches all the 5 goal points. The reward received for each episode during training period is shown in **Figure 6**. The curve is noisy and unstable because of the effect of exploration noise. After training, we remove the exploration noise and perform testing, which is also deployment stage for the trained RL agent. **Figure 7** shows

some of the results in testing period. It can be seen that the agent successfully learnt to control the propagation of crack to reach all the 5 goal points sequentially. The reward received for each episode during testing period without exploration noise is shown in **Figure 8**. Although we gave a relative strict criterion from the training environment to decide if the crack tip is closed enough to the goal point to give the single-step additional reward, the testing period still have a great behavior in terms of controlling. The learning agents exhibit excellent control for all random test cases.

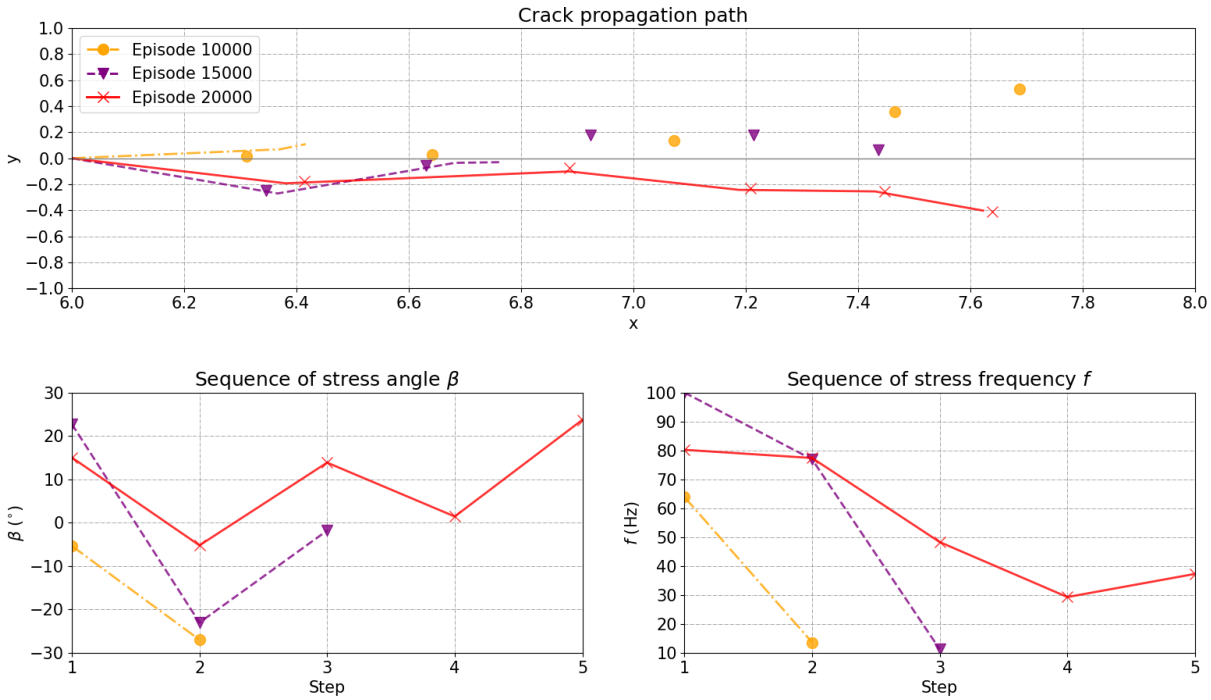


**Figure 5**. The crack propagation paths (top) and corresponding actions (bottom) taken by the RL agent at three training episodes (episodes 10000, 15000 and 20000). The discrete dotted points on top show predefined discrete goal points that the crack propagation need to track. A successful control requires the propagating crack to touch each of the 5 discrete goal points per crack path. RL agent's action includes an optimal selection of stress angle $\beta$ $[-30°, 30°]$ and stress frequency $f$ $[10, 100]$ Hz. The agent progressively learns to control the crack propagation.

