Controlling the Propagation of Mechanical Discontinuity using Reinforcement Learning

Yuteng Jin

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. The ability to control mechanical discontinuity is relevant for industries dependent on natural, synthetic and composite materials, e.g. construction, aerospace, oil and gas, ceramics, metal, and geothermal industries, to name a few. The paper is a proof-of-concept development and deployment of a reinforcement learning framework to control the propagation of mechanical discontinuity. The reinforcement learning framework is coupled with an OpenAI-Gym-based environment that uses the mechanistic equation governing the propagation of mechanical discontinuity. Learning agent does not explicitly know about the underlying physics of propagation of discontinuity; nonetheless, the learning agent can infer the control strategy by continuously interacting the simulation environment. The Markov decision process, which includes state, action and reward, had to be carefully designed to obtain a good control policy. The deep deterministic policy gradient (DDPG) algorithm is implemented for learning continuous actions for the desired reinforcement learning. It is also observed that the training efficiency is strongly determined by the formulation of reward function. An adaptive reward function involving reward shaping improves the training. The reward function that forces the learning agent to stay on the shortest linear path between crack tip and goal point performs much better than the reward function that aims to reach closest to the goal point in minimum number of steps. After close to 500 training episodes, the reinforcement learning framework successfully controlled the propagation of discontinuity in a material despite the complexity of the propagation pathway determined by multiple goal points.

Keywords: Reinforcement learning; policy gradient; discontinuity; propagation; control

Introduction

Controlling Mechanical Discontinuity in Materials

Consider a homogeneous and linear-elastic solid material containing a mechanical discontinuity (also referred as crack) of a specific size and geometry. When such a solid material is subject to an increasing external force, the mechanical discontinuity will propagate inside the solid material. The mechanical behavior of the discontinuity embedded in solid material is characterized by the stress intensity factors K_I, K_{II} and K_{III}. The three stress intensity factors

correspond to three crack opening modes. Mode I describes an opening mode due to the tensile stress which is normal to the crack plane. Mode II describes a sliding mode due to the shear stress, which is parallel to the crack plane and perpendicular to the crack front. Mode III describes a tearing mode due to the shear stress, which is parallel to the crack plane and parallel to the crack front **(Perez, 2017)**. According to the principle of maximum circumferential stress proposed by **Erdogan and Sih (1963)**, the crack growth will occur at the crack tip once K_I reaches the mode I critical stress intensity factor/fracture toughness K_{IC}, and the crack propagates toward a direction which is perpendicular to the direction of greatest tension.

The ability to control mechanical discontinuity enhances the human understanding and manipulation of the bulk mechanical, chemical and physical properties. For example, a control of the subsurface fractures improve the production of natural gas and petroleum, the development of engineered geothermal systems, and the long-term storage of CO_2 **(Pyrak-Nolte et al., 2015)**. The control of propagation of mechanical discontinuity is an active research topic in various disciplines dependent on natural, synthetic and composite materials, e.g. construction, aerospace, oil and gas, ceramics, metal, and geothermal industries, to name a few.

Chen et al. (2019) found the fusion-bonded dots of the veil can be used to control the crack propagation of veil interleaved composite. **Haverkamp et al. (2021)** controlled the adhesion crack movement in a jamming-based switchable adhesive by dynamically controlling the rigidity through an integrated jamming layer. **Sugita et al. (2009)** designed a cutting method for bone to control the crack propagation and improve the surface roughness. **Namazu et al. (2011)** proposed multiple ignition method to control the crack location and propagation direction in the Al/Ni exothermic reaction during solder-bonding.

Several attempts have been made to control the propagation of mechanical discontinuity within a solid body. **Xu et al. (2009)** made cracks propagate along the desired trajectories by controlling the direction of an applied external point force. The crack propagation path was verified using extended finite element method (XFEM). **Cheng and Wang (2018)** proposed a controllable crack propagation strategy using back propagation neural network assisted particle swarm optimization method with an efficient re-analysis based XFEM solver. The crack propagation was controlled by arranging holes in the design domain. **Cheng et al. (2018)** performed experimental and numerical analysis on the crack propagation control in directional hydraulic fracturing

technique. The influence of horizontal stress difference coefficient and hydraulic slotting deviation angle on the crack propagation was investigated. Their result showed that the direction of crack propagation can be redirected to the directional hydraulic fracturing zone formed by creating internal slotting in the samples.

Reinforcement learning and its application

Reinforcement learning (RL) is a machine learning technique suited for sequential decision making and control problems. In reinforcement learning, the learning agents continuously interact with an external environment by taking actions, observing the state of the environment, and receiving rewards based on the action and state, such that the agents learn from experience through trial and error, i.e. exploration and exploitation. The goal of reinforcement learning is to enable an agent to learn a control policy wherein optimal sequence of actions is carried out under uncertainty, aiming to maximize the expected cumulative reward.

Unlike the supervised and unsupervised learning, reinforcement learning learns from dynamic interaction (i.e. experience) with an environment without assuming independent and identically distributed samples rather than processing a static training dataset. While the supervised and unsupervised learning learn through a cost function, reinforcement learning take use of the feedback from the environment **(Li and Misra, 2021)**. Reinforcement learning can solve complex problems, which is hard to be solved by traditional supervised learning methods. During the training stage, the RL agents learn to correct errors in the control strategy and improve the learning through the feedback obtained from the environment. Once the error is corrected, it is unlikely for the RL agents to make the same error, so the RL agents can progressively learn to better control the system. While continuously interacting with the environment, the RL agents aim to learn the optimal/ideal control policy for a particular problem by focusing on the long-term reward and balancing exploration and exploitation, whereas the supervised and unsupervised learning aim to learn certain representations by exploiting the existing dataset **(Li and Misra, 2021)**. In this regard, the reinforcement learning process is closer to the learning behavior of animals. In many tasks like the Atari games and the game of Go, reinforcement learning has outperformed human experts.

Computationally tractable training of the reinforcement learning 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 **(Zhang et al., 2018)**. 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. 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.

There are several model-free reinforcement learning (RL) based on the approach of learning the optimal control policy. On-policy learning algorithms enable the RL agents to learn the value of a policy based on the policy itself. Every visit Monte Carlo algorithm and state–action–reward–state–action (SARSA) algorithm are examples of on-policy learning algorithms suitable for discrete state and action space. Trust region policy optimization (TRPO) algorithm **(Schulman et al., 2015)**, asynchronous advantage actor-critic (A3C) algorithm **(Mnih et al., 2016)**, and proximal policy optimization (PPO) algorithm **(Schulman et al., 2017)** are on-policy learning algorithms suitable for continuous state and action space. As for off-policy learning algorithms, where the RL agents evaluate a policy using the experience gathered by following different policies, Q-learning algorithm was developed for discrete state and action space. By replacing the Q table in Q-learning

with deep neural network as function approximator, the deep Q network (DQN) algorithm can deal with continuous state space but the action space remains discrete **(Mnih et al., 2015)**. By combining the deterministic policy gradient method and actor-critic approach, deep deterministic policy gradient (DDPG) algorithm is suitable for problems with continuous state and action space **(Lillicrap et al., 2015)**.

