Sim-to-real transfer reinforcement learning for control of thermal effects of an atmospheric pressure plasma jet

Matthew Witman[1,2]‡, **Dogan Gidon**[1]‡, **David B. Graves**[1], **Berend Smit**[1,2], **Ali Mesbah**[1]

[1] Department of Chemical and Biomolecular Engineering, University of California, Berkeley 94720, United States

[2] Laboratory of Molecular Simulation (LSMO), Institut des Sciences et Ingénierie Chimiques, Valais, Ecole Polytechnique Fédérale de Lausanne (EPFL), Rue de l'Industrie 17, CH-1951 Sion, Switzerland

E-mail: `mesbah@berkeley.edu`

Abstract. Applications of atmospheric pressure plasma jets (APPJs) present challenging feedback control problems due to the complexity of the plasma-substrate interactions. The plasma treatment of complex substrates is particularly sensitive to changes in the physical, chemical, and electrical properties of the substrate, which may vary considerably within and between target substrates. The increasingly popular reinforcement learning (RL) methods hold promise for learning-based control of APPJ applications that involve treatment of complex substrates with time-varying or non-uniform characteristics. This paper demonstrates the use of a deep RL method for regulation of thermal properties of APPJs on substrates with different thermal and electrical characteristics. Using simulated data from an experimentally-validated, physics-based model of the thermal dynamics of the plasma-substrate interactions, an RL agent is trained to perform temperature setpoint tracking. It is shown that training the RL agent using a wide range of simulated thermal dynamics of the plasma-substrate interactions allows for capturing the diverse temperature responses of different substrates. Experimental demonstrations on a kHz-excited APPJ in He show that the proposed RL agent enables effective temperature control over a wide variety of substrates with drastically different thermal and electrical properties.

1. Introduction

Atmospheric pressure plasma jets (APPJs) are unique tools for the treatment of many different types of substrates for a wide variety of purposes [1]. Due to their ability to locally generate and deliver thermal, chemical, and electrical effects to substrates [2], APPJs have found wide use in materials processing for polymerization [3], etching and surface activation [4], as well as in medicine for the promotion of wound healing, blood coagulation, disinfection of infected tissue, and tumor shrinking [5, 6, 7, 8]. However,

‡ Both authors contributed equally.

reliable and effective operation of APPJs can be challenging, particularly over complex substrates. APPJ treatments generally require precise control of the integral, nonlinear nature of (possibly multiple) plasma effects on a substrate. However, APPJs exhibit nonlinear and multivariable dynamics, which are sensitive to exogenous disturbances (e.g., changes in jet tip-to-substrate separation distance) and run-to-run variabilities [9, 10, 11, 12, 13, 14]. Recently, advanced model-based control strategies have been employed to address these challenges [15, 16, 17, 18, 19].

A significant challenge in feedback control of APPJs is variations of the plasma and substrate dynamics that can result from changes in plasma properties (e.g., due to mode transitions, run-to-run variabilities) and time-varying or non-uniform characteristics of complex substrates. For example, the α-γ transition in RF APPJs can lead to significant variability in the APPJ dynamics [20] and therefore the delivered plasma effects to a substrate. Variations in properties of a substrate can also drastically affect the plasma dynamics. For example, plasma tends to spread over dielectric substrates due to charge accumulation on the surface. On the other hand, the so-called "discharge re-strike" phenomena observed on conductive substrates can result in a significantly higher power deposition in the plasma, along with considerable changes in electric fields and species densities delivered to the substrate [9, 10]. Moreover, complex substrates with heterogeneous and time-varying characteristics often arise in materials processing and medical applications. For example, healthy tissue and wounds generally have different thermal and electrical properties [21]. Skin electrical conductivity can change from patient to patient, or even from point to point in the same patient. Drying of the skin during treatment or coagulation of blood during treatment for wound healing can lead to time-varying dynamics [22, 23]. These characteristics of plasma-substrate interactions pose significant challenges to even the most advanced process control schemes such as model predictive control [15, 16].