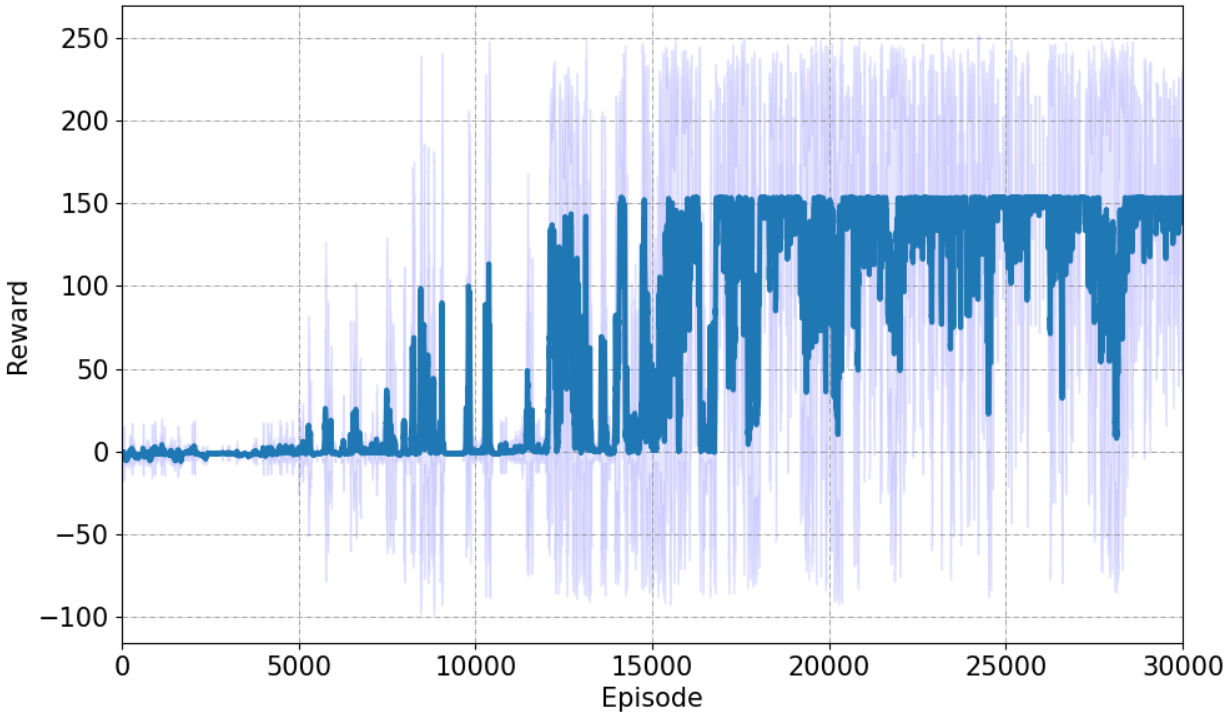


**Figure 6**. Reward history received by the RL agent during training period. The entire training process of 30000 episodes takes 10 minutes on an Intel® Xeon® E5-1650 v3 CPU. The rewards are positive most of the times. The curve is noisy because of the effect of exploration noise.

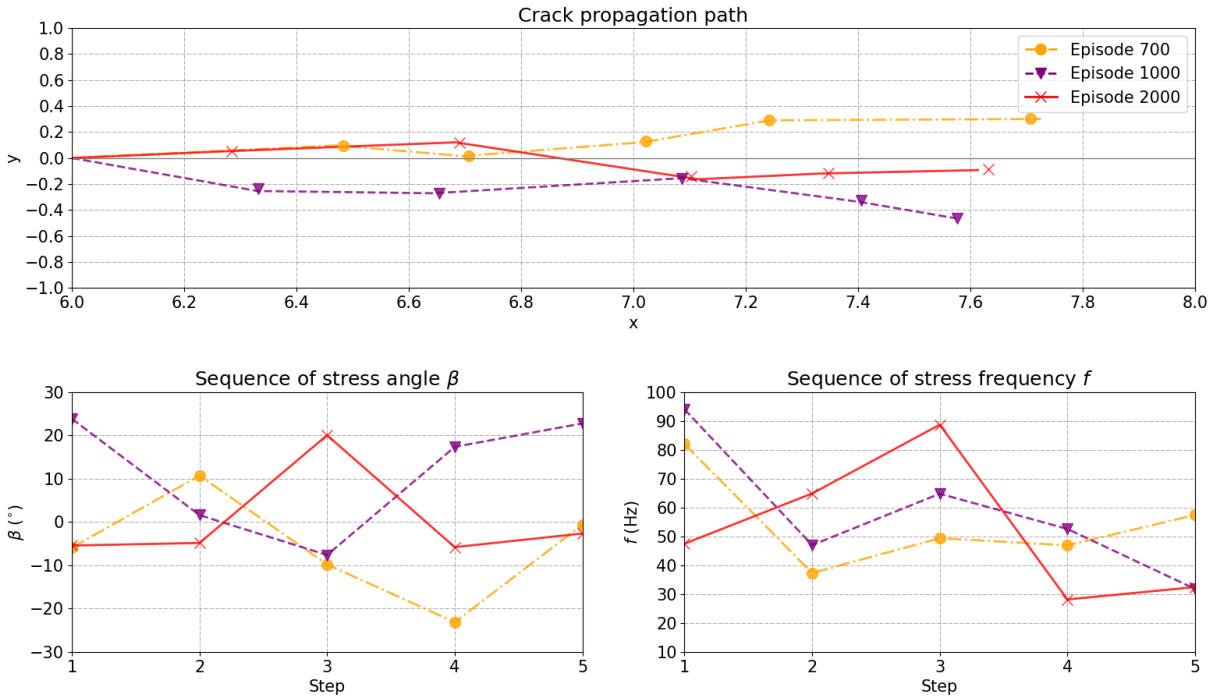


**Figure 7**. The crack propagation paths (top) and corresponding actions (bottom) taken by the RL agent at three testing episodes (episodes 700, 1000 and 2000). The discrete dotted points on top show predefined discrete goal points that the crack propagation need to track. A successful control requires the propagating crack to touch each of the 5 discrete goal points per crack path. RL agent is able to take actions that result in perfect tracking of the predefined crack path.