Reinforcement learning is widely applied in the control problems like self-driving **(Kendall et al., 2019)**, trajectory tracking **(Choi et al., 2017)**, robot controlling **(Kumar and Sharma, 2018)**, object detection **(Thornton et al., 2020)**, active flow controlling **(Rabault et al., 2018; Ren et al., 2020)**, and aerospace engineering **(Peng and Ma, 2021)**. Some of the applications are in the oil & gas industry. A predictive control model with reinforcement learning was developed by **Talavera et al. (2010)** to control the oil production from a petroleum reservoir. Their model was capable of controlling oil production even with disturbances in the producing well. **Wu et al. (2018)** developed a comprehensive control system where they used a DQN-based reinforcement learning algorithm to jointly control the valve opening, furnace temperature, and pump pressure to avoid wax formation in the crude oil gathering pipe. Recently, reinforcement learning is being applied in the petroleum engineering domain, especially on reservoir optimization and production forecasting problems. **Hourfar et al. (2019)** introduced an approach based on reinforcement learning for managing and optimizing the waterflooding process in the oil reservoirs. **Ma et al. (2019)** applied several reinforcement learning algorithms on the waterflooding optimization problem to find the optimal water injection rate under geological uncertainties. Their work was extended by **Miftakhov et al. (2020),** who trained a reinforcement learning agent that is capable of controlling the injection rate to maximize the net present value of waterflooding by using only pixel data. Meanwhile, **Guevara et al. (2018)** implemented the SARSA reinforcement learning algorithm to optimize steam injection in heavy oil reservoirs. **Li and Misra (2021)** developed a reinforcement-learning-based automated history matching technique based on the DDPG algorithm to improve the forecast of hydrocarbon production.

Novelty and significance of the work

The paper is a proof-of-concept development and deployment of a reinforcement learning framework capable of controlling the propagation of mechanical discontinuity. There does not exist any prior work on using reinforcement learning or other alternatives for robust control of

mechanical discontinuity. To ensure computational tractability of this proof-of-concept development, we assumed a 2D material containing one slow-propagating piecewise linear mechanical discontinuity. In addition, the learning agent trains by interacting with a simulation environment, which implements one mechanistic equation that models the propagation of the discontinuity in a 2D material under biaxial stress. Learning agent does not explicitly know about the underlying physics of propagation of discontinuity; nonetheless, the learning agent can infer the control strategy by continuously interacting the numerical simulation within a training environment, based on the OpenAI Gym environment.

The following fundamental questions are investigated and answered in this paper:

- Can reinforcement learning be used to control the propagation of mechanical discontinuity?
- Does the deep deterministic policy gradient (DDPG) algorithm ensure a robust reinforcement-learning-based control of discontinuity?
- What should be the parameters (state, action, reward) of the Markov decision process (MDP) for the reinforcement-learning-based control of discontinuity? How can the MDP be properly defined using a numerical simulator?
- How does the design of reward function affect the training outcome and efficiency of the reinforcement-learning-based control of discontinuity?

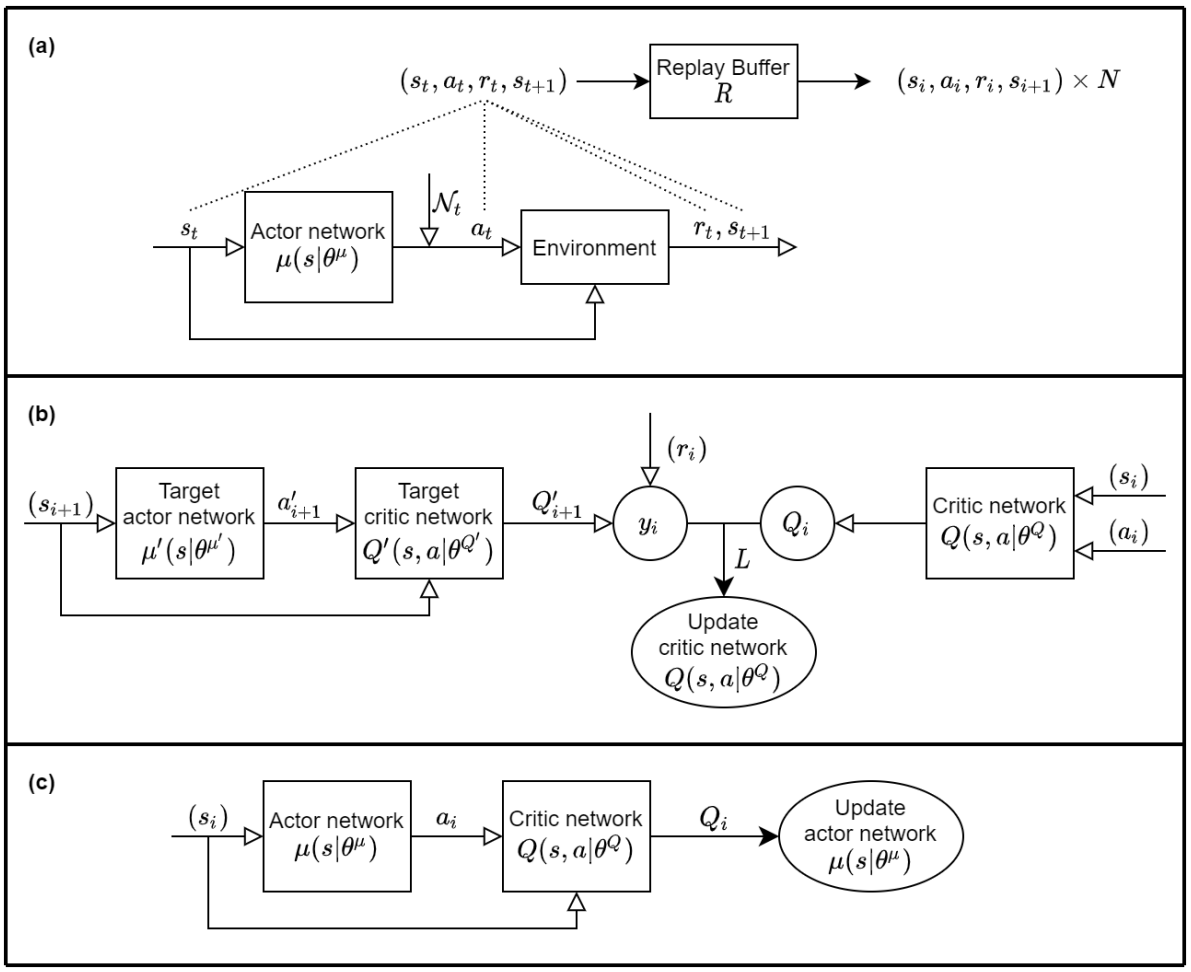

Figure 1. Illustration of the DDPG algorithm developed by **Lillicrap et al. (2015)** that is adapted and modified for the proposed reinforcement-learning-based control of mechanical discontinuity. (a) The interaction between the RL agent and environment and the use of replay buffer. (b) The procedure for updating the critic network. (c) The procedure for updating the actor network. In the subplots (b) and (c), the parameter in the bracket indicates that it is sampled from the replay buffer.

Methodology

Reinforcement learning algorithm

In this paper, we use the DDPG algorithm, which was developed by **Lillicrap et al. (2015)**. DDPG algorithm uses deep neural network as function approximator to estimate action-value (Q-value) function. Q-value evaluates the goodness of the control policy. Q-value is the expected reward after taking a specific action for a specific state and thereafter following that policy. DDPG is a model-free, off-policy actor-critic algorithm based on the deterministic policy gradient (DPG) method. It can learn policies in high-dimensional, continuous action spaces and can solve problems with high-dimensional, continuous observation spaces. This makes it an ideal reinforcement

learning algorithm for the crack-propagation control problem investigated in this paper. The DDPG algorithm is illustrated in **Figure 1** and explained in the following paragraphs. Expected return is also known as expected cumulative discounted future rewards.

