Dynamic metasurface control using Deep Reinforcement Learning

Ying Zhao, Liang Li, Stéphane Lanteri, Jonathan Viquerat

Dynamic metasurface control using Deep Reinforcement Learning

Ying Zhao[a], Liang Li[a,*], Stéphane Lanteri[b], Jonathan Viquerat[c]

[a] *School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, P.R. China*

[b] *INRIA, 2004 Route des Lucioles, BP 93 06902 Sophia Antipolis Cedex, France*

[c] *CEMEF Mines ParisTech, 1, rue Claude Daunesse, BP207 06904 Sophia Antipolis Cedex, France*

Abstract

Dynamic metasurface is an emerging concept for achieving a flexible control of electromagnetic waves. Generalized sheet transition conditions (GSTCs) can be used to model the relationship between the electromagnetic response and surface susceptibility parameters characterizing a metasurface. However, when it comes to the inverse problem of designing and controlling a metasurface in a space-time varying context based on GSTCs, the dynamic synthesis of the susceptibility parameters is a difficult and non-intuitive task. In this paper, we transform the inverse problem of solving dynamic susceptibility parameters into a sequence of control problems. Based on FDTD numerical simulations, a Deep Reinforcement Learning (DRL) framework using a proximal policy optimization (PPO) algorithm and a fully connected neural network is designed to control the susceptibility parameters intelligently and efficiently, promoting the further expansion of the application range of metasurface and thus helping realize more flexible and effective control of electromagnetic waves. We provide numerical results in a one-dimensional setting to show the applicability, correctness and effectiveness of the proposed approach.

Keywords: Dynamic metasurface, Artificial neural networks, Deep Reinforcement Learning, Proximal Policy Optimization,

1. Introduction

Metasurfaces are two-dimensional artificial layered electromagnetic structures composed of sub-wavelength scattering particles arranged on a plane, which can be regarded as the low-dimensional extension and functional expansion of general three-dimensional metamaterials [1]. The compact structure and unique composition endow them with flexible control capabilities for electromagnetic waves, and enable them to find application potential in a variety of fields, such as broadband focusing [2], holograms generation [3], solar cells [4] and remote processing [5]. However, although the

*Corresponding author

Email addresses: Yingzhao@std.uestc.edu.cn (Ying Zhao), plum_liliang@uestc.edu.cn, plum.liliang@gmail.com (Liang Li), Stephane.Lanteri@inria.fr (Stéphane Lanteri), jonathan.viquerat@mines-paristech.fr (Jonathan Viquerat)

past few decades have witnessed many mature applications of the metasurface technology, most of the metasurfaces that have been proposed are static. To transform electromagnetic waves in a specified manner, engineers carefully select and arrange the scattering particles to design the required metasurface whose structure and composition is thus fixed, meaning that it will be equipped with fixed and limited functions once designed and shaped. From this point of view, static metasurfaces tend to cause a waste of resources, since their ability to control electromagnetic waves cannot be adjusted and new metasurfaces need to be fabricated from scratch for different purposes.

Therefore, as science and technology move towards new physics, new devices, and new applications, an increasingly considered option is to introduce time-dependent capabilities in metasurfaces by modulating each individual scattering particle in time [6, 7]. Dynamic metasurfaces avoid the cost of repetitive manufacturing, and also get rid of some physical constraints unique to static materials, such as the Bode-Fano criterion, Lorentz reciprocity, etc. [7, 8] thereby bringing more degrees of freedom to control electromagnetic waves. Especially with the advent of tunable materials [9] such as graphene, susceptibilities of dynamic metasurfaces can be adjusted in a flexible way according to the surrounding electromagnetic field to precisely transform the incident wave into any specified desired scattered field.

In the present work, we focus on the problem of adaptive control of susceptibilities, which is an important component in the design of dynamic metasurfaces [10]. Automatically determining susceptibilities by collecting the sensed field information not only realizes the transition from passive design to active design, but also facilitate flexible function switching of metasurfaces [6].

Nevertheless, finding efficient active control strategies is not a simple matter. The generalized sheet transition conditions (GSTCs) [11, 12], a widely used metasurface modeling method, closely link the incident field, the scattered field and the susceptibilities at any time, providing an analytical solution for the susceptibility parameters [7]. However, the premise of the utilization of GSTCs is the specific expressions for the desired reflected and transmitted wave, and it also involves complex integration operations, which often leads to failure to satisfy most applications. When faced with some very simple situations, it may be feasible for engineers to specify susceptibilities based on their physical intuition and experience, which is a totally human-based method and is thus usually inefficient and infeasible when dealing with complex problems due to the huge search space.

A huge number of high-profile applications of machine learning methods implies that data-driven approaches have increasingly been able to satisfy the pursuit of intelligence in various fields without manual assistance. Deep learning techniques draw support from powerful function approximation capabilities of artificial neural networks and are naturally suitable for solving complex and nonlinear systems [13, 14]. Recently, researchers designed an environmental adaptive metasurface by combining deep learning and coded metasurface [6], in which the external environmental information is perceived by a sensor. A deep neural network is resorted to process these information and quickly output control signals, based on which the controller subsequently adjusts diodes on the metasurface to achieve the purpose of adapting to the environment. In similar way, H.S. Chen [15] has developed an adaptive stealth device by training deep neural network to control the voltage of each unit on the metasurface. The key step of a deep learning method is to train a deep neural network using a pre-established static training set and learn the mapping between the control parameters and the target response. When it comes to the design of adaptive devices, due to the need

to consider a huge number of possible environmental states, the training set preparation is usually a huge workload, and if a environmental state does not lie within the coverage of the training set, users cannot accurately retrieve the corresponding control signal.

In contrast, imitating the way human learn knowledge, Reinforcement Learning (RL) [16] is a machine learning method for solving complex tasks through trial-and-error, which employs reward or punishment to drive the exploratory learning. Instead of using a labeled training data set to approximate the mapping function between the input and output space in a one-shot decision style, RL generates training data in the interaction with the environment, employs the environment to evaluate the strategy, and deploys the relevance between decisions, which is thus a sequential decision-making process. In the case of RL, an agent interacts with an environment in a close-loop fashion: by observing environment state, the agent performs an action to influence the environment, while the environment will then feed back a numeric reward to the agent, based on which the agent optimizes its cognition to maximize the received reward. Since RL is mostly agnostic to the environment details and there is no need to prepare the training set in advance, it is a smart method truly free of human intervention and is suitable for any task that can be expressed or transformed into a sequential decision problem and can provide a "state-action-reward" interactive interface. Especially with the aid of a deep neural network, Deep Reinforcement Learning (DRL) has demonstrated unprecedented advantages in the field of artificial intelligence, and has been widely used in various sequence decision tasks, from Go [17], Atari [18] to smart medical [19], physical simulation [20, 21] and optimal control [22]. Yet, there is only very sparse literature discussing the DRL in the design of dynamic nano-devices, and no attempt has been made on the control of adaptive dynamic metasurfaces.

Therefore, motivated by the sequential decision-making characteristics of DRL and its potential in solving optimal control problems, in this work, we regard the problem of calculating the time-varying susceptibilities as a time-series decision-making problem, and for the first time evaluate the feasibility of DRL for the control of dynamic metasurfaces. Combining with fully connected neural network, Proximal Policy Optimization (PPO) [23] algorithm is exploited to discover the control strategies of susceptibilities, with a FDTD scheme based on GSTCs [11, 12, 24] being applied to provide an interactive environment and evaluate the output of the neural network. In the following, we describe the time-varying form of GSTCs and derive the analytical solution of susceptibilities in section 2. Section 3 outlines the basic principles of DRL and PPO. The numerical method of simulating dynamic metasurfaces and the designed DRL framework for learning susceptibilities are addressed in section 4. Finally, in section 5 we provide two 1D control results to assess the feasibility of the proposed framework.

2. Generalized sheet transition conditions

According to Huygens principles, metasurfaces are best regarded as zero-thickness "surface polarization current sheets" [10], which support both electric and magnetic field discontinuities and can be characterized via the four 3×3 surface susceptibility tensors $\bar{\bar{\boldsymbol{\chi}}}_{ee}$, $\bar{\bar{\boldsymbol{\chi}}}_{mm}$, $\bar{\bar{\boldsymbol{\chi}}}_{em}$ and $\bar{\bar{\boldsymbol{\chi}}}_{me}$. By combining surface polarization densities with Maxwell equations, generalized sheet transition conditions (GSTCs) is a commonly used approach to model a metasurface.

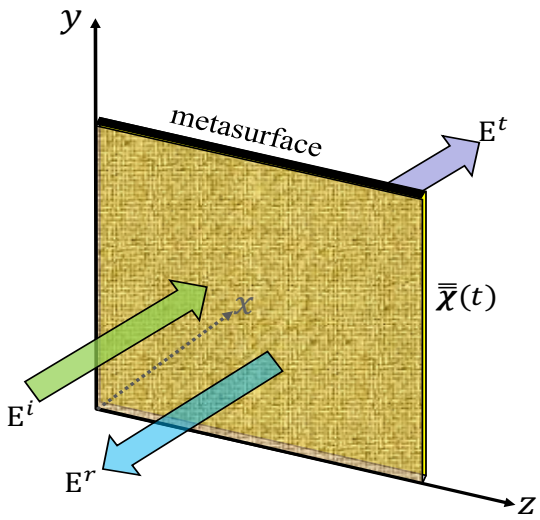


Figure 1: General dynamic metasurface with time-modulated susceptibilities, transforming an incident wave $\mathbf{E}^i$ into specified transmitted wave $\mathbf{E}^t$ and reflected wave $\mathbf{E}^r$.