Learning-based control holds promise for addressing the challenges in feedback control of APPJs, especially in applications with time-varying dynamics [24]. In this paper, we focus on deep reinforcement learning (RL), an emerging area of research in machine learning. In deep RL, an artificial neural network (ANN) is trained to take optimal actions to maximize a reward (or minimize a penalty) through continuous feedback during training [25]. RL has found remarkable success in a range of applications including complex gameplay, robotic control, and autonomous navigation [26, 27, 28, 29, 30, 31, 32, 33]. The goal of this paper is to present a proof-of-concept demonstration of deep RL for controlling the thermal dynamics of APPJ-treated substrates with drastically different electrical and thermal properties. Deep RL agents are developed and tested on a kHz-excited APPJ in He for controlling the substrate temperature of glass, aluminum, and polyimide substrates.

A key practice in RL is to use simulated data to train an RL agent such that it is capable of performing the same task in a real environment, otherwise known as *sim-to-real transfer learning.* Training the RL agent based on simulated data has multiple advantages. Firstly, the RL agent's exploration during the learning process can raise safety concerns if deployed in a live environment (e.g., potential thermal damage to the substrate). Secondly, deep RL algorithms generally have high sample complexity (i.e., require a large amount of training data), precluding their training with live data collection, which may be many orders of magnitude slower to generate than simulated data. Finally, simulations permit the generation of data under conditions that may be impractical to probe experimentally. These challenges make the sim-to-real transfer of RL agents an exciting prospect provided the model on which the agent is trained can adequately describe the real system dynamics.

Nonetheless, transferring the simulated performance of RL agents to a real-world environment can be challenging given the "reality gap", i.e., the mismatch between system dynamics in the simulated and real-world environments. Building a high-fidelity model or more complex simulators that can capture every aspect of a real environment is often prohibitively expensive or may even be impractical. Therefore, methods that enrich the data sets on which RL agents are trained have been proposed to address the reality gap [34, 35, 36]. These methods are generally intended to account for uncertain and difficult-to-model aspects of the real environments to enable successful sim-to-real transfer of RL agents using limited experimental data. Here, we use *dynamics randomization* [37] to allow for training RL agents for controlling the thermal effects of the APPJ using a relatively simple model of the thermal dynamics of the plasma-substrate interactions. Dynamics randomization involves randomizing the model parameters during the training process to account for the mismatch between the simulated and real environments (i.e., unmodeled dynamics and noise) as well as the different dynamics the RL agents may encounter during the plasma treatment (i.e., different substrates).

In this work, we use data generated from a physics-based model of the substrate thermal dynamics to train three RL agents based on three different degrees of dynamics randomization of the model parameters. It is shown that enriching the training data by using numerous realizations of model parameters, along with accounting for measurement noise, allow for training RL agents that can more effectively handle the thermal dynamics of different substrates and operational variabilities of the APPJ. We first test the RL agents in simulations to asses their performance in temperature setpoint tracking for a broad range of model parameters pertaining to thermal and electrical properties of a target substrate. We then experimentally compare the ability of the RL agents to track temperature setpoints on glass, aluminum, and polyimide substrates. We further demonstrate that the best performing RL agent is also capable of rejecting step disturbances in the jet tip-to-substrate separation distance.