As shown in **Figure 1(a)**, within the same episode, for each time step t, 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 θ represents the weights in the deep neural network and the superscript μ 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 θ^{μ}. The action is selected 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=20 transitions (s_i, a_i, r_i, s_{i+1}) are sampled from R to update the weights of 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 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

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

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

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

where Q' denotes the target critic network and μ' 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 $\mu'(s|\theta^{\mu'})$ given the next state s_{i+1}. γ is a discount factor for future rewards. The use of target networks enhances the learning stability in the cost of learning speed.

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:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_{i}^{N} \left(\left(\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i|\theta^\mu)} \right) \left(\nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i} \right) \right) \quad (3)$$

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 τ, where $\tau \ll 1$:

$$\theta^{Q'} = \tau\theta^Q + (1 - \tau)\theta^{Q'} \quad (4)$$

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

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.

Architectures of the actor and critic networks

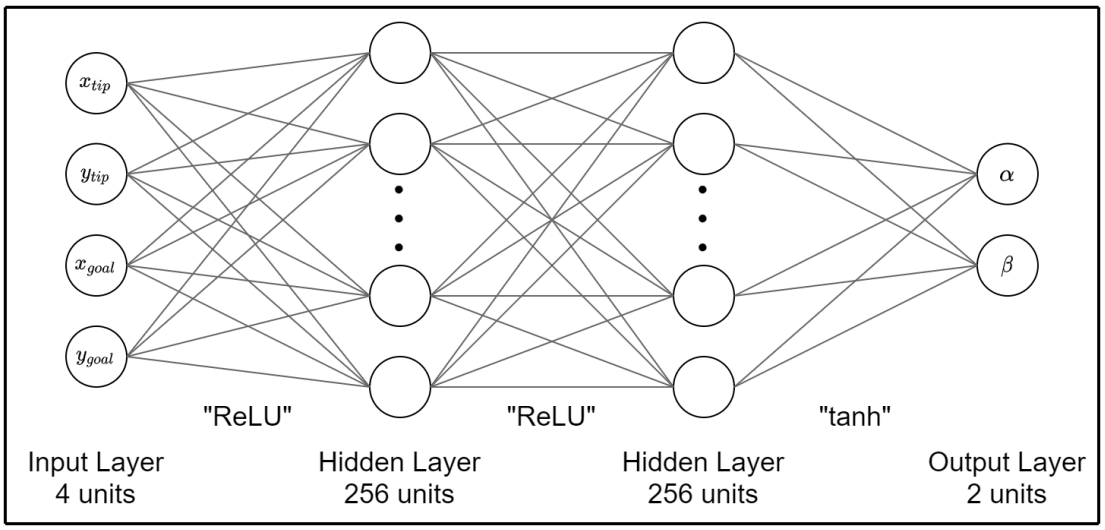

Figure 2. Architecture of the actor network that predicts an action a_t that should be taken given the current state s_t. Bias units are not shown in this architecture. The inputs to the network are the x and y coordinates of the goal point and the x and y coordinates of the current tip of the propagating discontinuity. The network generates the action that includes the stress ratio α and the stress angle β.

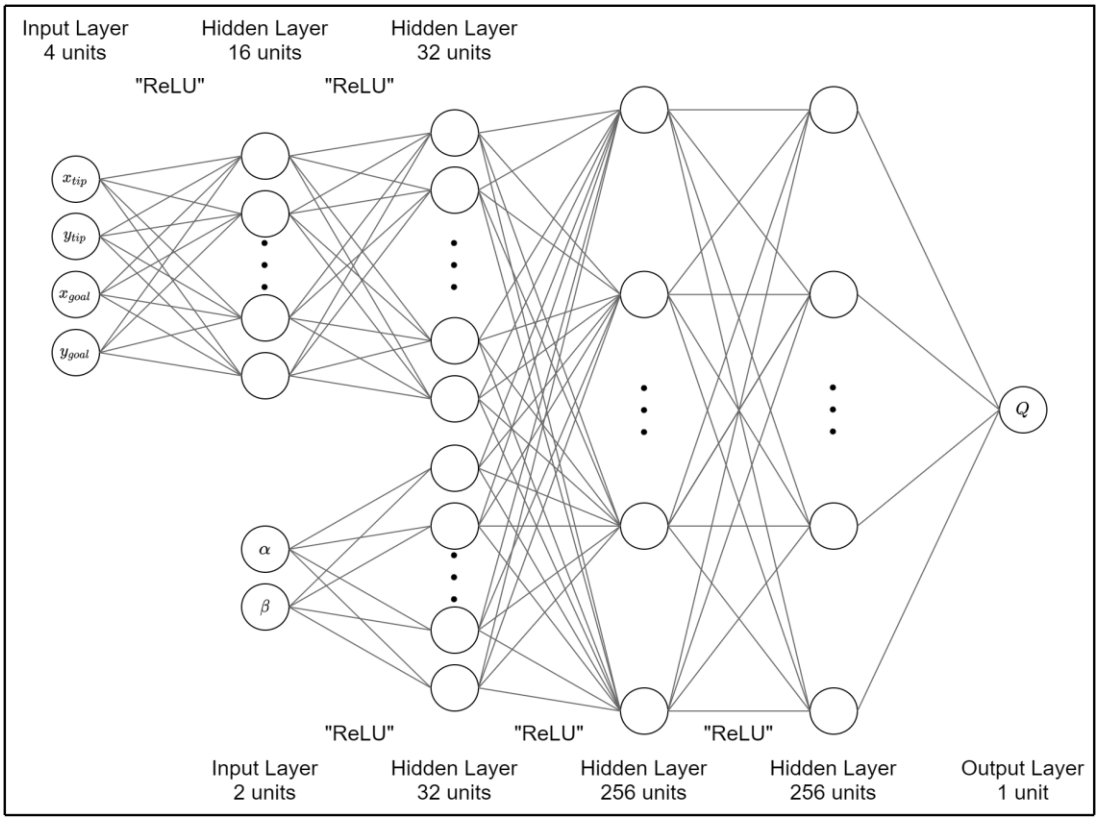

Figure 3. Architecture of the critic network that predicts to predict the action-value (Q-value) Q_i, which represents the expected return after taking action a_i at state s_i, where i represents a transition from the minibatch of N transitions selected from the replay buffer. Bias units are not shown in this architecture. Four of the six inputs of the critic network are similar to the actor network, the remaining two of the six inputs are the stress ratio α and the stress angle β.

Four deep neural networks are used as the learning agents in the reinforcement learning framework: actor network μ, target actor network μ', 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 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 4 and 5. The actor network takes state (x_{tip}, y_{tip}, x_{goal}, y_{goal}) as input and then deterministically computes the specific action (α, β), while the critic network takes both the state and action (α, β) 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.