We consider a non-dispersive time-varying metasurface as depicted in Fig. 1, where an incident wave $\mathbf{E}^i$ is transformed into a transmitted wave $\mathbf{E}^t$ and a reflected wave $\mathbf{E}^r$. At any moment t, the time-varying form of GSTCs relates the field discontinuity to the effective polarization currents on the metasurface

$$\begin{cases} \mathbf{n} \times \Delta\mathbf{H} = \dfrac{\partial \mathbf{P}_t}{\partial t} - \mathbf{n} \times \nabla \mathbf{M}_n, \\ \mathbf{n} \times \Delta\mathbf{E} = -\mu_0 \dfrac{\partial \mathbf{M}_t}{\partial t} - \dfrac{1}{\varepsilon_0} \mathbf{n} \times \nabla \mathbf{P}_n, \end{cases} \tag{1}$$

in which the terms on the left-hand sides represent the difference of the fields around the metasurface, i.e., $\Delta\boldsymbol{\Psi} = \boldsymbol{\Psi}^+ - \boldsymbol{\Psi}^-$, $\boldsymbol{\Psi} = \mathbf{E}$ or $\mathbf{H}$, with $+$ and $-$ respectively specifying the fields downstream an upstream of the metasurface. $\mathbf{P}$ and $\mathbf{M}$ are the electric and magnetic surface polarization densities, respectively, $\mathbf{n}$ is the metasurface unit normal vector, and subscripts n and t refer to the normal and tangential components with respect to the metasurface.

For simplicity, we constrain ourselves here to a diagonal monoanisotropic metasurface that does not support normal polarization currents, i.e. $\mathbf{P}_n = \mathbf{M}_n = 0$ and

$$\begin{cases} \bar{\bar{\boldsymbol{\chi}}}_{ee} = \chi_{ee}^{zz} \widehat{\mathbf{z}} + \chi_{ee}^{yy} \widehat{\mathbf{y}}, \\ \bar{\bar{\boldsymbol{\chi}}}_{mm} = \chi_{mm}^{zz} \widehat{\mathbf{z}} + \chi_{mm}^{yy} \widehat{\mathbf{y}}, \\ \bar{\bar{\boldsymbol{\chi}}}_{em} = \bar{\bar{\boldsymbol{\chi}}}_{me} = 0. \end{cases} \tag{2}$$

With the constitutive relations

$$\begin{cases} \mathbf{P}_t(t) = \dfrac{\varepsilon_0 \bar{\bar{\boldsymbol{\chi}}}_{ee}(t)(\mathbf{E}^+(t) + \mathbf{E}^-(t))}{2}, \\ \mathbf{M}_t(t) = \dfrac{\bar{\bar{\boldsymbol{\chi}}}_{mm}(t)(\mathbf{H}^+(t) + \mathbf{H}^-(t))}{2}, \end{cases} \tag{3}$$

the sought after susceptibilities are thus found as

$$\begin{cases} \bar{\bar{\boldsymbol{\chi}}}_{ee}(t) = \dfrac{2}{\varepsilon_0}\displaystyle\int_{-\infty}^{t} \mathbf{n} \times [\mathbf{H}^+(t') - \mathbf{H}^-(t')]dt'./(\mathbf{E}^+(t) + \mathbf{E}^-(t)), \\ \bar{\bar{\boldsymbol{\chi}}}_{mm}(t) = -\dfrac{1}{\mu_0}\displaystyle\int_{-\infty}^{t} \mathbf{n} \times [\mathbf{E}^+(t') - \mathbf{E}^-(t')]dt'./(\mathbf{H}^+(t) + \mathbf{H}^-(t)), \end{cases} \tag{4}$$

where ./ is the array division operator. Replacing relations (3) with its bianisotropic version, equations (4) can be easily generalized to the other general metasurface (see [10] for more details).

From equations (4), with the premise being the known form of incident and scattered fields, the analytical calculations for susceptibilities $\bar{\bar{\boldsymbol{\chi}}}_{ee}(t)$ and $\bar{\bar{\boldsymbol{\chi}}}_{mm}(t)$ involve complex integration and division operations, which is difficult and unfeasible for most practical applications. However, from another point of view, solving time-dependent susceptibilities is actually a time sequential control problem and aims to discover how to use the past fields data perceived by the metasurface to decide the future susceptibilities, in view of which, reinforcement learning is a potential candidate to handle this task.

3. Deep reinforcement learning and proximal policy optimization

3.1. Deep neural networks

Inspired by biological neural structures, a deep neural network (DNN) is an arithmetic model made up of unit neurons. As sketched in Fig. 2(a), provided an input vector $\mathbf{x}$, accompany with a weight vector $\mathbf{w}$ and a bias value b, a neuron performs a weighted summation $\mathbf{w} \cdot \mathbf{x} + b$, whose output is computed after applying an activation function $o = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$. In a DNN, by connecting many unit neurons together, the output of one neuron can be the input of another neuron, for example, Fig. 2(b) shows a simple fully connected network composed of an input layer, L hidden layers and an output layer. Repeating the calculation pattern of a single neuron, the deep neural network is actually a deep composite function and is capable of acting as an universal function approximator [13] with unprecedented efficiency and accuracy, which have shed light on their ability to handle many complex tasks.

The learning of the neural network consists in the process of iteratively adjusting all the bias and weight values in order to reduce the value of a predefined loss function that measures the quality of the network prediction. This is an optimization problem, which is usually solved by gradient descent algorithm [14, 25], in which the gradient of the loss function with respect to the weights and bias are obtained through a back-propagation algorithm.

3.2. Deep reinforcement learning

As mentioned in the introduction, the reinforcement learning (RL) process can be formulated as a closed-loop of roughly three stages as sketched in Fig. 3. At each interactive step, firstly, the agent receives an (possibly noisy) observation o_t of the environment state s_t (for convenience, we assume $o_t = s_t$ and only use s_t in following paragraphs). Secondly, based on s_t, the agent executes an

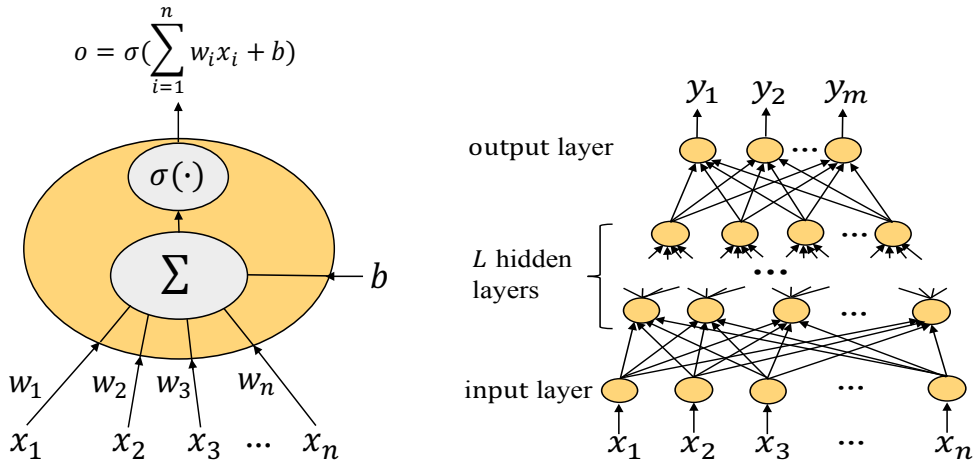


Figure 2: Artificial neural networks. (a) Presentation of the functioning of a unit neuron. (b) A fully connected neural network with L hidden layers, mapping an input in $\mathbb{R}^n$ to an output in $\mathbb{R}^m$.

action a_t to influence subsequent evolution states of the environment (i.e., $s_{t+1}, s_{t+2}, \cdots$). Finally, the environment evaluates the quality of a_t and provides the agent with a feedback information r_t, according to which the agent updates its cognition and then obtains the observation s_{t+1} of the next environment state. This loop repeats until some termination conditions are reached. The purpose of the agent is to discover an optimal decision policy $\pi : s_t \rightarrow a_t$ (get an action a_t under a certain observation s_t) that can maximize the discounted cumulative reward [16] over an episode, i.e. a sequence of state-action-reward triples $\tau = ((s_0, a_0, r_0), (s_1, a_1, r_1), (s_2, a_2, r_2), \cdots, (s_T, a_T, r_T))$:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t \tag{5}$$

where $\gamma \in [0, 1]$ is a discount factor used to show the preference to the immediate over the distant rewards and to ensure the convergence of the cumulative reward in the case of very long episode.