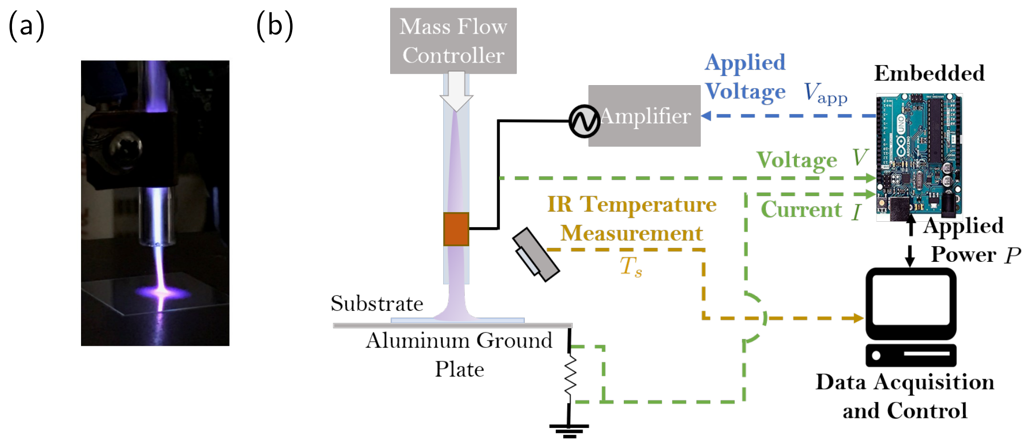

Figure 1: (a) Visual appearance of the plasma, and (b) schematic of the APPJ setup used for testing of the RL agents and validating the model from which the training data is generated. The applied voltage V_{app} is manipulated via an embedded proportional-integral controller to control the plasma power P. The embedded controller uses real-time measurements of voltage V and current I. The substrate temperature T_s is measured using an infra-red thermal camera.

2. Experimental setup

2.1. Atmospheric pressure plasma jet

The kHz-excited APPJ in He used for testing the RL agents and validating the model from which the training data is generated is shown in Figure 1 and has been described in detail elsewhere [15, 16]. The APPJ consists of a quartz dielectric tube (inside diameter of 3 mm and outside diameter of 4 mm) and a powered copper ring electrode placed 1 cm from the tube nozzle. An aluminum plate acts as the ground. The target substrates are placed on the aluminum plate under the APPJ for treatment. A helium flow of 1.5 slm is maintained in the tube via a mass flow controller. The APPJ is ignited with AC high voltage, generated by a custom designed function generator (XR-2206CP) at a frequency of 20 kHz.

The applied discharge power P is maintained by an embedded proportional-integral (PI) controller that manipulates the applied voltage based on analog measurements of voltage and current. The PI controller is implemented with a sampling time of 20 ms on an Arduino UNO microcontroller. The substrate temperature at the plasma incident point is measured via a radiometric infra-red thermal camera (Lepton FLIR 3.5) pointed to the substrate. The APPJ inputs and measured outputs are coordinated via a Wi-Fi enabled single board controller (Raspberry Pi 3). The measurement sampling time is fixed at 1.3 s. The RL agents are implemented on a remote computer, where input and output information is exchanged via TCP/IP protocol over Wi-Fi.

Table 1: The normalized cooling and heating rate parameters of the physics-based model (1) for the substrates borosilicate glass, aluminum plate, and polyimide tape as estimated from experimental data.

Substrate	μ_1	μ_2
Glass	2.39	0.82
Aluminum	1.17	0.71
Polyimide	3.71	1.92

2.2. Physics-based model of thermal dynamics of substrate

The complexity of plasma-substrate interactions and disparity of the timescales of physical phenomena generally make physics-based modeling of APPJs challenging. However, under fairly standard assumptions, the thermal response of substrates to plasma treatment can be described by physics-based models based on volume-averaged mass and energy balances [15]. Here, we conduct a volume-averaged energy balance on the substrate to describe the dynamics of the substrate temperature, T_s, at the plasma incident point

$$\frac{dT_s}{dt} = \frac{1}{\rho c_p A_c d}\big(\bar{\mu}_2 \eta P - \bar{\mu}_1 2\pi r d k (T_s - T_\infty)\big). \tag{1}$$

Here, ρ, c_p, and k are the density, heat capacity, and thermal conductivity of borosilicate glass, which is the nominal substrate; r is the internal radius of the APPJ tube; $A_c = \pi r^2$ is the cross-sectional area of the APPJ tube; d is the depth of the averaged substrate volume; η represents the fraction of power dissipated on the substrate P; and T_∞ is the ambient temperature. $\bar{\mu}_1$ and $\bar{\mu}_2$ are parameters that scale the cooling and heating rate of the substrate respectively. We fit the values $\bar{\mu}_1$ and $\bar{\mu}_2$ based on experimental data collected over three substrates with different electrical and thermal dynamics: bare borosilicate glass coverslip, grounded aluminum plate, and borosilicate glass coverslip with polyimide tape on the surface. Table 1 lists the estimated parameters $\bar{\mu}_1$ and $\bar{\mu}_2$ normalized to the order of one; note that the normalized parameters are denoted by μ_1 and μ_2. The values of the model parameters and the normalization factors are presented in Appendix A. The physics-based model (1) is used to generate *in silico* data for training and testing of the RL agents.