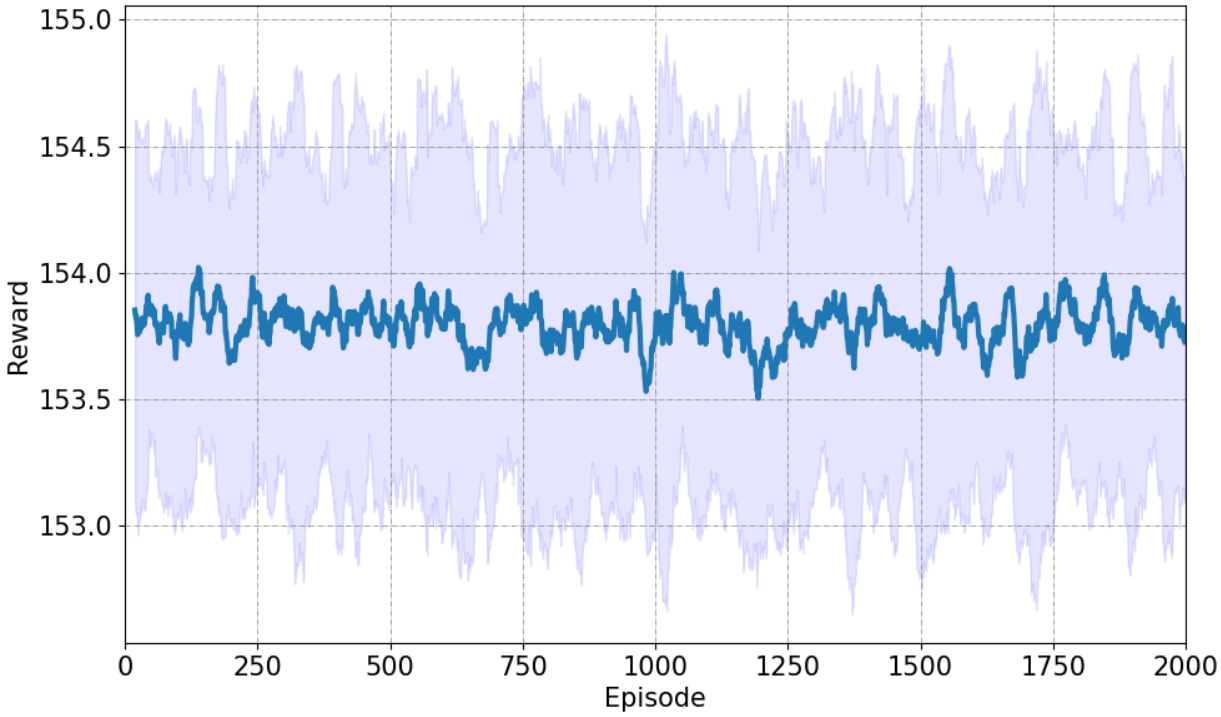


**Figure 8**. Reward history received by the RL agent during testing period. The control was good for all the 2000 testing episodes.

## 3.1. Limitations

The training is performed in a self-created environment by coupling a simple 2D governing equation for fatigue crack growth, which may not be accurate for modelling the crack propagation in complex heterogeneous materials. We assume a high cycle fatigue crack growth in stage II with crack length higher than 10μm, such that the fatigue crack propagation complies with the Paris–Erdogan law. The development of reinforcement learning environment and framework, including the design of reward function and the selection of training parameters, are highly specialized for this particular task. A more generalized environment/framework will be hard to design. The proposed propagation is under sinusoidal uniaxial stress field in an infinite material.

Computationally tractable training of the reinforcement learning on physical systems and on large-scale numerical simulations is not easy. Several training episodes are required for the RL agent to find the optimal policy, and the training parameters need to be carefully tuned to promote a fast and precise training process. Like other machine learning methods, poorly designed reinforcement learning suffers from overfitting. The reinforcement learning models need

continuous interaction with environment as a Markov decision process (MDP), where the next state of the environment depends only on the current state of the environment and the action taken by the agent. The MDP assumption is useful for modeling many real-world sequential decision-making problems. However, the MDP assumption is not necessarily true for all systems and processes. Moreover, according to **Kober et al. (2013)**, reinforcement learning suffers from the following disadvantages:

(1) Curse of dimensionality, wherein the number of interactions with the environment required for training grows exponentially as the dimension of state space increases;

(2) Curse of real-world samples, wherein the physical environment presents several challenges, such as time discretization, delays in sensing and actuation, uncertainty in measurement, disturbance in the environment, inability to observe all states, time, labor and maintenance cost, safety concerns, and external factors, that limit the behavior of reinforcement learning;

(3) Curse of under-modeling and model uncertainty when using a digital simulator as the environment is challenging in light of the difficulty in building a sufficiently accurate model when there exist complex mechanical interactions. The control policy learned from a simulated environment often performs poorly in real world, especially when the system is unstable where small variations can cause drastic divergences;

(4) Curse of goal specification occurs when it is difficult to define a proper reward function and there exists trade-offs between the complexity of the reward function and the complexity of the learning problem.