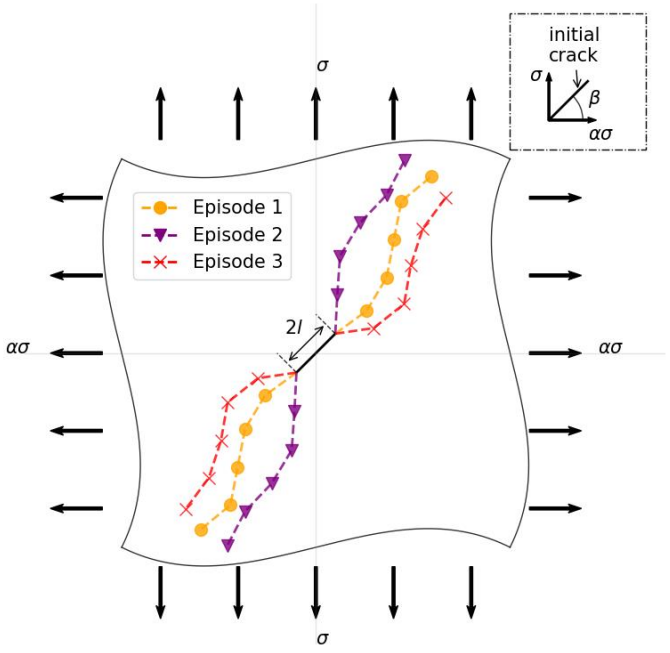

Figure 4. Three distinct and independent propagation paths in a 2D infinite material for the 3 different training episodes. The reinforcement learning scheme will train the learning agents to control the propagation of mechanical discontinuity to track each of the 3 paths during each training episode. The propagation of discontinuity is under the influence of step-wise varying biaxial stress field in the 2D infinite material. The propagation of discontinuity starts from the center. The reinforcement learning agent needs to learn to control the stress ratio α and the stress angle β, so that the discontinuity can track the 10 discrete goal points, which a symmetrical across the center of the initial location of discontinuity.

Simulation environment used by the learning agents

The training environment was specifically developed for the propagation of discontinuity in an infinite 2D material under biaxial stress field based on the OpenAI Gym environment. In these trainings, the crack propagation starts from the middle of an infinite plate containing a small slanted initial crack in a biaxial stress field, as shown in **Figure 4**. The initial crack half-length is l. The two perpendicular stresses are denoted as σ and $\alpha\sigma$, respectively. The angle between initial crack and the direction of stress $\alpha\sigma$ is β. The crack environment is centrosymmetric at (0, 0). The stress σ is kept constant as 100MPa throughout the simulation, while the stress $\alpha\sigma$ varies by controlling the ratio α. We assume the crack will propagate under this stress magnitude. The initial locations of the crack tips are (0.1, 0.1) and (−0.1, −0.1). Reinforcement learning agent will need to learn to control the stress ratio α and the stress angle β, so that the propagating crack can track any 10 pre-defined discrete goal points. For each episode of training the RL agent, first a random desired crack-propagation path comprising 10 goal points is predefined prior to learning. The crack-propagation path comprises 10 randomly selected goal points. The RL agent needs to learn to track the 10 goal points within one episode containing at most 25 training steps.

As shown in the propagation paths for the 3 random training episodes in **Figure 4**, the desired crack path are symmetric 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 ratio α and stress angle β 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 a total of 25 training steps are complete (for reward function 1) or when the crack propagates in the opposite direction of the current goal point (for reward function 2). The current state of the environment that is accessible to the RL agent is the location of propagating crack tip and the location of the current goal point along the desired path. Each episode of learning focuses on controlling crack propagation along a randomly chosen crack path, as predefined by the 10 discrete points shown in **Figure 4**. Each episode poses a different crack path to the learning agent. We observe that the RL agent learns to perfectly control the crack propagation in less than 500 learning episodes. This will be shown in subsequent figures.

The state space $\mathcal{S} = \mathbb{R}^4$ because the plate is assumed to be infinite. The reinforcement learning agent will learn to control the crack to propagate along the desired path by manipulating the stress ratio α and the stress angle β. The action space $\mathcal{A}$ of α is [−2, 2] and that of β is [−90°, 90°]. The mechanistic equation governing the crack propagation 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.

Assuming a quasi-static crack growth where the dynamic effects like wave propagation does not affect the crack propagation, the crack propagation angle θ_p is calculated as **(Patricio and Mattheij, 2007)**

$$\theta_p = 2\tan^{-1}\left(\frac{K_I - \sqrt{K_I^2 + 8K_{II}^2}}{4K_{II}}\right) \tag{6}$$

where the mode I and II stress intensity factors are expressed as **(Sih et al., 1962)**

$$K_I = \sigma\sqrt{\pi a}(\cos^2\beta + \alpha\sin^2\beta) \tag{7}$$

$$K_{II} = \sigma\sqrt{\pi a}(1-\alpha)\sin\beta\cos\beta \tag{8}$$

For each time step, the crack propagates by a fixed length Δl = 0.0667. This is a big assumption that needs to be addressed in the future work. We consider the goal point is reached when the distance between crack tip and the goal point is less than Δl. The RL scheme needs to control the α and β at each time step such that the discontinuity propagates at various propagation angle θ_p slowly moving closer to the desired goal point.

Tuning parameters to optimize the reinforcement-learning based control

Table 1 summarizes the tuning parameters used during the training stage. The exploration noise is not a tuning parameter. The RL agent can learn to successfully accomplish the task without the help of exploration noise, as shown in the results section. The use of exploration noise will take extra time and computational resource; therefore, it is not used in this paper. 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 define 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	0.00005
Critic network learning rate	0.001
Target networks update rate τ	0.005
Discount factor γ	0.5
Capacity of replay buffer R	10000
Minibatch size N	64
Total training episodes	2000

Reward Functions

The design of the reward function determines the effectiveness of training. 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 process of reward shaping is equivalent to learning in a more informative environment **(Laud, 2004)**. In this paper, we demonstrate two ways of constructing the reward function that are provided to the learning agents based on the new state and the current action and state. We will also discuss the impact of the reward function on the training results. Both reward functions are designed such that the RL agent needs to learn the optimal policy that maximizes the total reward (as less negative as possible). In our formulation, small negative reward indicates robust control, whereas large negative reward represents poor control.

Reward function 1 is formulated as follows: For each step, the RL agent receives a reward of -10. The more steps the agent takes to reach a goal point, the more negative reward the agent earns, which is not optimal for the desired control. When the current goal point is reached within a certain error margin, the RL agent will receive an additional reward of $30-200d_1$, where d_1 is the distance between crack tip and the current goal point. This ensures a larger reward when the goal point is reached as close as possible in limited number of steps. Compared to the reward

function 2, reward function 1 tries to minimize the number of steps to reach a goal point as close as possible. Reward function provides only the information of the location of the current goal point with respect to crack tip at the end of the training episode.

Reward function 2 is formulated as follows: For each step taken, the reward is $-1000d_2$, where d_2 is the distance between crack tip and the current-optimal path, which is the straight line connecting the crack tip before taking the step and the current goal point. Reward function 2 penalizes the crack for deviating from the desired path by forcing the agent to stay on the shortest linear path at each step in the training episode. Compared to the reward function 1, the reward function 2 provides the information of the current-optimal path and the location of the goal point at each step of training.

Results and Discussions

Reinforcement-learning based control of discontinuity using reward function 1

This section shows the RL control of discontinuity using reward function 1. Each learning episode ends after 25 steps, or when all the 10 goal points are reached in less than 25 steps. During each episode, the learning agent can take at the most 25 sequential actions that lead to the tracking of the predefined crack path. **Figure 5** shows the early-stage training results. As a reminder, each episode poses a different crack path to the learning agent. At episode 50, the crack simply propagated without honoring the desired path. The learning agent did not know how to adjust the actions, accordingly, as demonstrated by the lack of variance in the action sequence. At episode 150, it learnt to follow the goal points but still missed the extreme goals point. At episode 220, it reached all the goal points but the propagation path is tortuous. **Figure 6** shows the late-stage training results. It can be seen that the agent successfully learnt to control the propagation of crack to reach all the 10 goal points sequentially. Notably, the number of training steps in each learning episode is much smaller than 25 during the late learning episodes. The reward received for each episode during training is shown in **Figure 7**. It shows that the RL agent was able to accomplish the controlled crack propagation task after about 200 episodes of training. However, no more obvious improvements can be observed from episode 200 to 2000.