3. Design of reinforcement learning agents

We aim to design RL agents that dictate the APPJ input power P to track user-specified setpoints for the substrate temperature under the range of thermal dynamics spanned by glass, aluminum, and polyimide substrates in the presence of inherent variabilities of the APPJ. We refer to an agent that has been trained for this purpose as an RL controller (RLC). As shown in Figure 2, RL generally relies on four key steps to train an agent: (i) generating simulated data and/or collecting live data for training, (ii) evaluating the reward for the training data based on a user-specified control objective, (iii) fitting a model to allow evaluation of the agent's performance, i.e., to provide reinforcement, and (iv) updating the RL agent to improve its performance based on the feedback. The following sections review basic RL concepts and discuss the training procedure of three RLCs for the APPJ thermal treatment considered in this paper. More details on the RLC training can be found in Appendix B.

3.1. States, actions, and rewards

Designing an RL agent requires defining the state of the system at each time step t, s_t, the action available to the agent, a_t, as well as the reward signal, r_t, used to evaluate the RL agent's performance. Considering the discrete-time version of the model (1), we choose the state for the RL agent based on the history of the system as follows

$$s_t = \{T_{s,t-i} - T_s^{sp}, P_{t-1-i} \mid i \in [0 \,..\, m]\}\,, \tag{2}$$

where T_s^{sp} is the user-specified temperature setpoint. Thus, the state $s_t \in \mathbb{R}^{2(m+1)}$ consists of temperature deviations from the setpoint and applied power for the past m time steps. Here, we choose $m = 3$. Accounting for the history of the treatment in the state allows the RLC to implicitly build its own model of process dynamics [39]. The control action at the current time step, $a_t \in \mathbb{R}$, consists only of the applied power P

$$a_t = \{P_t\}\,. \tag{3}$$

Note here that P is constrained by the limitations of the experimental setup. Hence, actions dictated by the RLC falling outside the bounds $1.1\ W < P_t < 5.0\ W$ are saturated at the limits. The bounds on P introduce nonlinearity, which should be captured in the training data.

The reward function aims to quantify the control objective. In this case, the reward

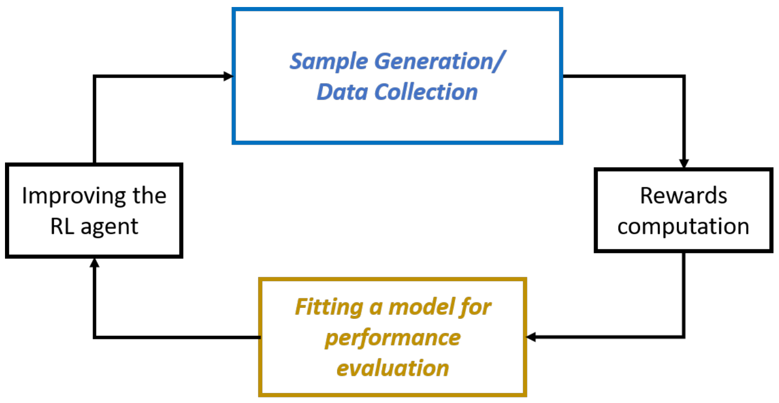
Sample Generation/ Data Collection
Rewards computation
Fitting a model for performance evaluation
Improving the RL agent

Figure 2: A simplified block diagram describing the key components of reinforcement learning [38].

function is designed to ensure the RLC maintains the temperature at the setpoint

$$r_t(s_t, a_t) = \begin{cases} 10 & T_{s,t+1} - T_s^{sp} \leq \epsilon \\ -|T_{s,t+1} - T_s^{sp}| & \text{otherwise.} \end{cases} \tag{4}$$

Here, ϵ denotes a tolerance level that corresponds to a setpoint tracking offset deemed to be as good as perfect setpoint tracking. This reward function is designed to account for the fact that, in practice, process and measurement noise may prevent perfect setpoint tracking. Therefore, the inclusion of the tolerance ϵ helps the RL agent learn a policy that tracks the setpoint more effectively. The value of $\epsilon = 0.1$ is chosen based on the observed noise level in the measurements, but it can be modified based on the application requirements.