## 4. Conclusions

The paper is a proof-of-concept development and deployment of a reinforcement learning framework to control both the direction and rate of the growth of fatigue crack embedded in an infinite homogeneous, planar material subject to a sinusoidal uniaxial stress field. To that end, we adapt the deep deterministic policy gradient (DDPG) algorithm and develop robust reward functions. The reinforcement learning scheme learns to control the direction and rate of fatigue

crack growth by interacting with a simulated environment based on the OpenAI Gym environment and adaptively changing two engineering parameters, namely, stress angle and stress frequency. The Markov decision process, which includes state, action and reward, must be carefully designed so that the reinforcement learning framework can learn an optimal, computational tractable, control policy. The state is defined as the crack half-length, the direction of the current goal point, and the distance between crack tip and the current goal point.

The key for robust and accurate control is the design of a good reward function. We designed a reward function to encourage the RL agent to get as close as possible to the optimum states by making the gradient of the reward function become higher as it approaches the optimum states. The reward is positive most of the time to encourage the RL agent to keep going to accumulate the reward rather than terminate early to avoid receiving high accumulated penalties. To prevent the RL agent from avoiding the terminals by slowly moving to the goal point to accumulate the reward, we give the agent an additional high reward if the crack tip is close enough to the goal point to encourage the RL agent to reach the goal points as quick as possible. The RL agent was successfully trained to accomplish the controlled fatigue crack propagation task.

## Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Chemical Sciences, Geosciences, and Biosciences Division under the Award Number DE-SC0020675.

# Nomenclature

## Acronyms

DDPG = deep deterministic policy gradient
MDP = Markov decision process
RL = reinforcement learning
TD = temporal difference

## Symbols

$a$ = action
$\mathcal{A}$ = action space
$\beta$ = stress angle
$C$ = Paris' constant
$d$ = distance between crack tip and the current goal point
$d_{new}$ = new distance between crack tip and the current goal point
$f$ = stress frequency
$\gamma$ = discount factor
$J$ = expected return from the start distribution
$k$ = current index of training episode
$K_I$, $K_{II}$ = mode I, II stress intensity factors
$\Delta K_{eq}$ = equivalent stress intensity range
$l$ = crack half-length
$l_0$ = initial crack half-length
$\Delta l$ = crack propagation length at each time step
$L$ = mean squared loss
$m$ = Paris' exponent
$M$ = total number of training episodes
$\mu$ = long term mean level
$\mu(s|\theta^{\mu})$ = actor network
$\mu'(s|\theta^{\mu'})$ = target actor network
$n$ = index of current goal point
$N$ = minibatch size or number of loading cycles
$\Delta N$ = number of loading cycles for each simulation step
$\mathcal{N}$ = Ornstein-Uhlenbeck noise
$Q$ = action-value (Q-value)
$Q(s, a|\theta^{Q})$ = critic network
$Q'(s, a|\theta^{Q'})$ = target critic network
$r$ = reward
$r_0$ = decay rate
$R$ = replay buffer
$s$ = state
$\mathcal{S}$ = state space
$\sigma$ = stress or instantaneous volatility
$\sigma_0$ = initial instantaneous volatility
$\Delta t$ = the time for each simulation step

$dt$ = time interval
$\tau$ = target networks update rate
$\theta$ = weights in the deep neural network
$\theta_d$ = the direction of the current goal point
$\theta_p$ = crack propagation angle
$\vartheta$ = speed of reversion
$W_t$ = the Wiener process
$y$ = moving target (TD target)
$Z$ = an independent standard normal variable

**Subscripts**

$i$ = denotes each transition selected from the replay buffer
$t$ = denotes time step

**Superscripts**

$'$ = denotes target networks
$\mu$ = denotes actor network
$\mu'$ = denotes target actor network
$Q$ = denotes critic network
$Q'$ = denotes target critic network

# Appendix A: Sensitivity of the Reinforcement Learning Framework

## The effect of neural network size

We re-trained the learning agents with smaller and larger neural networks in the same learning environment. The results are shown in **Figure A1** and **Figure A2**, respectively. Both the testing results are poor. We conclude that similar to supervised and unsupervised learning, there exists a "sweet point" of the neural network size. The size can neither be too large nor too small to obtain a good control policy. Large networks lead to overfitting and small networks lead to underfitting.