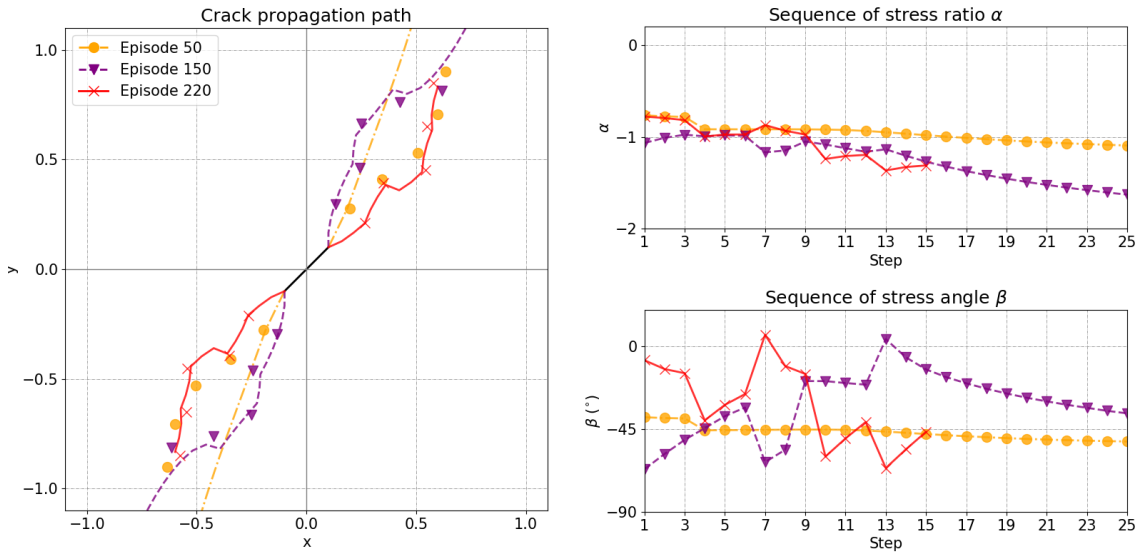

Figure 5. The crack propagation paths (left) and corresponding actions (right) taken by the RL agent at randomly selected early stages of training (episodes 50, 150 and 220). RL agent uses reward function 1 for the desired control. The discrete dotted points on left show predefined discrete goal points that the crack propagation need to track. The crack propagation initiates at the center. A successful control requires the propagating crack to touch each of the 10 discrete goal points per crack path. RL agent's action includes an optimal selection of stress ratio α [−2, 2] and stress angle β [−90°, 90°]. RL agent during early learning episodes is not able to take actions to perfectly track the predefined crack path.

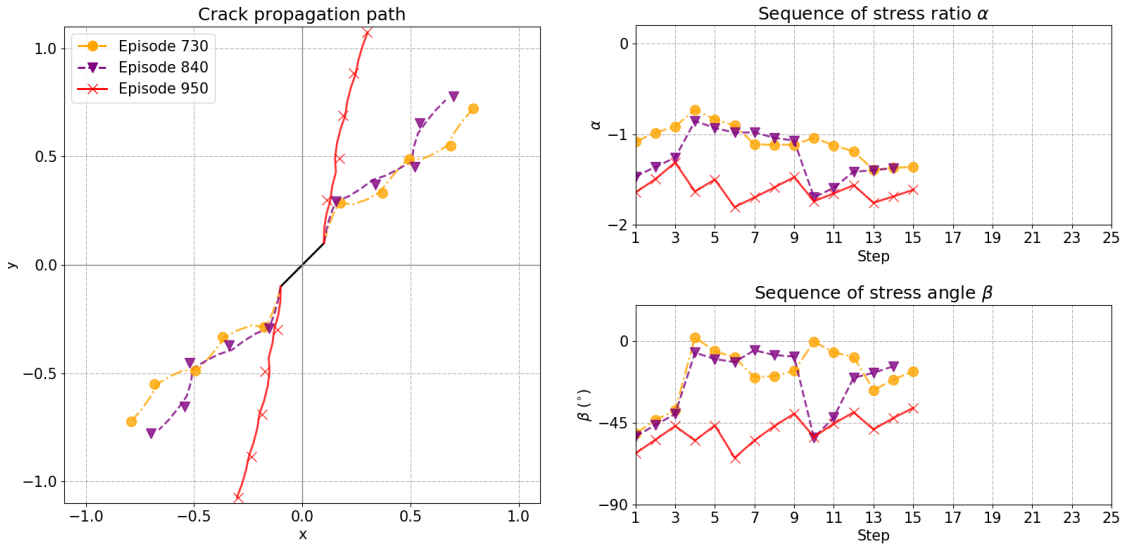

Figure 6. The crack propagation paths (left) and corresponding actions (right) taken by the RL agent at randomly selected early stages of training (episodes 730, 840 and 950). RL agent uses reward function 1 for the desired control. The discrete dotted points on left show predefined discrete goal points that the crack propagation need to track. The crack propagation initiates at the center. A successful control requires the propagating crack to touch each of the 10 discrete goal points per crack path. RL agent during late learning episodes is able to take actions that result in perfect tracking of the predefined crack path.

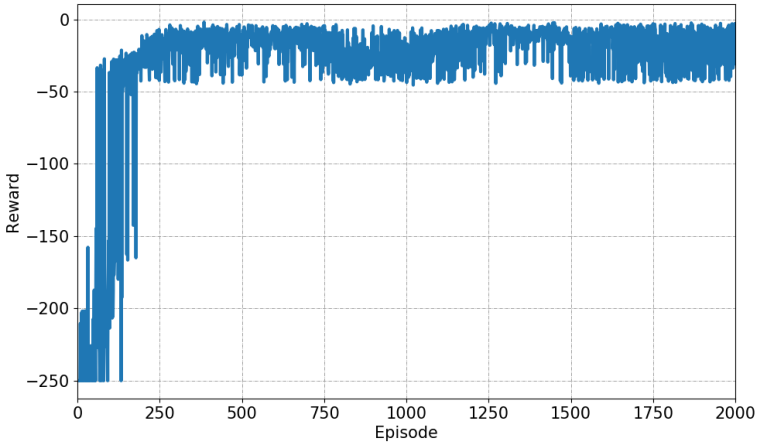

Figure 7. Reward history received by the RL agent when using reward function 1. The reinforcement-learning-based control is achieved in less than 300 learning episodes. The entire training process of 2000 episodes takes 145 seconds on an Intel® Xeon® E5-1650 v3 CPU. The reward does not improve with the increase in learning episodes. The rewards are small negative values during the early training episodes.

Reinforcement-learning based control of discontinuity using reward function 2

This section shows the RL control of discontinuity using reward function 2. The episode ends when the crack reaches all the 10 goal points with a certain error margin or when the crack propagates in the opposite direction of the current goal point. The early-stage and late-stage training results are shown in **Figure 8** and **Figure 9**, respectively. We used the same seed for random number generator to ensure that for each episode, the locations of the goal points are the same as that in the previous case when testing the reward function 1. This helps us compare the training efficiency between using the two different reward functions in a more convenient way. At the early stage of training, the episode quickly ended because the crack propagated to the unfavorable direction. The RL agent was able to control the propagation of crack along the desired path at the late stage of training. **Figure 10** shows the reward received for each episode during training. As we can see, the RL agent is able to control the crack propagation in less than 500 learning episodes. Notably, the reward improves gradually with the learning episodes, because the information of the current-optimal path is provided to the learning agent. By using reward function 2, the agent receives a more accurate reward signal for each step taken, in such a way that it no longer need to find but instead follow the path. Such an improvement was not seen for reward function 1.