3.2. Reinforcement learning control

The RLC takes the mathematical form of what is known as a control policy. This control policy, $\pi(a|s)$, is a probability distribution of the applied power (i.e., the action space) conditioned on the history of applied power and temperature deviations from setpoint (i.e., state). In other words, querying the control policy by feeding s_t returns the parameters of a conditional distribution from which the action a_t to be taken is sampled. While the policy specifies how the RLC takes actions, in general, a state transition model $p(s_{t+1}|s_t, a_t)$ provides a probabilistic description of how the system's state evolves in time in response to the control actions. The transition model is determined by the system dynamics, as modeled by the substrate dynamics given by (1). Nominally, this model generates deterministic transitions. Note that model (1) is simplistic and may deviate significantly from the true dynamics of the system, $p^*(s_{t+1}|s_t, a_t)$.

3.3. Goal of reinforcement learning control

Using the control policy and the state transition model, we can propagate the system state forward in time to generate simulated trajectories or "roll-outs" of time length T, denoted by a sequence of state-action pairs $\tau = (s_0, a_0, s_1, ..., a_{T-1}, s_T)$. Therefore, the probability of generating a given roll-out

$$p(\tau|\pi) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \tag{5}$$

depends on *both* the system dynamics and the RLC policy. The objective during the learning procedure is to find some optimal policy π^{opt} that maximizes the expected return of the agent, $J(\pi)$,

$$\pi^{opt} = \arg\max_{\pi} J(\pi) \tag{6}$$

with

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} r_t(s_t, a_t) \right]. \tag{7}$$

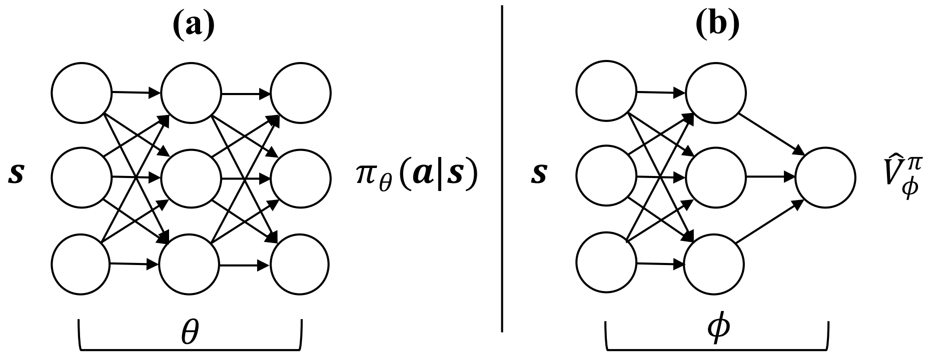

Figure 3: Generic schematic of (a) an actor policy and (b) a critic network. The state is the input to each network. The actor takes the form of a control policy, $\pi_\theta(a|s)$, parameterized by weights θ. The critic approximates the value function for the policy, $\hat{V}_\phi^\pi$, parameterized by weights ϕ.

where $\mathbb{E}$ is the expectation operator. In other words, we seek to maximize the expectation value of the accumulated rewards over the roll-outs sampled from the distribution of trajectories generated by the model (1) and the control policy, $\tau \sim p(\tau|\pi)$. The expectation of $J(\pi)$ can be computed by sampling trajectories of the state dynamics by Monte Carlo simulations.

3.4. Actor-critic algorithm

In this work, we use a variant of policy gradient RL [38, 40], along with dynamics randomization for sample generation [37] to achieve effective substrate temperature control across a range of substrate dynamics (see next section). Specifically, we use the actor-critic algorithm that relies on using two ANNs termed the *actor* and *critic* networks that are trained in tandem; as shown in Figure 3 [28, 41]. The actor neural network, parameterized by weights θ, defines the control policy $\pi_\theta(a_t|s_t)$. Since the action space is one-dimensional in this application (only the applied power P), the actor outputs the parameters of a Gaussian distribution, which is then sampled to yield an action. On the other hand, the critic network is parameterized by weights ϕ, and is used to approximate the value function, $\hat{V}_\phi^\pi(s_t)$. This value function is the expected value of future rewards at the current state s_t under the current control policy. That is, $\hat{V}_\phi^\pi(s)$ indicates how valuable it is to be at a given state under π based on the reward function. During training, $\hat{V}_\phi^\pi(s)$ is used to approximate the gradient of the reinforcement learning objective $\nabla_\theta J(\pi)$.