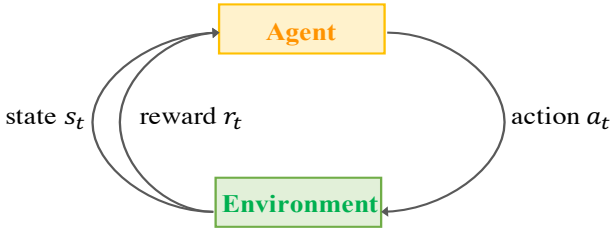


Figure 3: Sketch of the generic reinforcement learning process

Leveraging the advantages of DNNs, deep reinforcement learning (DRL) uses a DNN to play the role of the agent and optimizes the decision policy in the process of DNN training. Two important

reinforcement learning algorithms used to DNN training are the value-based method represented by Q-learning algorithm [18, 26] and the policy-based method represented by the Policy Gradient (PG) algorithm [20, 27]:

1) The value-based Q-learning method focuses on learning a value function to evaluate the performance of an action and gives a policy by selecting the action with the highest value. In this method, the neural network with parameters θ works as an action-value function approximator that maps the state and action to a value $Q^{\pi}(s,a;\theta) = \mathbb{E}_{\tau\sim\pi}\left[R(\tau)|s_0 = s, a_0 = a;\theta\right]$, which is the expected discounted cumulative reward after taking action a at the starting state s and afterwards continuing to act according to policy π. Taking s and a as inputs, the neural network outputs the value of $Q(s,a)$, and constantly update itself to approach the optimal solution via temporal difference (TD) learning [16] by exploiting the Bellman equation

$$Q(s,a;\theta_{i+1}) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a) + \gamma \max_{a'} Q(s',a';\theta_i)\right]. \tag{6}$$

2) Instead of adopting a value function to determine the optimal policy, the policy-based policy gradient (PG) method is known as a probability-based method, in which the neural network is instructed to directly output the probability distribution of actions $\pi_\theta(s,a)$ and to take actions using sampling. In order to discover the optimal policy π_{θ^*}, vanilla PG uses the gradient ascent algorithm to perform the optimization by maximizing the expected discounted cumulative reward $J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right]$, hence the optimal parameters θ^* are sought as

$$\theta^* = \arg\max_{\theta} J(\pi_\theta). \tag{7}$$

In light of the policy gradient theorem [28], the gradient of policy performance with respect to parameters θ is derived as [20, 29]

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right] \\
&= \nabla_\theta \int_\tau P(\tau|\theta) R(\tau) \\
&= \int_\tau \nabla_\theta P(\tau|\theta) R(\tau) \\
&= \int_\tau P(\tau|\theta) \nabla_\theta \log P(\tau|\theta) R(\tau) \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log P(\tau|\theta) R(\tau)\right],
\end{aligned} \tag{8}$$

where the very first state s_0 is sampled from the start-state distribution ρ_0 and the episode probability is

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t), \tag{9}$$

whose log value gradient can thus be expressed as

$$\nabla_\theta \log P(\tau|\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{10}$$

As a result, the policy gradient to network parameters are commonly estimated using a set of episodes $\mathcal{D}$ as

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] \approx \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau). \tag{11}$$

Moreover, for neural network, its loss function now can be expressed as

$$L(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\log \pi_\theta(a_t|s_t)R(\tau)\right]. \tag{12}$$

According to whether the action space is continuous or discrete, the random policies to be learnt need to be further divided into "categorical policies" and "diagonal Gaussian policies" [30].

(i) Categorical policies are naturally appropriate for discrete action space. Building a neural network for a categorical policy is just like building a classifier network, which inputs an observation and outputs a probability of all action value, i.e. a vector form probability distribution $P_\theta(s)$. The agent samples on the basis of vector $P_\theta(s)$, with the log likelihood of the sampled action being given by

$$\log \pi_\theta(a|s) = \log[P_\theta(s)]_a. \tag{13}$$

(ii) Assume a continuous n-dimensional action space, the diagonal Gaussian policies focus on determining a Gaussian using its mean $\mu = \mu_\theta(s)$ and logarithmic variance $\log\sigma = \log\sigma_\theta(s)$, in terms of which, the agent do sampling with

$$a = \mu_\theta(s) + \sigma_\theta(s)\odot z, \tag{14}$$

where $z \sim \mathcal{N}(0, I)$ denotes a noise from a spherical Gaussian and $\odot$ is used to perform the element-wise product of two vectors. The log likelihood can hence be obtained as

$$\log \pi_\theta(a|s) = -\frac{1}{2}(\sum_{i=1}^{n}(\frac{(a_i-\mu_i)^2}{\sigma_1^2} + 2\log\sigma_i)) + n\log 2\pi. \tag{15}$$

In Q-learning, since the environment feedback is definite, the agent takes actions in a deterministic way, and thus even small variations of value function is likely to completely change the selected action, thereby leading to a lack of robustness in the convergence process. In contrast, through learning stochastic policies, the policy-based PG method possesses better convergence properties and exhibit better capabilities in dealing with high dimensional action spaces.

However, the vanilla PG method owns the nature of high variance and has the difficulty of low learning efficiency. Two useful tricks used to reduce variance are

(i) Using reward-to-go policy gradient:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{t'=t}^{T}R(s_{t'}, a_{t'}, s_{t'+1})\right], \tag{16}$$

where $\widehat{R}_t \doteq \sum_{t'=t}^{T}R(s_{t'}, a_{t'}, s_{t'+1})$ is called reward-to-go. In this form, actions can only be reinforced on the basis of rewards obtained after they are taken and executed.

(ii) Subtracting a baseline function $V^{\pi_\theta}(s)$ from $\widehat{R}_t$ can further reduce the variance, where the state-value function $V^{\pi_\theta}(s) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)|s_0 = s]$ represents the expected discounted cumulative reward obtained through a sequence of interaction following the policy π when starting in state s. Observing that $\widehat{R}_t \approx Q(s_t, a_t)$, one defines an advantage function $A^{\pi_\theta}(s, a) = Q(s_t, a_t) - V^{\pi_\theta}(s_t)$ to depicts how much better to take action a in state s than to the average result of all actions that could be taken in states s. Thus a most commonly used gradient estimator is

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s, a)\right], \tag{17}$$

and the neural network loss function can be redefined as

$$L(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s, a)\right]. \tag{18}$$

Actor-critic framework. The actor-critic framework integrates the advantages of the value-based method and the policy-based method. Unlike traditional policy-based method, which is based on the cumulative reward of a complete episode, policy in actor-critic framework can be updated every step and does not need to wait for a complete episode to be completed, which is beneficial to improve learning efficiency and thus is the most commonly considered framework in practical applications [31].

The gradient estimator introduced in equation (17) can essentially be divided into two parts: one is the policy π_θ, which behaves as an "actor" to select and perform an appropriate action. The other is the advantage function A^{π_θ}, which can be regarded as a "critic" and plays the role of evaluating the actor performance. The actor-critic method proposes to exploit two networks to complete works for both actor and critic in parallel, in which the actor network takes an action in terms of the environment state and the critic network then offers a score for this action. In practice, founding on the fact that $Q(s_t, a_t) \approx r_t + V^{\pi_\theta}(s_{t+1})$, the advantage function is usually approximated as $A^{\pi_\theta}(s_t, a_t) = r_t + V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$ to avoid having to evaluate two value functions. Besides, especially in policy-based method, $A^{\pi_\theta}(s_t, a_t)$ is always estimated using generalized advantage estimation (GAE) [32] to help reduce the variance of gradient estimation and thus reduce the number of samples needed for training.

Proximal policy optimization. The PG method updates its parameters as

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta_k} J(\pi_{\theta_k}), \tag{19}$$

in which α represents the step length of update and a suitable step length should at least make the value of the reward after the policy update not worse. However, as a hyper parameter, α needs to be set manually. If set inappropriately, the updated policy becomes worse, causing the worse learning and the possible final crash. How to find a new policy to make the value of the new reward monotonously increase (or monotonously not decrease) is imperative for reinforcement learning.

Trust region policy optimization (TPRO) was introduced in 2015 [33] as an essential modification to vanilla PG to cope with the unsteady policy update problem by imposing a trust region constraint

to limit the difference between the old and updated policies. With this method, the optimization object is replaced by a surrogate loss

$$L(\theta) = \mathbb{E}_{(s,a)\sim\pi_{\theta_k}}[\frac{\pi(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a)], \tag{20}$$

which is maximized subjected to a constraint on the size of update $\mathrm{KL}(\theta,\theta_k) < \delta$ with $\mathrm{KL}(\theta,\theta_k)$ being the Kullback-Leibler divergence between old and updated policy distribution. The standard solution of TRPO consists of the second-order Taylor expansion for the constraint, which however will be very computationally expensive when the policy is represented by a deep neural network.

As a first-order approximation of TRPO, proximal policy optimization (PPO) was proposed by OpenAI in 2017 [23], which not only has good convergence performance (especially for continuous control problems), but is also easier to implement than TRPO method. Instead of using Kullback-Leibler divergence constraint, the PPO algorithm directly introduces the clipping operation in the surrogate loss

$$L^{CLIP}(\theta) = \mathbb{E}_{(s,a)\sim\pi_{\theta_k}}[\min(\frac{\pi(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a)))], \tag{21}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geqslant 0 \\ (1-\epsilon)A & A < 0, \end{cases} \tag{22}$$

and $\epsilon \in [0.1, 0.3]$ is a user-defined parameter defining how far away the updated policy is allowed to go from the old. The positive value of $A^{\pi_{\theta_k}}(s,a)$ indicates that taking action a in state s is preferable to randomly sampling an action from π_{θ_k} and it is natural to update the policy to favor this action. Furthermore, the ratio $\frac{\pi(a|s)}{\pi_{\theta_k}(a|s)}$ is clipped to $1+\epsilon$ to avoid too large policy update. Conversely, the negative value of $A^{\pi_{\theta_k}}(s,a)$ means that taking action a in state s represents a poorer choice than other random action and it is thus natural to update the policy to decrease the probability of taking this action. In the same fashion, the ratio $\frac{\pi(a|s)}{\pi_{\theta_k}(a|s)}$ is clipped to $1-\epsilon$ to ensure the steadiness of update.