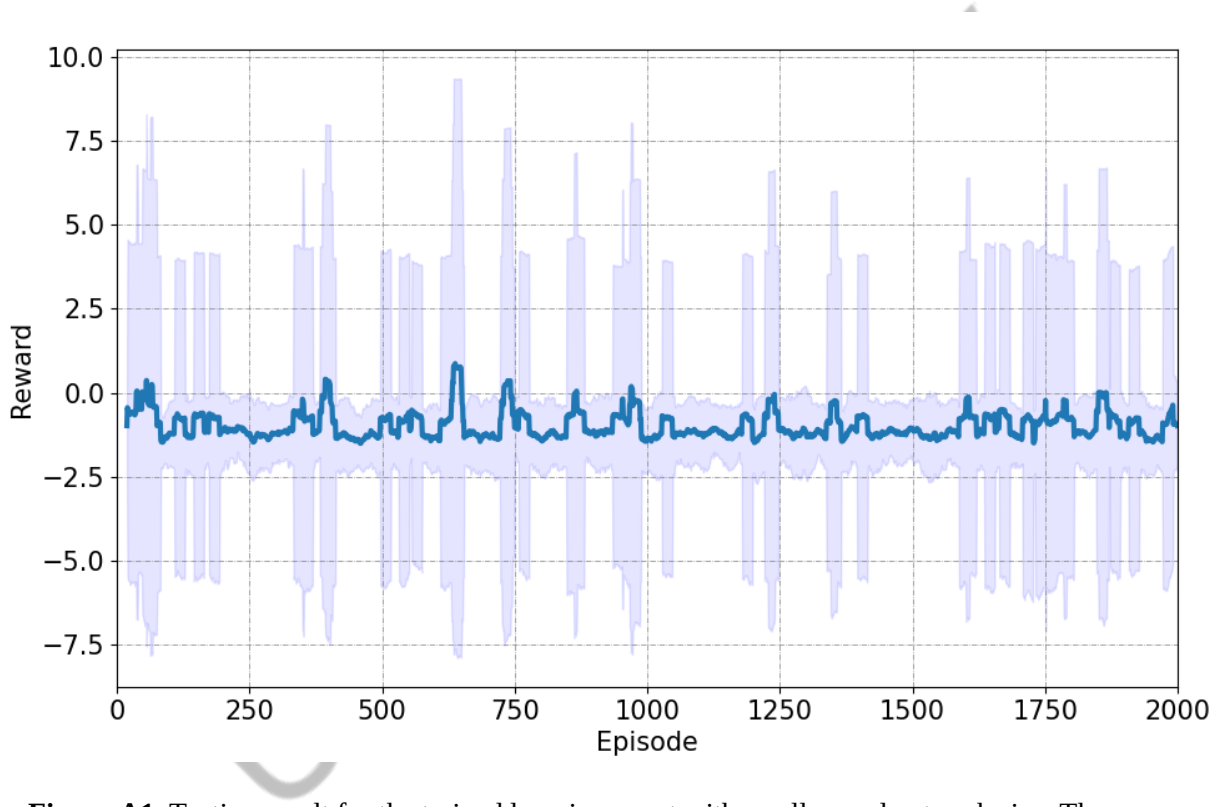


**Figure A1**. Testing result for the trained learning agent with small neural network size. The average reward is below 0, indicating the agent did not learn anything.

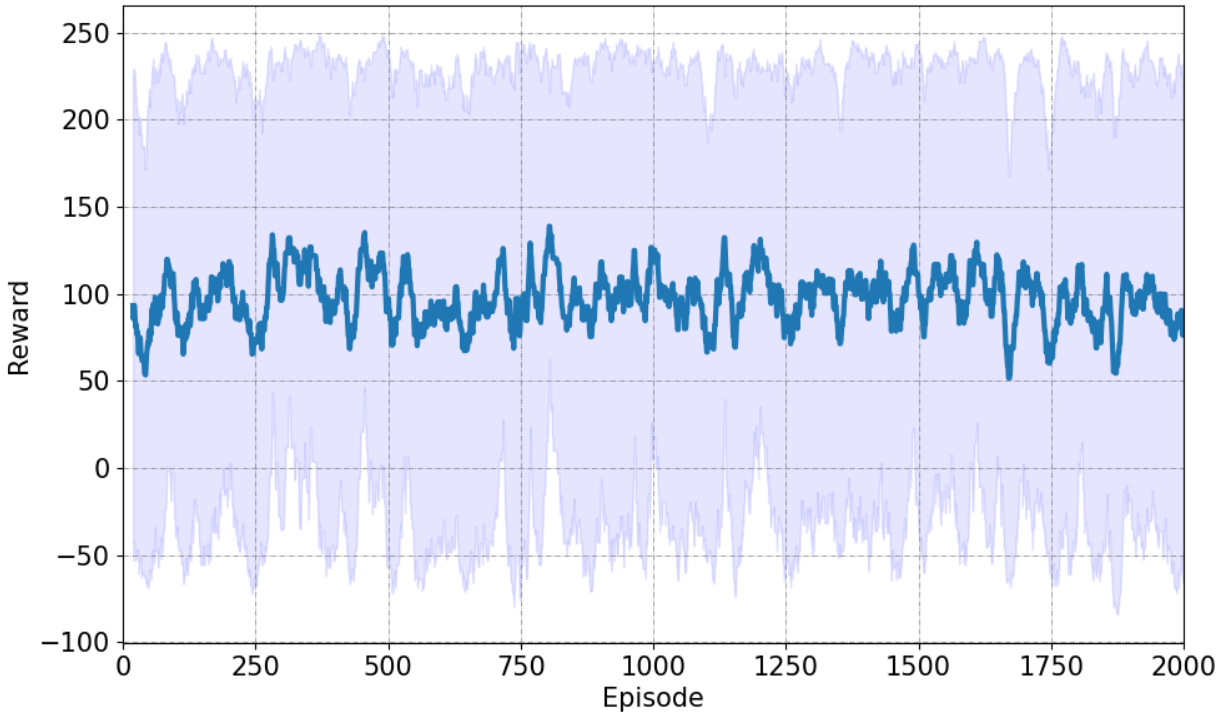


**Figure A2**. Testing result for the trained learning agent with larger neural network size. The curve is unstable with a very large uncertainty range, indicating that the learning agent was unable to learn a good control policy.