The weights θ and ϕ are trained to maximize the RL objective (7) by the actor-critic algorithm, the details of which are presented in Appendix B. Briefly, the weights of the actor and the critic are both initialized randomly at the beginning of training. Thus, both the actor and critic may commence with poor performance and try to improve, respectively, in approximating the optimal control policy and the value function. The actions computed by the actor are initially sub-optimal actions as the actor "explores" the system dynamics. During each training iteration, simulated roll-outs are collected (see next section) and rewarded. A gradient step is then taken to update the actor and critic network weights to improve their performance in their respective tasks. As the training proceeds, the variance of the distribution of the actions shrinks as the control policy is optimized toward accumulating higher rewards. In this work, deep ANNs are used for both the actor and the critic, with two hidden layers and 64 nodes per layer. The algorithm is implemented using Tensorflow [42] and the scripts are provided in the repository at [43].

3.5. Simulated data generation for sim-to-real transfer with dynamics randomization

Each training iteration (or epoch) requires the generation of a new batch of simulated data from the current policy, after which the gradient of (7) is computed and used to update the weights of the actor network (see Appendix B for details). Each training epoch consists of collecting N simulated roll-outs of time length T from the current policy, i.e., sampling trajectories according to (5). This is achieved using Monte Carlo sampling where the transition dynamics are given by the physics-based model (1). Here, we use the nominal values of $N = 100$ roll-outs and $T = 100$ time steps per roll-out for generating the training data at each epoch. At the beginning of each roll-out, a new setpoint is randomly chosen from the uniform distribution, $T_{sp} \sim \mathcal{U}(34, 46)$. Each roll-out then consists of the response of the system model to the actor policy for a given setpoint change.

The parameters μ_1 and μ_2 of the physics-based model (1) must also be specified to generate the training roll-outs. These model parameters are estimated from experimental data for every target substrate (see Table 1). However, the model merely provides an approximation of the thermal dynamics of the different substrates. This makes training with dynamics randomization of the parameters μ_1 and μ_2 critical. Using the dynamics randomization method [37], the RL objective is now to maximize the expected rewards when the model parameters are no longer constant, but sampled from a parameter distribution ρ_μ

$$J(\pi) = \mathbb{E}_{\mu \sim \rho_\mu} \left[\mathbb{E}_{\tau \sim p(\tau|\pi,\mu)} \left[\sum_{t=0}^{T-1} r_t(s_t, a_t) \right] \right]. \tag{8}$$

Now, at the beginning of each training roll-out, we sample $\mu_1 \sim \rho_{\mu_1}$ and $\mu_2 \sim \rho_{\mu_2}$ such that the transition dynamics, $p(\tau|\pi, \mu)$, are different in each roll-out. This allows development of a single RL agent for controlling the temperature of the substrate across the three different substrates considered in this work. Here, we choose ρ_μ such that it represents the range of thermal dynamics of the different substrates. In addition to the dynamics randomization, the substrate temperature predictions of the physics-based model are "corrupted" by white noise with standard deviation $\sigma_T = 0.1$ to account for the actual measurement noise of the experimental data. We train 3 RLCs in this work

Table 2: The parameter sampling scheme used to train the Glass RLC (G-RLC), Glass Uncertainty RLC (GU-RLC), and Ensemble RLC (E-RLC).

	μ_1	μ_2	σ_T
G-RLC	2.38	0.82	0.0
GU-RLC	$\mathcal{N}(2.38, 0.2)$	$\mathcal{N}(0.82, 0.05)$	0.11
E-RLC	$(\mu_1, \mu_2) \sim \{(0.80, 0.40), (0.80, 0.55), (0.80, 0.65), (1.12, 0.56), (1.12, 0.71), (1.12, 0.86), (2.39, 0.82), (2.39, 1.02), (2.39, 1.22), (4.00, 1.50), (4.00, 1.80), (4.00, 2.20)\}$		0.11