In most practical applications, PPO is always implemented by the actor-critic framework with shared parameters between the policy network and the value function network [21]. By doing so, it's necessary to define a loss function that combines the policy surrogate loss and the value function loss

$$L^{CLIP+VF+S}(\theta) = L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S\left[\pi_\theta\right](s_t), \tag{23}$$

where c_1, c_2 are two hyper parameters, L^{VF} is a value function squared-error loss $(V^\theta(s_t) - \widehat{R}_t)^2$ and S represents an entropy bonus, which is conducive to force the agent not to be over-confident and therefore encourage sufficient exploration. The implementation procedure of PPO method with actor-critic frameworks is shown in Algorithm 1.

Algorithm 1 PPO in AC style with shared networks parameters

1: Input: initial shared policy and value function parameters θ_0 and user-defined hyper parameters c_1 and c_2.
2: **for** $k = 1, 2, \cdots$ **do**
3: Collect set of episodes $\mathcal{D}_k = \{\tau_i\}$ by running policy π_{θ_k} in the environment.
4: Compute rewards-to-go $\widehat{R}_t$.
5: Compute advantage A_t through GAE based on the current value function V^{θ_k}.
6: Update the policy and value function simultaneously by maximizing the PPO surrogate loss

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \{ \min(\frac{\pi(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a))) - c_1((V^{\theta_k}(s_t) - \widehat{R}_t)^2 + c_2 S\left[\pi_{\theta_k}(s_t)\right] \}$$

7: **end for**

4. Methods

4.1. GSTCs-based environment

To provide an environment interacting with the DRL agent, the FDTD method is implemented to solve the coupled system of GSTCs and Maxwell equations. In this work, we investigate the 1D problem of a 0D dynamic metasurface, which also represents the 3D problem of a uniform 2D metasurface perpendicularly illuminated by a TEM plane wave.

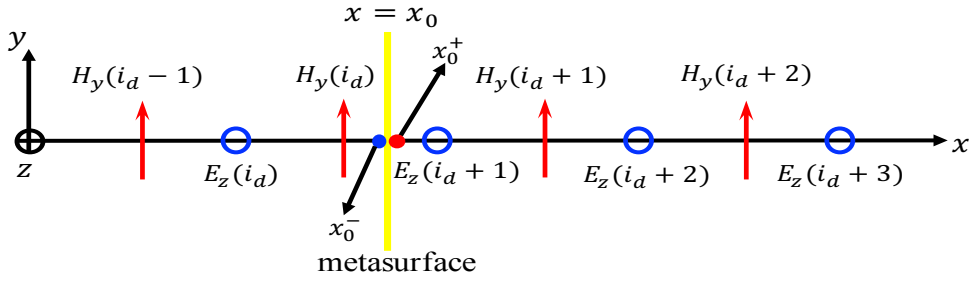


Figure 4: The 1D staggered Yee FDTD grid for the 0D metasurface analysis. A H-virtual node at $x = x_0^+$ and an E-virtual node at $x = x_0^-$ are introduced to explain the discontinuity brought by the metasurface, which is placed between two virtual nodes.

As shown in Fig.4, the TEM incident wave (with nonzero components E_z and H_y), which is launched at the original point $x = 0$, propagates in the $x-$direction, and the dynamic metasurface is placed at $x_0(> 0)$ between the i_dth cell and the i_{d+1}th cell. According to the conventional FDTD update

scheme, the electromagnetic fields are calculated iteratively via

$$\begin{cases} \mathrm{H}_y^{n+\frac{1}{2}}(i) = \mathrm{H}_y^{n-\frac{1}{2}}(i) + \dfrac{\Delta t}{\mu_0 \Delta x}[\mathrm{E}_z^n(i+1) - \mathrm{E}_z^n(i)], \\ \mathrm{E}_z^n(i) = \mathrm{E}_z^{n-1}(i) + \dfrac{\Delta t}{\varepsilon_0 \Delta x}[\mathrm{H}_y^{n-\frac{1}{2}}(i) - \mathrm{H}_y^{n-\frac{1}{2}}(i-1)]), \end{cases} \tag{24}$$

in which Δx is the grid size, Δt represents the numerical time step and the actual time $t = n\Delta t$. In the following, to ensure numerical convergence and stability, under incident wavelength λ, the grid size is set as $\Delta x = \lambda/40$ and a constant numerical time step $\Delta t = \Delta t/c$ is used, with c being the wave speed in vacuum. In addition, the Mur boundary condition [24] is imposed on the rightmost boundary.

However, Eq.(24) fails to account for the discontinuity due to the presence of a metasurface. Therefore, some special treatment must be applied to the update for $\mathrm{E}_z^n(i_d+1)$ and $\mathrm{H}_y^{n+\frac{1}{2}}(i_d)$. This is done as follows[12]:

- Consider first the update for $\mathrm{E}_z^n(i_d+1)$. The original FDTD scheme relies on the field values $\mathrm{H}_y^{n-\frac{1}{2}}(i_d)$ and $\mathrm{H}_y^{n-\frac{1}{2}}(i_d+1)$. To characterize the presence of metasurface, a H-virtual node is introduced at the right side of the metasurface $x = x_0^+$ and $\mathrm{H}_y^{n-\frac{1}{2}}(i_d)$ is replaced by $\mathrm{H}_y^{n-\frac{1}{2}}(x_0^+)$. With the help of GSTCs, $\mathrm{H}_y^{n-\frac{1}{2}}(x_0^+)$ is obtained as

$$\mathrm{H}_y^{n-\frac{1}{2}}(x_0^+) = \mathrm{H}_y^{n-\frac{1}{2}}(i_d) + \frac{\varepsilon_0}{\Delta t}[(\chi_{ee}^{zz}\mathrm{E}_{z,av})^n - (\chi_{ee}^{zz}\mathrm{E}_{z,av})^{n-1}], \tag{25}$$

  which is substituted in the original FDTD update equation to yield

$$\begin{aligned} \mathrm{E}_z^n(i_d+1)A_{ee}^{zz,n} =& \mathrm{E}_z^{n-1}(i_d+1)A_{ee}^{zz,n-1} + \frac{\Delta t}{\varepsilon_0 \Delta x}[\mathrm{H}_y^{n-\frac{1}{2}}(i_d+1) - \mathrm{H}_y^{n-\frac{1}{2}}(i_d)] \\ & - \frac{\chi_{ee}^{zz,n}}{2\Delta x}\mathrm{E}_z^n(i_d) + \frac{\chi_{ee}^{zz,n-1}}{2\Delta x}\mathrm{E}_z^{n-1}(i_d), \end{aligned} \tag{26}$$

  where $A_{ee}^{zz,n} = 1 + \dfrac{\chi_{ee}^{zz,n}}{2\Delta x}$.

- Similarly, $\mathrm{H}_y^{n+\frac{1}{2}}(i_d)$ conventionally depends on $\mathrm{E}_z^n(i_d)$ and $\mathrm{E}_z^n(i_d+1)$. As previously done, a E-virtual node is introduced at the left side of the metasurface x_0^-, with $\mathrm{E}_z^n(x_0^-)$ playing the role of $\mathrm{E}_z^n(i_d+1)$ and is expressed by invoking GSTCs as

$$\mathrm{E}_z^n(x_0^-) = \mathrm{E}_z^n(i_d+1) - \frac{\mu_0}{\Delta t}[(\chi_{mm}^{yy}\mathrm{H}_{y,av})^{n+\frac{1}{2}} - (\chi_{mm}^{yy}\mathrm{H}_{z,av})^{n-\frac{1}{2}}], \tag{27}$$

  which likewise leads to

$$\begin{aligned} \mathrm{H}_y^{n+\frac{1}{2}}(i_d)A_{mm}^{yy,n+\frac{1}{2}} =& \mathrm{H}_y^{n-\frac{1}{2}}(i_d)A_{mm}^{yy,n-\frac{1}{2}} + \frac{\Delta t}{\mu_0 \Delta x}[\mathrm{E}_z^n(i_d+1) - \mathrm{E}_z^n(i_d)] \\ & - \frac{\chi_{mm}^{yy,n+\frac{1}{2}}}{2\Delta x}\mathrm{H}_y^{n+\frac{1}{2}}(i_d+1) + \frac{\chi_{mm}^{yy,n-\frac{1}{2}}}{2\Delta x}\mathrm{E}_z^{n-\frac{1}{2}}(i_d+1), \end{aligned} \tag{28}$$

  where $A_{mm}^{yy,n} = 1 + \dfrac{\chi_{mm}^{yy,n}}{2\Delta x}$.

In the process of TEM wave propagation, the electric field and the magnetic field are closely related to each other through the impedance η as $\mathrm{E}_z = \eta \times \mathrm{H}_y$, one of which can thus be replaced by the other. Therefore, from Eq.(4), the analytical $\chi_{ee}^{zz}(t)$ and $\chi_{mm}^{yy}(t)$ are actually expressed in terms of the electric field (or only the magnetic field) only.

In order to provide environment observation and reward feedback to the DRL network, two electric detectors are placed at the electric nodes on both sides of the dynamic metasurface (i.e. nodes of $\mathrm{E}_z(i_d)$ and $\mathrm{E}_z(i_d+1)$) to record the electric field values sensed by the metasurface in a period of interest before the current time, including information about the incident wave and the scattered transmitted and reflected fields. The network input is obtained by concatenating and flattening two detector's records. The network input thus consists of detailed electric field information, which in turn influences the value of $\chi_{ee}^{zz}(t)$ and $\chi_{mm}^{yy}(t)$.