**The effect of reward function**

We re-trained the learning agent in a learning environment where the reward function is represented as

$$r = \left[1 - \left(\frac{3d_{new}}{d}\right)^{0.2}\right] + \left[1 - \left(\frac{|\theta_d - \theta_p|}{5}\right)^{0.2}\right] \quad (18)$$

In the above equation, the additional reward term that encourage the learning agent to reach the goal point is removed. Some of the testing results are shown in **Figure A3**. We can see that instead of reaching each goal point within one single action step, the agent learned to approach the goal points slowly to accumulate the rewards. Although the desired propagation path is followed and the goal points are matched well, the agent failed to control the rate of propagation.

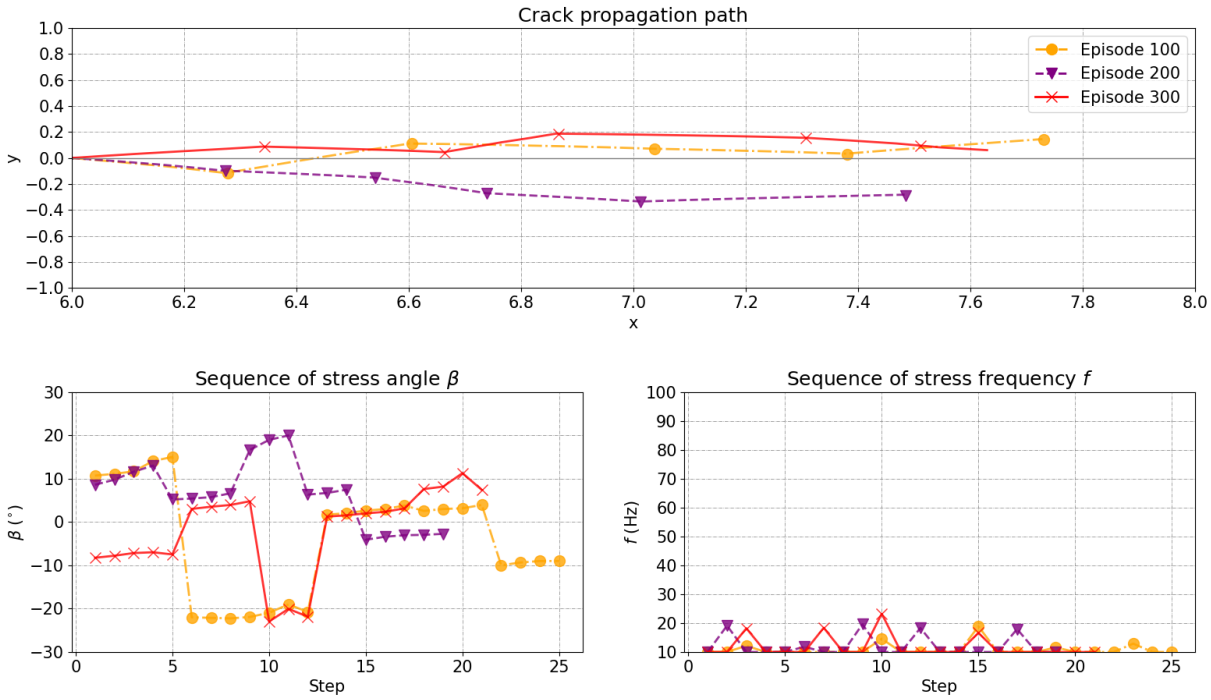


**Figure A3**. Testing results for episodes 100, 200 and 300 for the learning agent trained in a learning environment with a deficient reward function. Although the desired propagation path is followed and the goal points are matched well, the agent failed to control the rate of propagation.

**The effect of state and action space**

We trained the RL agent in an environment with a much larger state and action spaces. The state space of $l$ is $[6.0, 8.2]$ mm, that of $\theta_d$ is $[-60°, 60°]$, and that of $d$ is $[0.1, 0.55]$ mm. The action space of $\beta$ is $[-60°, 60°]$ and that of $f$ is $[10, 1000]$ Hz. This control task is hard because within such an action space, the crack propagation length at each time step $\Delta l$ can be as small as 0.0058 mm and as large as 10.1 mm, depending on the different combination of action parameters. To accommodate the increased state and action spaces, we increase the total number of training episodes to 200000 and the buffer size is increased to 100000. All the other tuning parameters remain the same. The testing result is shown in **Figure A4**. With more training, the learning agent was able to learn a relatively good control policy for this harder control task, but several cases are missed. This harder task required one order more of training episodes. If the state space is further increased to the following: $l$ to $[6.0, 10.8]$ mm, $\theta_d$ to $[-60°, 60°]$, and $d$ to $[0.2, 1.2]$ mm, and the action space is further increased to the following: $\beta$ to $[-60°, 60°]$ and $f$ to $[10, 3000]$ Hz, the

training results are shown in **Figure A5**. Unfortunately, the learning agent was unable to learn a good control policy for this much harder control problem. We can conclude that the reinforcement learning is sensitive to the training environment. It can fail in a poorly-defined training environment with large action and state spaces. For such training environments, we may consider to improve the training by implementing prioritized experience replay technique to honor extreme cases, or by using multi-agent reinforcement learning algorithm.