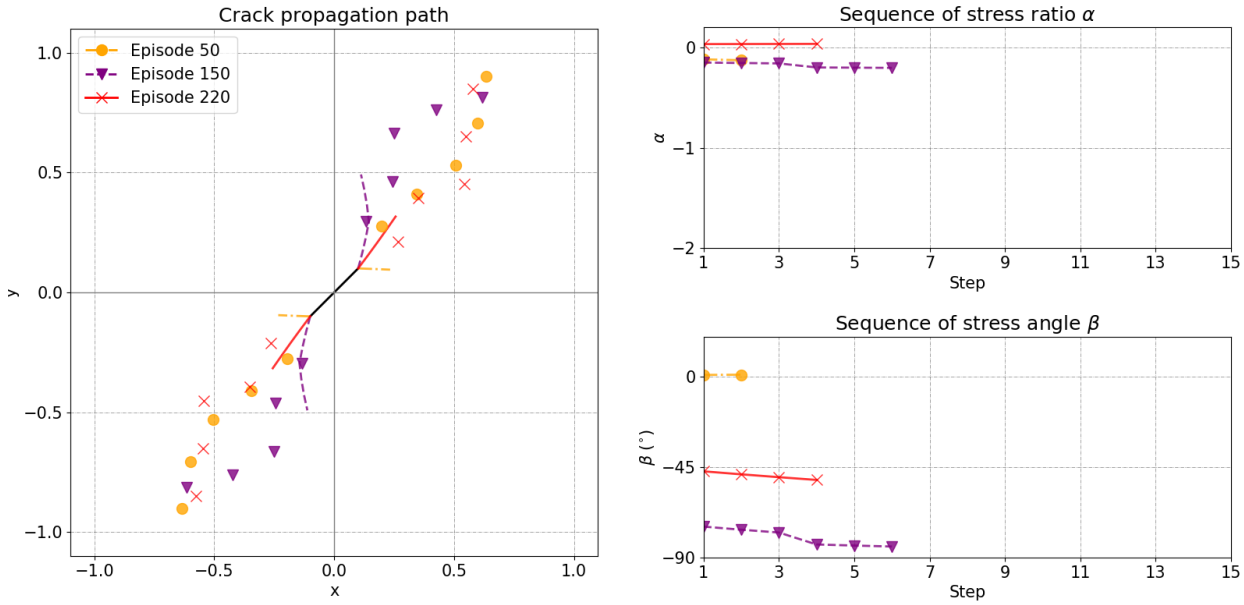

Figure 8. The crack propagation paths (left) and corresponding actions (right) taken by the RL agent at randomly selected early stages of training (episodes 50, 150 and 220). RL agent uses reward function 2 for the desired control. RL agent during early learning episodes is not able to take actions to perfectly track the predefined crack path. The early performance using reward function 2 is worse than the reward function 1.

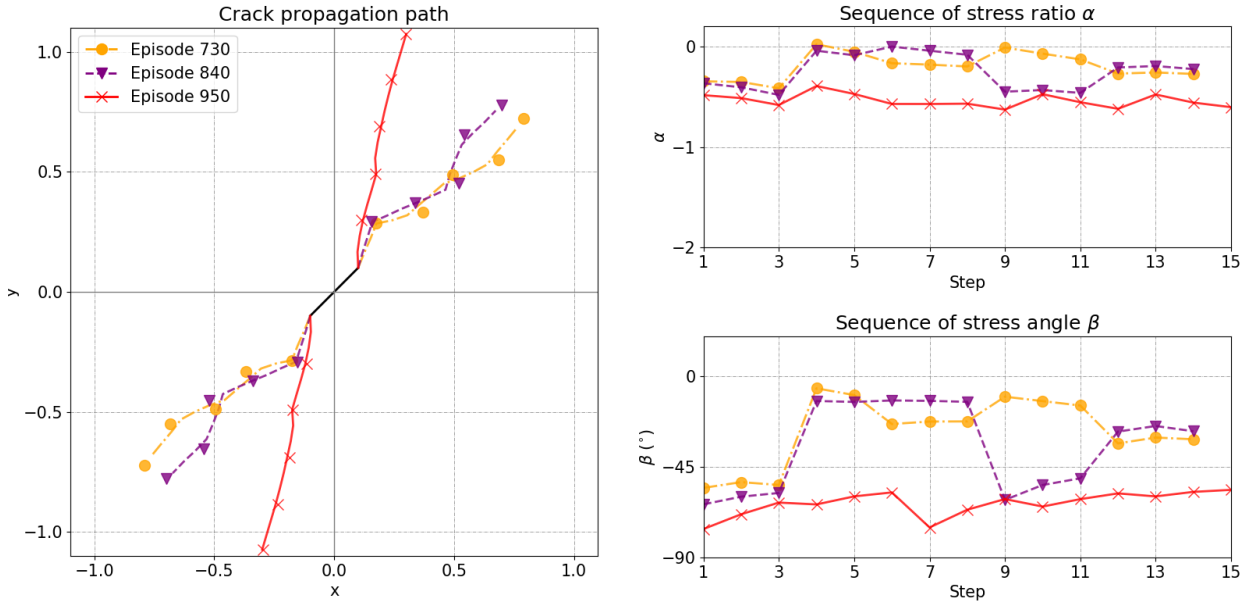

Figure 9. The crack propagation paths (left) and corresponding actions (right) taken by the RL agent at randomly selected early stages of training (episodes 730, 840 and 950). RL agent uses reward function 2 for the desired control. RL agent during late learning episodes is able to take actions that result in perfect tracking of the predefined crack path.

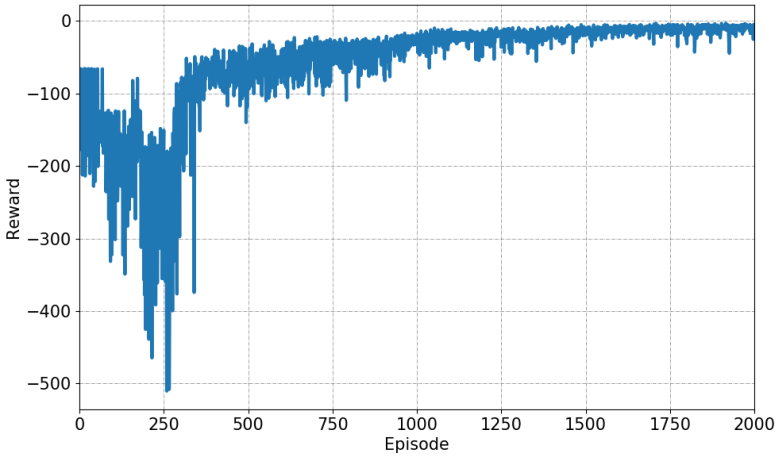

Figure 10. Reward history received by the RL agent when using reward function 2. The reinforcement-learning-based control is achieved in less than 500 learning episodes. The entire training process of 2000 episodes takes 122 seconds on an Intel® Xeon® E5-1650 v3 CPU. Notably, the reward improves with the increase in learning episodes.