4.2. GSTCs-PPO control framework

To exploit the dynamic metasurface principle for controlling the propagation of the incident wave to satisfy any specified time evolution shape of the target scattering field, DRL is used to obtain in real time values of $\chi_{ee}^{zz}(t)$ and $\chi_{mm}^{yy}(t)$ just by observing the electric field recorded by detectors. More specifically, as schematically depicted in Fig.5, the DRL agent deploys DNN to interact with FDTD simulation through three elaborately designed channels:

- The observation s_t describes the basis for the agent to perform actions, which materializes as a vector containing electric field seen by thhe metasurface and is obtained from electric detectors;
- The action a_t must provide the possibility to achieve the desired goal, which can directly be the sought after $\chi_{ee}^{zz}(t)$ and $\chi_{mm}^{yy}(t)$;
- The instantaneous reward r_t reflects the relevance of a_t and guides the control strategy towards getting closer to exact solution. In this work, the reward has been defined as the difference between the specified desired (electric) scattering field and the (electric) scattering field simulated using a_t, which can be expressed as follow

$$r_t = -|\mathrm{E}_z^{desired-t}(t) - \mathrm{E}_z^{t}(t)| - |\mathrm{E}_z^{desired-r}(t) - \mathrm{E}_z^{r}(t)|, \tag{29}$$

  where $\mathrm{E}_z^{desired-t}(t)$ and $\mathrm{E}_z^{desired-r}(t)$ respectively represent the specified target transmitted and reflected wave, while $\mathrm{E}_z^{t}(t)$ and $\mathrm{E}_z^{r}(t)$ are the transmitted and reflected wave simulated by a_t, respectively.

In addition to the well-defined values of s_t, a_t and r_t, their preprocessing and postprocessing are critically necessary for the DRL performance. Specifically, before being fed into DNN, we scale raw s_t to $[-1, 1]$ and normalize it to mean-zero, variance-one vectors to facilitated DNN convergence. In similar way, we limit the DNN output action values a_t (e.g. a 2D vector $a_t = [a_{t1}, a_{t2}]$) into $[-1, 1]$, and further transform it into valid $\chi_{ee}^{zz}(t)$ and $\chi_{mm}^{yy}(t)$ through some nonlinear transformations as

$$\begin{cases} \chi_{ee}^{zz}(t) = s_1 \times 10^{-(\alpha_1 |a_{t1}| + \beta_1)}, \\ \chi_{ee}^{yy}(t) = s_2 \times 10^{-(\alpha_2 |a_{t2}| + \beta_2)}, \end{cases} \tag{30}$$

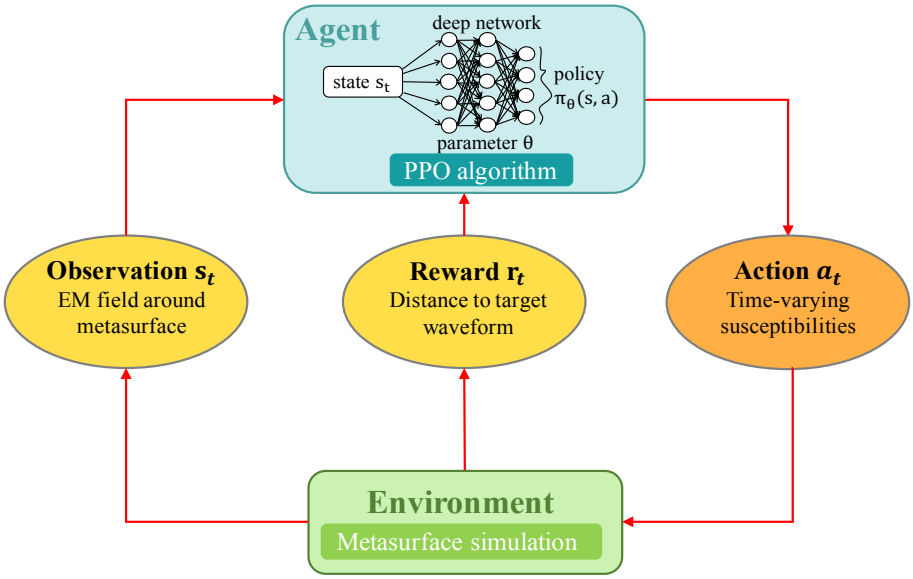


Figure 5: Illustration of the DRL closed-loop framework used in this paper for the dynamic metasurface control problem. The environment, i.e., a FDTD simulation of the metasurface, interacts with a fully connected artificial neural network iteratively through three channels, i.e., an observation s_t, an action a_t and a reward r_t . By observing the field around the metasurface (s_t), the network will output the distribution of susceptibilities and sample an action a_t (i.e., $\chi_{ee}^{zz}(t)$ and $\chi_{mm}^{yy}(t)$) from it. After simulating using a_t, the environment produces a reward r_t, based on which the network updates itself using PPO algorithm. Meanwhile, the environment evolves to the next state s_{s+1}, thus entering the next closed-loop learning stage.

in which s_1, s_2, α_1, α_2, β_1 and β_2 need to be set manually as hyperparameters. However, in fact, the above nonlinear transformations are not compulsory, but it can effectively accelerate the convergence of DRL. Moreover, we clip r_t within a preset range $[-10, 10]$, since the clipping operation helps filter out those rewards whose absolute value is too large, which can promote the agent to learn more in an unbiased way.

Based on agent-environment closed-loop interaction, the PPO algorithm trains the DNN to discover control strategies $\pi_\theta : s_t \to a_t$, so as to maximize r_t. Since the time-shift variant feature of time-varying structure, the incident wave must be synchronized with the evolution of the metasurface. In our cases, the simulation is firstly run with $\chi_{ee}^{zz}(t) = \chi_{mm}^{yy}(t) = 0$ until the incident wave reaches the metasurface, while PPO control works from the moment the incident wave reaches the metasurface to when it completely leaves the metasurface, which corresponds to an entire episode.

Remark 1. *In DRL, it is often necessary for the agent to "look far enough" to make reasonable decisions. "Looking far" ostensibly refers to the large number of steps considered forward by the agent, but in essence it indicates a broad system dynamic evolution span. Especially for the "fine grained" problem with very small numerical steps Δt considered in the present work, if the agent makes a decision every Δt, even a long episode only corresponds to limited system variation, meaning that the agent needs to interact with the environment many times to grasp the overall system characteristics. Considering that a too long episode tends to be an issue for the network training, we use the "frame skipping" technique to appropriately coarsen the decision granularity and update the action provided by DNN every several Δt, thus greatly shortening the episode length while ensuring*

sufficient control effect. In addition, we use a linear interpolation to enforce the continuity between DNN actions.

4.3. Control policies implementation

The GSTCs-PPO workflow (presented in Sec. 4.2 is implemented based on Python 3.7 scripts and relies on the Stable Baselines framework [34], an open source improved implementation of RL algorithms based on OpenAI Baselines in which the environment interface is offered by Gym library [35]. The neural network used is a simple fully-connected network consists of an input layer to receive the environment observation vector, an output layer to generate control signals for metasurface, and two hidden layers with 16 nodes, both characterizing the Relu activation function. This network architecture was found empirically and we cannot ruled out that there are other better configurations. Besides, the discount factor is set to $\gamma = 0.99$ in the training process.

5. Results

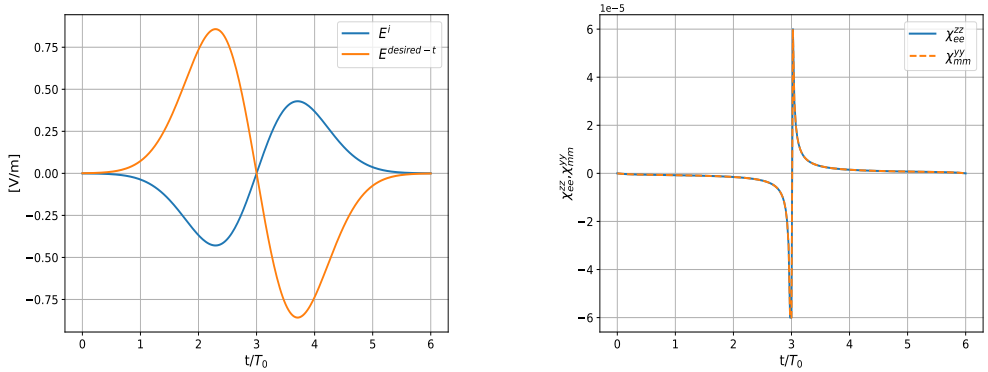
On the basis of the methodology proposed in the previous sections, we aim to demonstrate the ability of the GSTCs-PPO policies to control susceptibilities through partial observation of the electromagnetic field so that the incident wave is transformed into a specified scattering shape after passing through the metasurface. Specifically, in the considered 1D context, we introduce two test cases for a point-wise metasurface: a special case of zero-reflection metasurface with single control parameter and a general reflective-transmissive metasurface containing a two-dimensional action space. Both examples are simple enough to ensure a rapid simulation, leading to a relatively easy learning, while being sufficient to serve as a proof-of-concept to illustrate the feasibility of DRL in dynamic control of metasurface and to provide directions for future analysis and applications.