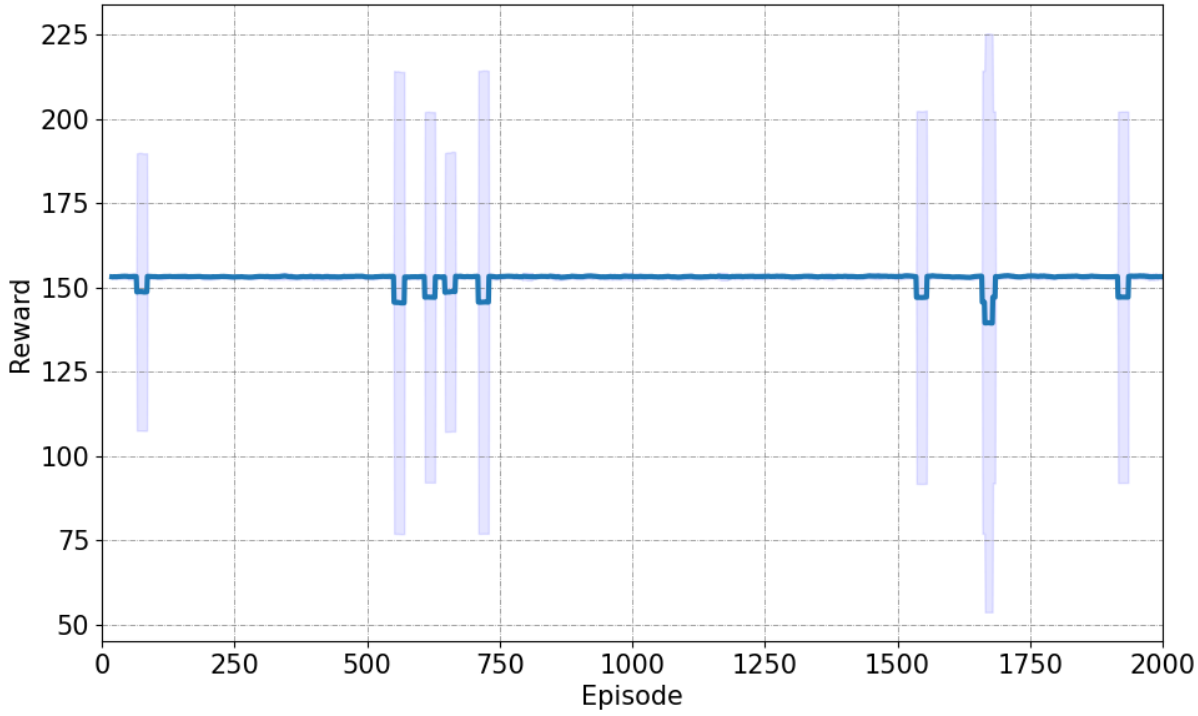


**Figure A4**. Testing result for the learning agent trained in a learning environment with larger state and action spaces. The state space of $l$ is $[6.0, 8.2]$ mm, that of $\theta_d$ is $[-60°, 60°]$, and that of $d$ is $[0.1, 0.55]$ mm. The action space of $\beta$ is $[-60°, 60°]$ and that of $f$ is $[10, 1000]$ Hz. The learning agent was able to learn a relatively good control policy for this harder control task, but several cases are missed.