Limitations

The training is performed in a self-created environment by coupling a simple 2D governing equation, which may not be accurate for modelling the complex propagation of discontinuity in complex heterogeneous materials. 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 of controlling the mechanical discontinuity. A more generalized environment/framework will be hard to design. The proposed propagation is under biaxial stress field in an infinite material.

Future Works

The DDPG reinforcement learning algorithm is proved to be robust for controlling the propagation of mechanical discontinuity. For the next step, we are planning to couple the DDPG algorithm with commercial software (e.g. Abaqus) that is capable of modeling crack propagation in realistic materials under external stresses. By the combination of reinforcement learning technique and industry-acknowledged software, it would provide more guidance and insights into the application of reinforcement learning in various engineering disciplines.

Conclusions

The paper is a proof-of-concept development and deployment of a reinforcement learning framework to control the propagation of mechanical discontinuity embedded in a homogeneous, planar material in a biaxial 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 propagation of mechanical discontinuity by interacting with a simulated environment based on the OpenAI Gym environment and adaptively changing two engineering parameters, namely, stress ratio and stress angle. 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 the location of propagating crack tip and the location of the current goal point along the desired path.

The key for robust and accurate control is the design of a good reward function. Reward function 1 encourages the crack to reach the goal points as close as possible in limited number of steps. Reward function 2 penalizes the crack for deviating from the desired path by forcing the agent to stay on the shortest linear path. Compared with reward function 1, the reward function 2 provides the information of the shortest linear path, which is the straight line connecting the crack tip before taking the step and the current goal point. The RL agent was successfully trained to accomplish the controlled crack propagation task using both designs of reward function. The training using reward function 1 required less episodes, but the improvements saturated after certain number of training episodes. On the other hand, the use of reward function 2 required more training episodes, but the training behavior gradually improved with the training episode because a more accurate reward signal is provided to the learning agent for each step taken. The reward function that forces the learning agent to stay on the shortest linear path between crack tip and goal point performs much better than the reward function that aims to reach closest to the goal point in minimum number of steps

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

A3C = asynchronous advantage actor-critic
DDPG = deep deterministic policy gradient
DPG = deterministic policy gradient
DQN = deep Q network
MDP = Markov decision process
PPO = proximal policy optimization
RL = reinforcement learning
SARSA = state–action–reward–state–action
TD = temporal difference
TRPO = trust region policy optimization
XFEM = extended finite element method

Symbols

a = action
$\mathcal{A}$ = action space
α = stress ratio
β = stress angle
d_1 = distance between crack tip and the current goal point
d_2 = distance between crack tip and the current optimal path
Δl = crack propagation length at each time step
γ = discount factor
J = expected return from the start distribution
K_I, K_{II}, K_{III} = mode I, II, III stress intensity factors
K_{IC} = mode I critical stress intensity factor/fracture toughness
l = crack half-length
L = mean squared loss
$\mu(s|\theta^{\mu})$ = actor network
$\mu'(s|\theta^{\mu'})$ = target actor network
N = minibatch size
$\mathcal{N}$ = 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 = replay buffer
s = state
$\mathcal{S}$ = state space
σ = stress
τ = target networks update rate
θ = weights in the deep neural network
θ_p = crack propagation angle
y = moving target (TD target)

Subscripts

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

Superscripts

$'$ = denotes target networks
μ = denotes actor network
μ' = denotes target actor network
Q = denotes critic network
Q' = denotes target critic network

Appendix

1. Difference between reward function 1 and 2:
    - Reward function 1 provides only the information of the location of the current goal point. The agent trained using this reward function is expected to explore and find the optimal path.
    - Reward function 2 provides the information of the current-optimal path, which is the straight line connecting the crack tip before taking the step and the current goal point. The agent trained using this reward function is expected to follow the current-optimal path.
2. For episodes 730, 840 and 950, a comparison of crack path and sequence of stress ratio α using reward function 1 and 2 is shown in **Figure A1**, a comparison of crack path and sequence of stress angle β using reward function 1 and 2 is shown in **Figure A2**.

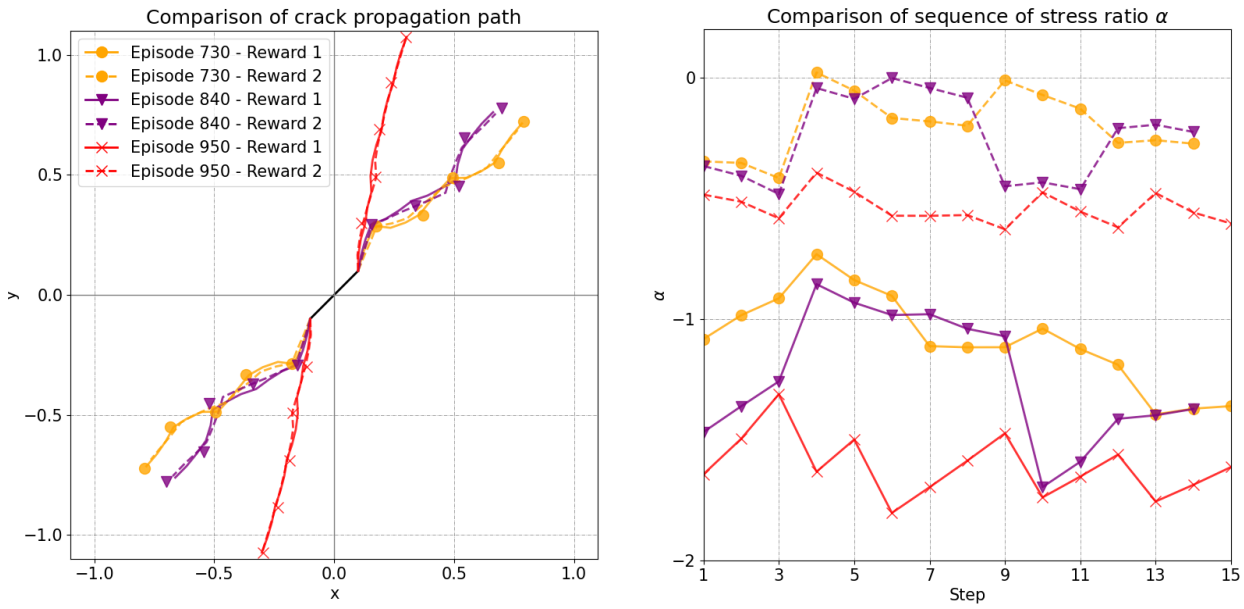

Figure A1. Comparison of crack path and sequence of stress ratio α using reward function 1 and 2.