5.1. Reflection-less Metasurface

For simplicity, we add the constraint $\chi_{ee}^{zz}(t) = \chi_{mm}^{yy}(t)$, leading to a reflection-less metasurface and 1D action that the agent needs to learn. Considering a differential Gaussian pulse impinging normally on the metasurface, susceptibilities are controlled to time reverse and amplify this pulse, that is, the incident wave and the desired transmitted wave look like the shape in Fig. 6(a) in blue and orange, respectively, and are given by

$$\begin{cases} \mathrm{E}_z^i(t) = \frac{(t-3T)}{T}\exp(-\frac{(t-3T)^2}{T^2}), \\ \mathrm{E}_z^{desired-t}(t) = -2\times\frac{(t-3T)}{T}\exp(-\frac{(t-3T)^2}{T^2}), \\ \mathrm{E}_z^{desired-r}(t) = 0, \end{cases} \tag{31}$$

where $T = \dfrac{1}{f_0}$ and f_0 represent the frequency of incident wave. In numerical simulation, placing the metasurface at $x = 10\Delta x$, the incident wave reaches the metasurface at $t = 10\Delta t$ and completely leaves the metasurface at $t = 250\Delta t$. Using equation (4), the exact solution of susceptibilities are



(a) (b)

Figure 6: Zero-reflection metasurface with single control parameter. (a) Specified differential Gaussian incident pulse (blue line) and the expected time reversed and amplitude doubling transmitted pulse (orange line). (b) The corresponding exact temporal evolution of susceptibilities based on (4).

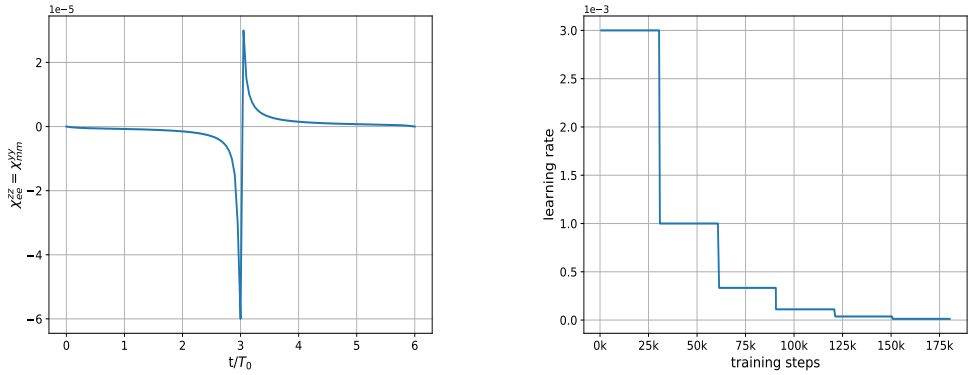
plotted in Fig. 6(b), from which we can notice that there is a singularity at $t = 130\Delta t$, which in fact corresponds to the center point of the differential pulse. At this point, the average field on the metasurface is close to zero and thus the metasurface needs to provide sufficiently large values of the susceptibilities to produce the required scattering effect.

Before training, we firstly sample the exact susceptibility solutions every $2\Delta t$ and get a new target shape as shown in Fig. 7(a), with which the desired wave transformation can also been achieved and the corresponding target episode reward is -0.965. Therefore, the time granularity of making a decision every $2\Delta t$ is reasonable, and the length of an episode is 120. During DRL training, the PPO loss clipping range is set to $\epsilon = 0.3$ and the adopted learning rate is the piecewise leaning rate depicted in Fig. 7(b). By applying $a_{t1} = a_{t2}$, $s_1 = s_2 = sign(a_{t1}) = sign(a_{t2})$ and $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 2.0$ and training for $180,00$ steps (i.e. 1500 episodes and takes about 35min) using 4 environments and 128 steps mini-batches, the action $a_{t1} = a_{t2}$ that is produced by network and the result susceptibilities $\chi_{ee}^{zz}(t) = \chi_{mm}^{yy}(t)$ are respectively shown in Fig. 8(a) and Fig.8(b). The learnt shape is perfectly in line with the target one in Fig. 7(a), based on which the simulated transmitted wave shown in Fig. 9 is completely coincident with the desired shape and the obtained episode reward -1.764 is close to the target one -0.965.

Furthermore, Fig. 10(a) illustrates the learning curve corresponding to 5 different training elements performed using different random seeds based on the same hyperparameters, where the light blue lines present individual training elements and the dark orange presents the average of five training elements. Fig. 10(b) shows the variance curve for these 5 training elements, indicating the robustness of DRL training.

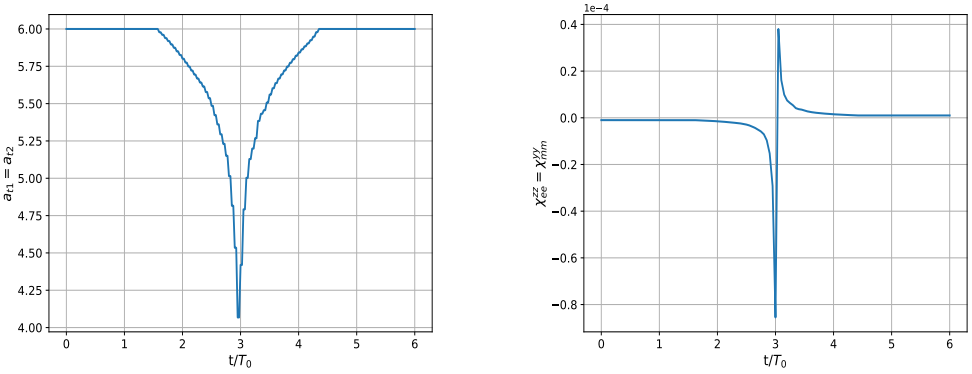
5.2. Transmittable and Reflective Metasurface

Next, we investigate a more general metasurface with both transmission and reflection functions and $\chi_{ee}^{zz}(t) \neq \chi_{mm}^{yy}(t)$. Taking Gaussian pulse as the incident wave, the susceptibilities shown



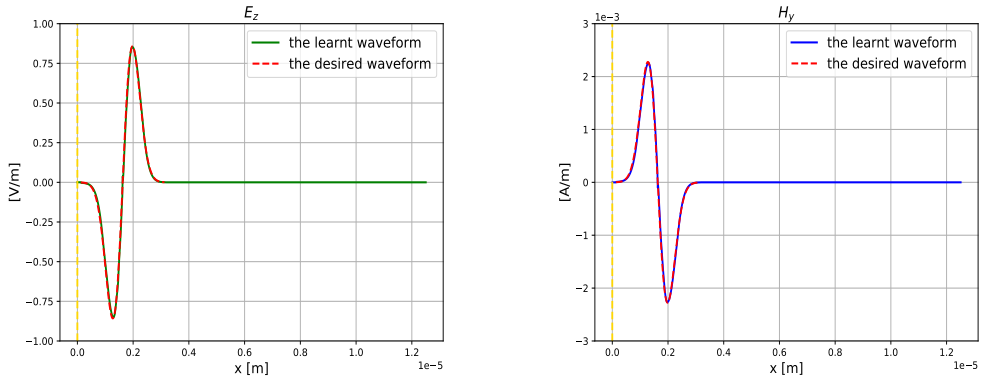
(a) (b)

Figure 7: Zero-reflection metasurface with single control parameter. (a) The target temporal evolution of susceptibilities when performing an action every $2\Delta t$. (b) The piecewise learning rate adopted by the neural network.



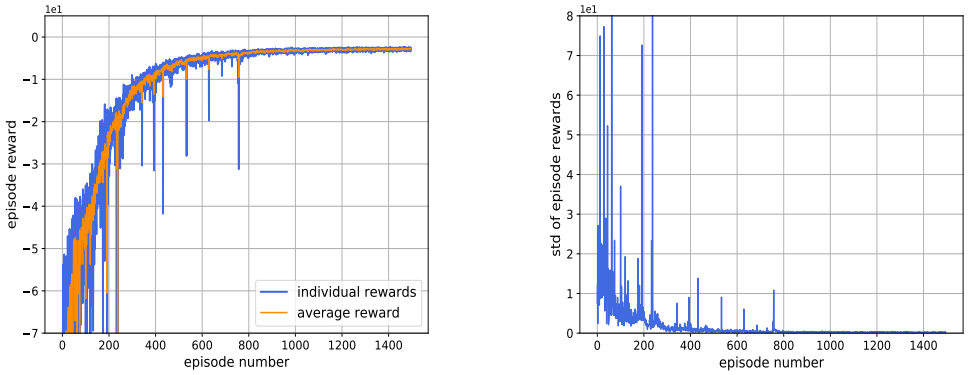
(a) (b)

Figure 8: Zero-reflection metasurface with single control parameter. Presentation of the best result obtained using deterministic prediction among 5 different individual training elements using the same hyperparameters, but different random seeds. (a) The action values $a_{t1} = a_{t2}$ output by network. (b) The result susceptibilities $\chi_{ee}^{zz}(t) = \chi_{mm}^{yy}(t)$ based on (30) with $s_1 = s_2 = sign(a_{t1}) = sign(a_{t2})$ and $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 2.0$.