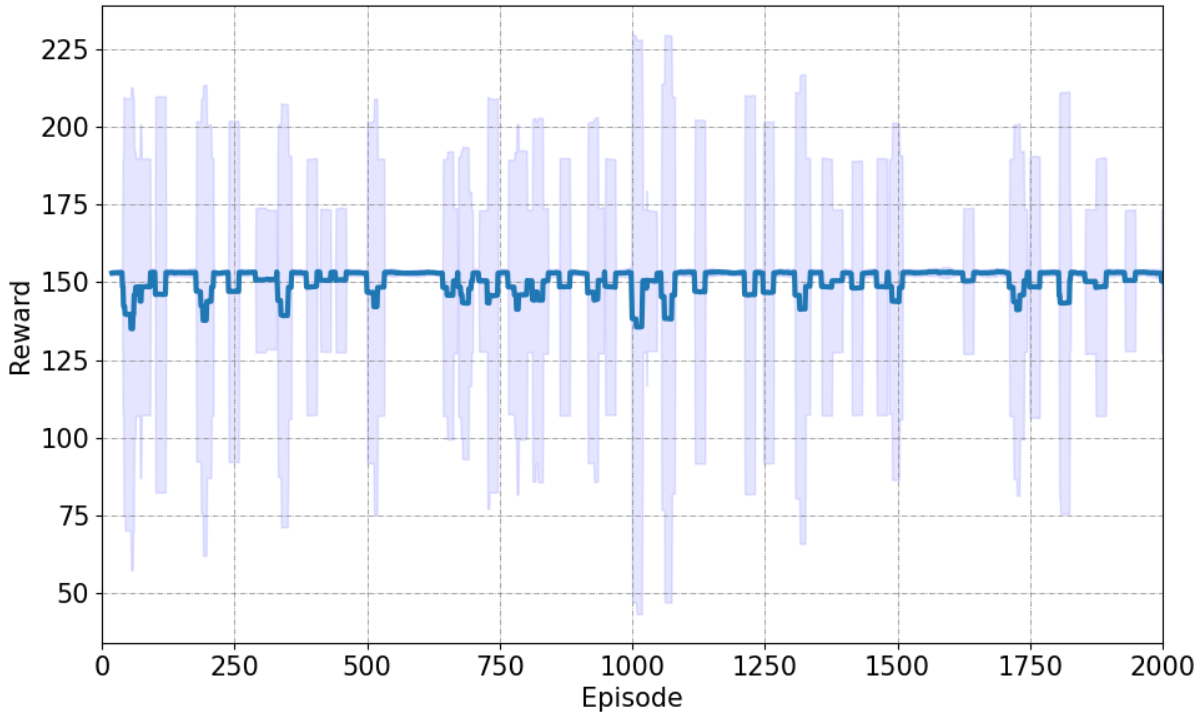


**Figure A5**. Testing result for the learning agent trained in a learning environment with even larger state and action spaces. The state space of $l$ is $[6.0, 10.8]$ mm, that of $\theta_d$ is $[-60°, 60°]$, and that of $d$ is $[0.2, 1.2]$ mm. The action space of $\beta$ is $[-60°, 60°]$ and that of $f$ is $[10, 3000]$ Hz. The learning agent was not able to learn a good control policy for the harder control task with much larger state and actions spaces.

# References


Alegre, Jm, Preciado, M & Ferreño, D 2007. Study of the fatigue failure of an anti-return valve of a high pressure machine. *Engineering Failure Analysis,* 14**,** 408-416.

Ding, Fei, Zhao, Tianwen & Jiang, Yanyao 2007. A study of fatigue crack growth with changing loading direction. *Engineering fracture mechanics,* 74**,** 2014-2029.

Erdogan, Fazil & Sih, Gc 1963. On the crack extension in plates under plane loading and transverse shear. *Journal of Fluids Engineering*.

Irwin, George R 1957. Analysis of stresses and strains near the end of a crack traversing a plate. *Journal of Applied Mechanics*.

Jin, Yuteng & Misra, Siddharth 2021. Controlling the Propagation of Mechanical Discontinuity using Reinforcement Learning. *Earth and Space Science Open Archive*.

Kendall, Alex, Hawke, Jeffrey, Janz, David, Mazur, Przemyslaw, Reda, Daniele, Allen, John-Mark, Lam, Vinh-Dieu, Bewley, Alex & Shah, Amar. Learning to drive in a day. 2019 International Conference on Robotics and Automation (ICRA), 2019. IEEE, 8248-8254.

Laud, Adam Daniel 2004. Theory and application of reward shaping in reinforcement learning.

Le, Jia-Liang, Manning, Jonathan & Labuz, Joseph F 2014. Scaling of fatigue crack growth in rock. *International Journal of Rock Mechanics & Mining Sciences,* 72**,** 71-79.

Li, Hao & Misra, Siddharth 2021. Reinforcement learning based automated history matching for improved hydrocarbon production forecast. *Applied Energy,* 284**,** 116311.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David & Wierstra, Daan 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:.02971*.

Ma, Hongze, Yu, Gaoming, She, Yuehui & Gu, Yongan. Waterflooding Optimization under Geological Uncertainties by Using Deep Reinforcement Learning Algorithms. SPE Annual Technical Conference and Exhibition, 2019. Society of Petroleum Engineers.

Mcbagonluri, F & Soboyejo, W 2005. Mechanical Properties: Fatigue. *Encyclopedia of Condensed Matter Physics.*

Ng, Andrew Y, Harada, Daishi & Russell, Stuart. Policy invariance under reward transformations: Theory and application to reward shaping. Icml, 1999. 278-287.

Paris, Paul & Erdogan, Fazil 1963. A critical analysis of crack propagation laws. *Journal of Fluids Engineering*.

Patricio, Miguel & Mattheij, R 2007. Crack propagation analysis. *CASA report***,** 07-03.

Qian, J & Fatemi, A 1996. Mixed mode fatigue crack growth: a literature survey. *Engineering fracture mechanics,* 55**,** 969-990.

Qiu, Chengrun, Hu, Yang, Chen, Yan & Zeng, Bing 2019. Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. *IEEE Internet of Things Journal,* 6**,** 8577-8588.

Ray, Sonalisa & Kishen, Jm Chandra 2011. Fatigue crack propagation model and size effect in concrete using dimensional analysis. *Mechanics of Materials,* 43**,** 75-86.

Sajith, S, Murthy, Ksr Krishna & Robi, Ps 2019. Prediction of Accurate mixed mode fatigue crack growth curves using the paris' law. *Journal of The Institution of Engineers: Series C,* 100**,** 165-174.

Sander, M & Richard, Ha 2006. Experimental and numerical investigations on the influence of the loading direction on the fatigue crack growth. *International journal of fatigue,* 28**,** 583-591.

Sih, G. C., Paris, P. C. & Erdogan, F. 1962. Crack-Tip, Stress-Intensity Factors for Plane Extension and Plate Bending Problems. *Journal of Applied Mechanics,* 29**,** 306-312.

Tanaka, Keisuke 1974. Fatigue crack propagation from a crack inclined to the cyclic tensile axis. *Engineering Fracture Mechanics,* 6**,** 493-507.

Uhlenbeck, George E & Ornstein, Leonard S 1930. On the theory of the Brownian motion. *Physical review,* 36**,** 823.