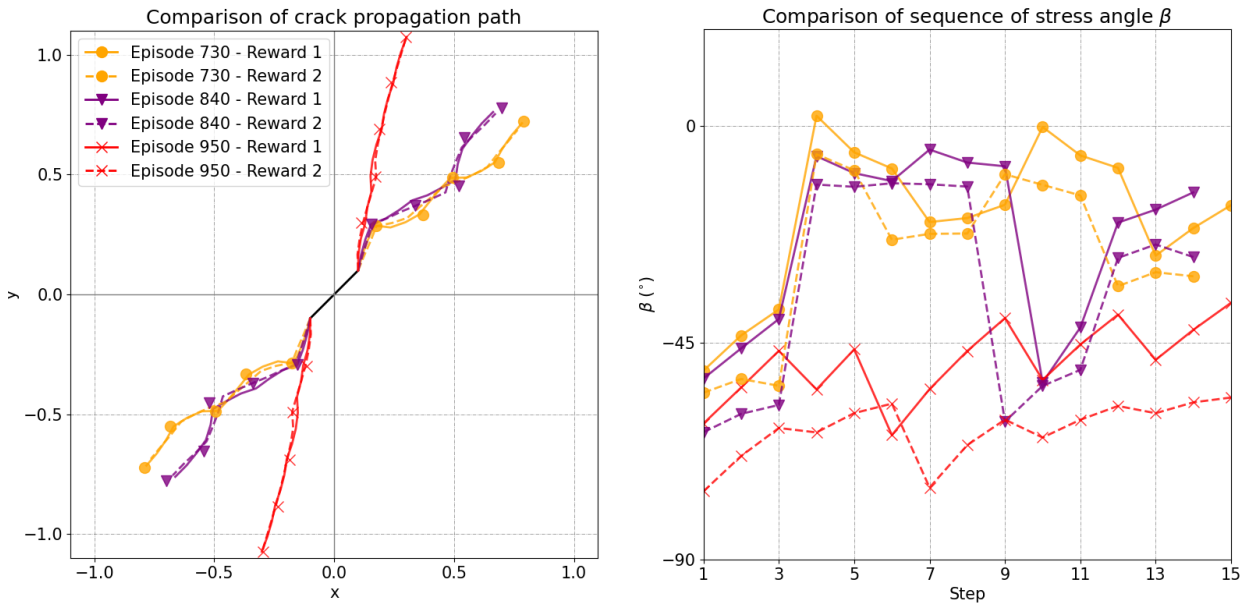

Figure A2. Comparison of crack path and sequence of stress angle β using reward function 1 and 2.

3. The main challenges encountered during developing this paper are as follows:
    - What should be the state seen by the RL agent? Should we use location, distance, or azimuth angle? Should we include any directional information? Should we provide the RL agent the locations of all the 10 goal points at the very beginning, or provide them one-by-one as crack propagates?
    - What should be a proper design of the reward function? Should we use a binary reward or a reward that that adapts during the learning steps?
    - How to properly tune the parameters?

References

Chen, Guangchang, Zhang, Jindong, Liu, Gang, Chen, Puhui & Guo, Miaocai 2019. Controlling the crack propagation path of the veil interleaved composite by fusion-bonded dots. *Polymers,* 11**,** 1260.

Cheng, Yugang, Lu, Yiyu, Ge, Zhaolong, Cheng, Liang, Zheng, Jingwei & Zhang, Wenfeng 2018. Experimental study on crack propagation control and mechanism analysis of directional hydraulic fracturing. *Fuel,* 218**,** 316-324.

Cheng, Zhenxing & Wang, Hu 2018. How to control the crack to propagate along the specified path feasibly? *Computer Methods in Applied Mechanics Engineering,* 336**,** 554-577.

Choi, Seungwon, Kim, Suseong & Kim, H Jin 2017. Inverse reinforcement learning control for trajectory tracking of a multirotor UAV. *International Journal of Control, Automation Systems,* 15**,** 1826-1834.

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

Guevara, Jl, Patel, Rajan G & Trivedi, Japan J. Optimization of Steam Injection for Heavy Oil Reservoirs Using Reinforcement Learning. SPE International Heavy Oil Conference and Exhibition, 2018. Society of Petroleum Engineers.

Haverkamp, Cole B, Hwang, Dohgyu, Lee, Chanhong & Bartlett, Michael D 2021. Deterministic control of adhesive crack propagation through jamming based switchable adhesives. *Soft Matter,* 17**,** 1731-1737.

Hourfar, Farzad, Bidgoly, Hamed Jalaly, Moshiri, Behzad, Salahshoor, Karim & Elkamel, Ali 2019. A reinforcement learning approach for waterflooding optimization in petroleum reservoirs. *Engineering Applications of Artificial Intelligence,* 77**,** 98-116.

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.

Kober, Jens, Bagnell, J Andrew & Peters, Jan 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research,* 32**,** 1238-1274.

Kumar, Abhishek & Sharma, Rajneesh 2018. Linguistic Lyapunov reinforcement learning control for robotic manipulators. *Neurocomputing,* 272**,** 84-95.

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

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.

Miftakhov, Ruslan, Al-Qasim, Abdulaziz & Efremov, Igor. Deep reinforcement learning: Reservoir optimization from pixels. International Petroleum Technology Conference, 2020. OnePetro.

Mnih, Volodymyr, Badia, Adria Puigdomenech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy, Harley, Tim, Silver, David & Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. International conference on machine learning, 2016. PMLR, 1928-1937.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K & Ostrovski, Georg 2015. Human-level control through deep reinforcement learning. *Nature*, 518**,** 529-533.

Namazu, T, Ohtani, K, Yoshiki, K & Inoue, S. Crack propagation direction control for crack-less solder bonding using Al/Ni flash heating technique. 2011 16th International Solid-State Sensors, Actuators and Microsystems Conference, 2011. IEEE, 1368-1371.

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

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

Peng, Chi & Ma, Jianjun 2021. Online integral reinforcement learning control for an uncertain highly flexible aircraft using state and output feedback. *Aerospace Science and Technology*, 109**,** 106442.

Perez, Nestor 2017. Linear-elastic fracture mechanics. *Fracture Mechanics.* Springer.

Pyrak-Nolte, Laura J, Depaolo, Donald J & Pietraß, Tanja 2015. Controlling subsurface fractures and fluid flow: a basic research agenda. USDOE Office of Science (SC)(United States).

Rabault, Jean, Kuchta, Miroslav, Jensen, Atle, Réglade, Ulysse & Cerardi, Nicolas 2018. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *arXiv preprint arXiv:.07664*.

Ren, Feng, Rabault, Jean & Tang, Hui 2020. Applying deep reinforcement learning to active flow control in turbulent conditions. *arXiv preprint arXiv:.10683*.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael & Moritz, Philipp. Trust region policy optimization. International conference on machine learning, 2015. PMLR, 1889-1897.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec & Klimov, Oleg 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:.06347*.

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.

Sugita, Naohiko, Osa, Takayuki, Aoki, Ryoma & Mitsuishi, Mamoru 2009. A new cutting method for bone based on its crack propagation characteristics. *CIRP annals - Manufacturing Technology*, 58**,** 113-118.

Talavera, Alvaro Gustavo, Tupac, Yvan Jesus & Vellasco, Marley Maria Bernardes Rebuzzi. Controlling oil production in smart wells by MPC strategy with reinforcement learning.

SPE Latin American and Caribbean Petroleum Engineering Conference, 2010. Society of Petroleum Engineers.

Thornton, Charles E, Kozy, Mark A, Buehrer, R Michael, Martone, Anthony F & Sherbondy, Kelly D 2020. Deep Reinforcement Learning Control for Radar Detection and Tracking in Congested Spectral Environments. *IEEE Transactions on Cognitive Communications Networking,* 6**,** 1335-1349.

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

Wu, Qianlin, Zhu, Dandan, Liu, Yi, Du, Aimin, Chen, Dong & Ye, Zhihui. Comprehensive Control System for Gathering Pipe Network Operation Based on Reinforcement Learning. Proceedings of the 2018 VII International Conference on Network, Communication and Computing, 2018. 34-39.

Xu, Baoxing, Chen, Xi & Waisman, Haim 2009. Crack propagation toward a desired path by controlling the force direction. *Engineering fracture mechanics,* 76**,** 2554-2559.

Zhang, Chiyuan, Vinyals, Oriol, Munos, Remi & Bengio, Samy 2018. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:.06893*.