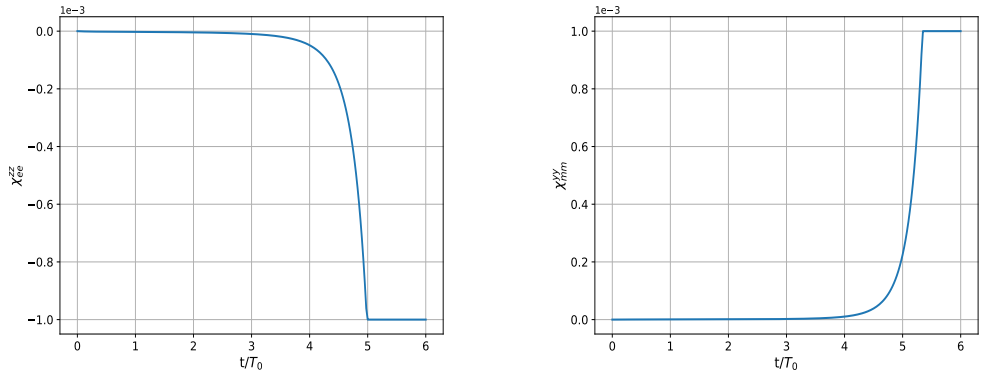
(a) (b)

Figure 9: Zero-reflection metasurface with single control parameter. Simulated FDTD fields at $t = 250\Delta t$ (when the incident wave entirely leaves the metasurface) using the learnt susceptibilities in Fig. 8(b), possessing the same shape with the desired transmitted wave. (a) The electric transmitted waveform. (b) The magnetic transmitted waveform.



(a) (b)

Figure 10: Zero-reflection metasurface with single control parameter. Episode rewards during agent learning, for 5 different individual training elements using the same hyperparameters, but different random seeds. (a) Episode rewards curves of 5 individual training elements (blue lines) and their average value curve (orange line). (b) Variance of the 5 individual episode rewards curves.



(a) (b)

Figure 11: Reflective-transmissive metasurface with two control parameters. The corresponding exact temporal evolution of susceptibilities based on (4). (a) The exact electric susceptibility χ_{ee}^{zz}. (b) The exact magnetic susceptibility χ_{mm}^{yy}.

in Fig. 11 transform the incident wave into time-reversed transmitted and reflected waves, with amplitude of 0.8 and 0.4 times the incident wave respectively. In Fig. 12, the incident pulse and the desired transmitted and reflected pulse are plotted in green, orange and blue, respectively, and are given by

$$\begin{cases} \mathrm{E}_z^i(t) = \exp(-\frac{(t-3T)^2}{T^2}), \\ \mathrm{E}_z^{desired-t}(t) = -0.8 \times \exp(-\frac{(t-3T)^2}{T^2}), \\ \mathrm{E}_z^{desired-r}(t) = -0.4 \times \exp(-\frac{(t-3T)^2}{T^2}), \end{cases} \tag{32}$$

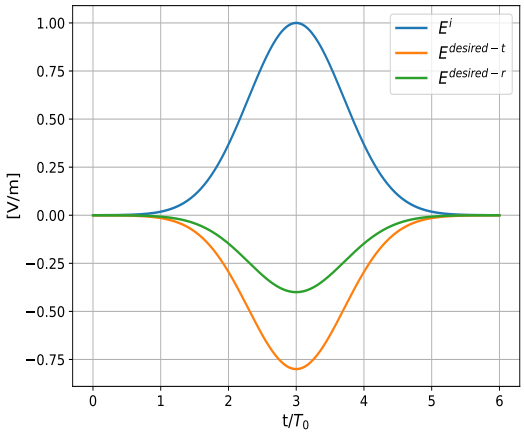


Figure 12: Reflective-transmissive metasurface with two control parameters. Specified Gaussian incident pulse (blue line) and the expected transmitted (orange line) and reflected (green line) waveform.

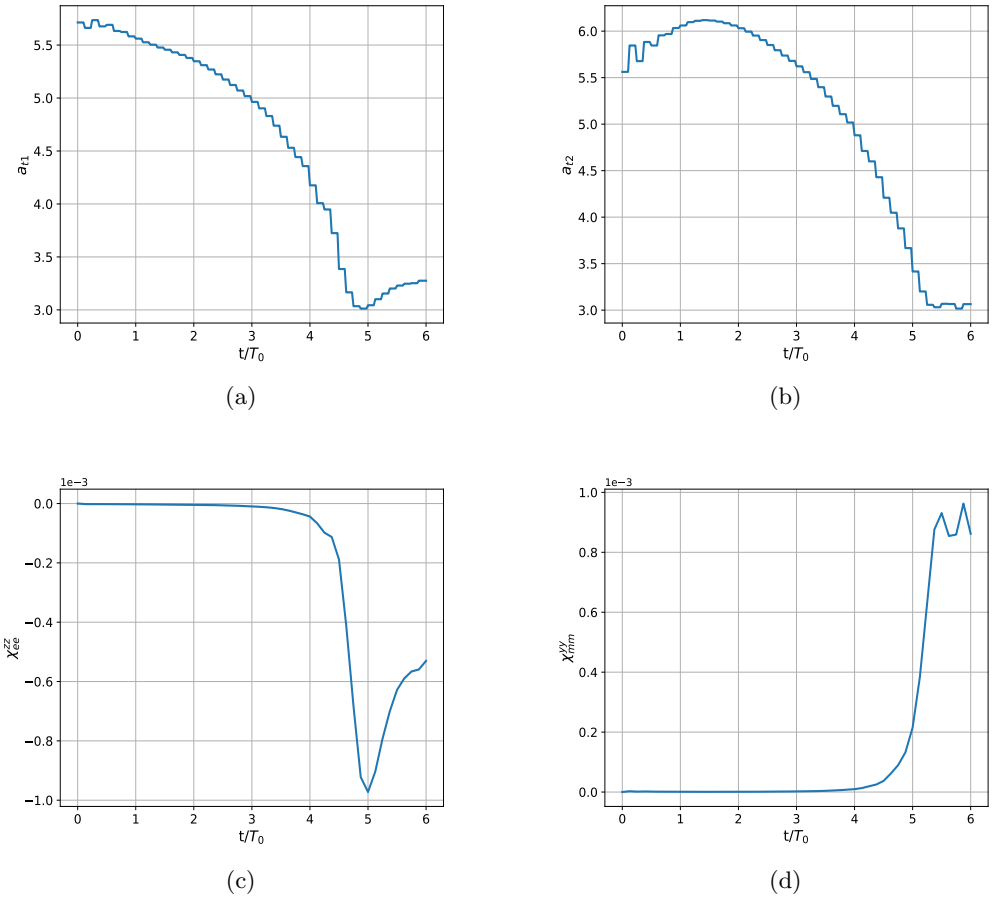
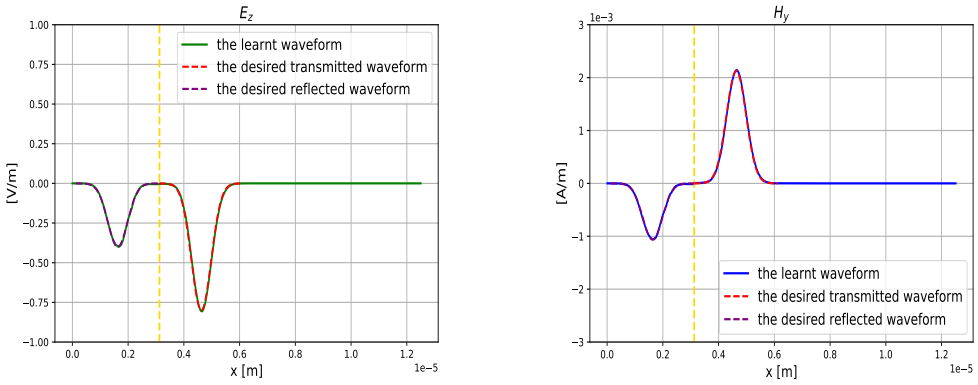


Figure 13: Reflective-transmissive metasurface with two control parameters. Presentation of the best result obtained using deterministic prediction among 5 different individual training elements using the same hyperparameters, but different random seeds. (a) The action values a_{t1} output by network. (b) The action values a_{t2} output by network (c) The result susceptibility χ^{zz}_{ee} based on (30) with $s_1 = 1.0$, $\alpha_1 = 4.0$, $\beta_1 = 3.0$. (d) The result susceptibility χ^{yy}_{mm} based on (30) with $s_2 = 1.0$, $\alpha_2 = 4.0$, $\beta_2 = 3.0$.

In DRL training, the time granularity of making a decision every $5\Delta t$ leads to an episode with a length of 48. We empirically set $s_1 = s_2 = 1.0$, $\alpha_1 = \alpha_2 = 4.0$, $\beta_1 = \beta_2 = 3.0$, the PPO loss clipping range $\epsilon = 0.2$ and the adaptive learning rate $lr = 3.0 \times 10^{-3} \times f$. Using 4 environments and 64 mini-batches, convergence is achieved after $280,000$ steps (i.e. about $5,800$ episodes), which takes about 130 minutes. The learnt actions a_{t1} and a_{t2} and the result susceptibilities $\chi^{zz}_{ee}(t)$ and $\chi^{yy}_{mm}(t)$ are shown in Fig. 13, which is slightly deviated from the exact solution, but its overall shape is reasonable. The obtained episode reward -1.764 is large enough and the corresponding simulated scattering fields coincide perfectly with the desired shape, as shown in Fig. 14, which illustrates the feasibility and effectiveness of the proposed method. Similarly, Fig. 15 shows the learning curve corresponding to 5 different training elements performed using different random seeds based on the same hyper-parameters, in which the "exploration" of the agent results in the large variance in the early training stage and the stable stage after $3,000$th episodes corresponds to the "exploitation"



(a) (b)

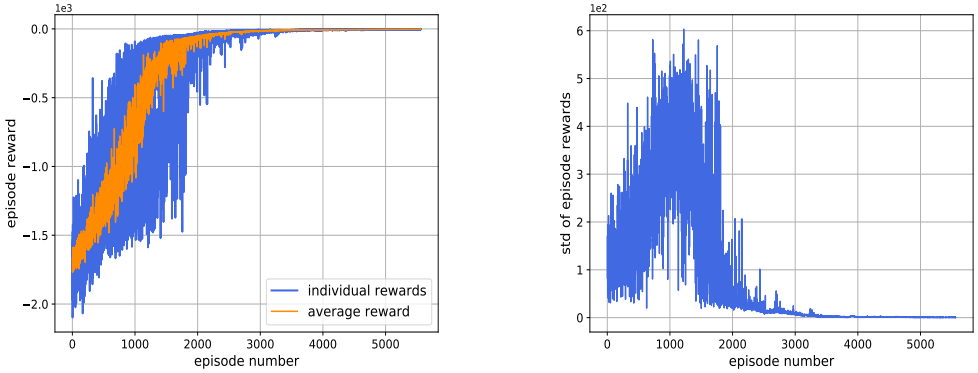
Figure 14: Reflective-transmissive metasurface with two control parameters. Simulated FDTD fields at $t = 490\Delta t$ (when the incident wave entirely leaves the metasurface) using the learnt susceptibilities in Fig. 13(c) and Fig. 13(d), possessing the same shape with the desired transmitted and reflected wave. (a) The electric transmitted and reflected waveform. (b) The magnetic transmitted and reflected waveform.

of the agent.

Two test cases with increasing level of complexity confirm the DRL ability in point-wise metasurface control. Although the training cost required is getting bigger with the complexity of the problem, interactive nature of DRL reduces the required manual work and enables more flexible and more comprehensive exploration. Besides, other technologies such as transfer learning can be used to reduce DRL training cost in similar problems.

6. Conclusions

In this work, we demonstrate for the first time a deep reinforcement learning (DRL) algorithm efficiently controlling the susceptibilities of dynamic metasurfaces. Based on the introduction of the generalized sheet transition conditions (GSTCs) and the basic principles of DRL, our control problem is translated into a sequential decision problem and a GSTCs-PPO framework is proposed by taking advantages of FDTD numerical simulations and the proximal policy optimization (PPO) algorithm. Using this framework, the metasurface can adaptively adjust itself to achieve specific wave control by solely using partial field observation near the metasurface discontinuity. Details on the algorithm implementation and two 1D results illustrate the feasibility and applicability of the proposed approach, which can discover efficient control strategies with limited time. Departing from supervised learning, reinforcement learning is an evaluation-based and interaction-based machine learning method, in which the agent decides its action in terms of the observed environment and optimizes its cognition in the process of maximizing the received reward values. Learning does not rely on any pre-prepared training data and is agnostic to the underlying physical knowledge, hence to some extent it is a truly intelligent automated technique that does not require human assistance. In addition, although our results are completely acceptable, it is worth noting that our work pays more attention to present the application potential of DRL, rather than focusing



(a) (b)

Figure 15: Reflective-transmissive metasurface with two control parameters. Episode rewards during agent learning, for 5 different individual training elements using the same hyperparameters, but different random seeds. (a) Episode rewards curves of 5 individual training elements (blue lines) and their average value curve (orange line). (b) Variance of the 5 individual episode rewards curves.

on the optimization of hyperparameters and training details, thus letting many aspects worthy of improvement to further optimize the results.

Our results provide a new perspective for the inverse problem of electromagnetic wave control, making it more environmentally adaptive and automated. Although discussed in simple 1D problems, it is sufficient to expose the high potential DRL to reliably perform electromagnetic wave control. Furthermore, environment parallelization enables DRL to scale well to large amounts of data and solve more realistic two-dimensional problems, and the transfer learning allows DRL to be used to solve similar new problems. Therefore, our future work will aim at examining how DRL solves the control of adaptive metasurfaces in a 2D context.

References

[1] C. Holloway, E. Kuester, J. Gordon, J. O'Hara, J. Booth, D. Smith, An overview of the theory and applications of metasurfaces: The two-dimensional equivalents of metamaterials, Antennas and Propagation Magazine, IEEE 54 (2012) 10–35. doi:10.1109/MAP.2012.6230714.

[2] A. Pors, M. Nielsen, R. Eriksen, S. Bozhevolnyi, Broadband focusing flat mirrors based on plasmonic gradient metasurfaces, Nano Letters 13 (2013) 829–834.

[3] G. Zheng, H. MÜhlenbernd, M. Kenney, G. Li, T. Zentgraf, S. Zhang, Metasurface holograms reaching 80% efficiency, Nature Nanotechnology 10 (2015) 308–312.

[4] K. Achouri, C. Caloz, Metasurface solar sail for flexible radiation pressure control, https://arxiv.org/abs/1710.02837, 2017.

[5] K. Achouri, G. Lavigne, M. A. Salem, C. Caloz, Metasurface spatial processor for electromagnetic remote control, IEEE Transactions on Antennas and Propagation 64 (2016) 1759–1767.

[6] S. Zhang, Ai empowered metasurfaces, Light: Science & Applications 9 (2020) 94.

[7] N. Chamanara, Y. Vahabzadeh, C. Caloz, Simultaneous control of the spatial and temporal spectra of light with space-time varying metasurfaces, IEEE Transactions on Antennas and Propagation 67 (2018) 2430–2441.

[8] A. Shaltout, A. Kildishev, V. Shalaev, Time-varying metasurfaces and lorentz non-reciprocity, Optical Materials Express 5 (2015).

[9] J. Zhang, X. Wei, I. Rukhlenko, H.-T. Chen, W. Zhu, Electrically tunable metasurface with independent frequency and amplitude modulations, ACS Photonics (2019) 265–271. doi:10.1021/acsphotonics.9b01532.

[10] K. Achouri, M. A. Salem, C. Caloz, General metasurface synthesis based on susceptibility tensors, IEEE Transactions on Antennas and Propagation 63 (2014) 2977–2991.

[11] Y. Vahabzadeh, N. Chamanara, K. Achouri, C. Caloz, Computational analysis of metasurfaces, IEEE Journal on Multiscale and Multiphysics Computational Techniques 3 (2017) 37–49.

[12] Y. Vahabzadeh, N. Chamanara, C. Caloz, Generalized sheet transition condition fdtd simulation of metasurface, IEEE Transactions on Antennas and Propagation 66 (2017) 271–280.

[13] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[14] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[15] C. Qian, B. Zheng, Y. Shen, L. Jing, E. Li, L. Shen, H. Chen, Deep-learning-enabled self-adaptive microwave cloak without human intervention, Nature Photonics 14 (2020) 383–290. doi:10.1038/s41566-020-0604-2.

[16] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, et al., Mastering the game of go without human knowledge, Nature 550 (2017) 354–359.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, Computer Science (2013).

[19] Z. Shen, Y. Wang, D. Wu, X. Yang, B. Dong, Learning to scan: A deep reinforcement learning approach for personalized scanning in ct imaging, 2021. arXiv:2006.02420.

[20] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control, Journal of Fluid Mechanics 865 (2019) 281–302. doi:10.1017/jfm.2019.62.

[21] J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, E. Hachem, Direct shape optimization through deep reinforcement learning, 2020. arXiv:1908.09885.

[22] M. A. Bucci, O. Semeraro, A. Allauzen, G. Wisniewski, L. Cordier, L. Mathelin, Control of chaotic systems by deep reinforcement learning, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 475 (2019) 20190351. URL: http://dx.doi.org/10.1098/rspa.2019.0351. doi:10.1098/rspa.2019.0351.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, https://arxiv.org/abs/1707.06347, 2017.

[24] B. Archambeault, O. M. Ramahi, C. Brench, The Finite-Difference Time-Domain Method, Springer US, Boston, MA, 1998, pp. 35–67. doi:10.1007/978-1-4757-5124-6_3.

[25] S. Ruder, An overview of gradient descent optimization algorithms, 2017. arXiv:1609.04747.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, et al., Human-level control through deep reinforcement learning, Nature 518 (2012) 529–33.

[27] R. S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Proceedings of the 12th International Conference on Neural Information Processing Systems, NIPS'99, MIT Press, Cambridge, MA, USA, 1999, p. 1057–1063.

[28] I. Grondman, L. Busoniu, G. A. D. Lopes, R. Babuska, A survey of actor-critic reinforcement learning: Standard and natural policy gradients, IEEE Transactions on Systems Man & Cybernetics Part C 42 (2012) 1291–1307.

[29] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, E. Hachem, A review on deep reinforcement learning for fluid mechanics, 2021. arXiv:1908.04127.

[30] J. Achiam, Spinning up in deep reinforcement learning, https://openai.com/blog/spinning-up-in-deep-rl/, 2018.

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, p. 1928–1937.

[32] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, 2018. arXiv:1506.02438.

[33] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel, Trust region policy optimization, 2017. arXiv:1502.05477.

[34] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, Stable baselines, https://github.com/hill-a/stable-baselines, 2018.

[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016. arXiv:arXiv:1606.01540.