based on three different choices of ρ_μ, as summarized in Table 2. The Glass RLC (G-RLC) is trained only using a single set of parameters that correspond to the values fitted for the borosilicate glass substrate and no measurement noise. This is the base case with no dynamics randomization. The Glass Uncertainty RLC (GU-RLC) is trained using data generated based on a normal distribution of the parameters centered around those fitted for the borosilicate glass substrate. The measurement noise is added to the model predictions in this case. Finally, the Ensemble RLC (E-RLC) is trained by discrete uniform sampling from an ensemble of 12 sets of (μ_1, μ_2), which are chosen to be a representative selection of possible substrate dynamics, in the presence of measurement noise. In this case, dynamics randomization allows us to explicitly define the range of temperature dynamics that the RLC is expected to handle. The substrate temperature responses for various parameter values sampled from ρ_μ during the training of the E-RLC are shown in Figure 4. Note that the input power values of 1.1 W and 5 W used in Figure 4 correspond to the minimum power at which the plasma remains ignited and the maximum power achievable given the constraints of the APPJ.

3.6. Performance evaluation

To quantitatively evaluate the performance of the three RLCs, we define the mean absolute error in setpoint tracking

$$\mathrm{MAE} = 1/(T-b)\sum_{t>b}^{T} |T_{s,t} - T_s^{sp}| \tag{9}$$

and the cumulative input change as

$$\mathrm{CI} = \sum_{t>b}^{T} |P_t - P_{t-1}|, \tag{10}$$

which quantifies the control effort. Note that a minimal control effort corresponds to better control. We choose $b = 60$, neglecting the first 60 time steps to make the quantitative performance comparisons independent of the initial state of the APPJ. This allows minimizing the effects of unavoidable day-to-day experimental variations in ambient temperature. A high-performance plasma treatment corresponds to a low MAE and low CI.

4. Results: *In silico* control performance

The goal of the sim-to-real transfer learning strategy with dynamics randomization is to allow the RLC to generalize its control performance to a large variety of target substrates. Before validating our approach in real-time experiments, we quantify the performance of the three RLCs in a simulation environment using the physics-based model.

4.1. Baseline setpoint tracking

We first test the G-RLC *in silico* on the model of the borosilicate glass substrate in the absence of measurement noise via the closed-loop setpoint tracking simulations shown in Figure 5. Since there is no noise and there is no mismatch between the substrate temperature dynamics and the data generated to train the G-RLC, the control performance is excellent as expected. Thus, this study provides a baseline for the performance of the different RLCs, with the performance of G-RLC quantified by a MAE = 0.16 and a CI = 69.2.

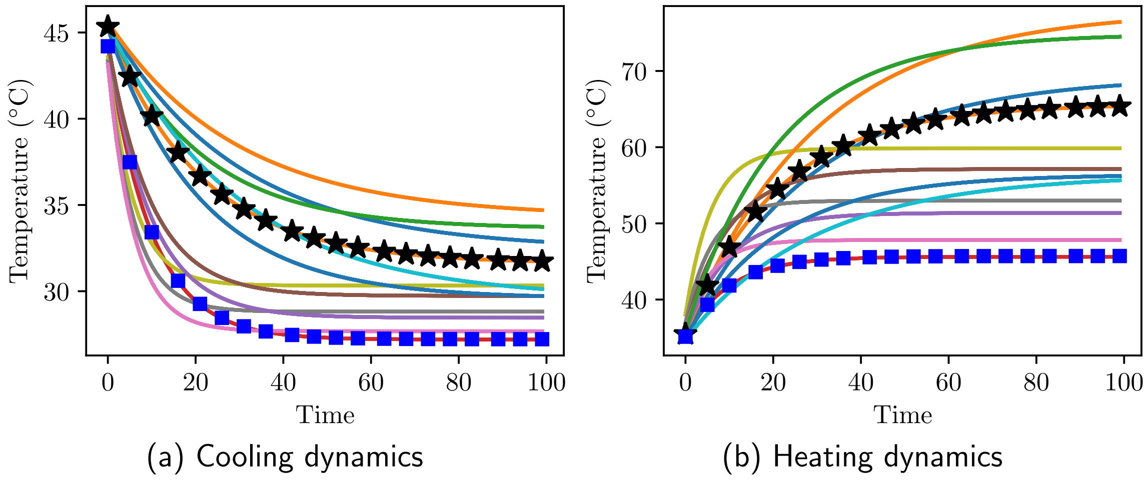

Figure 4: The predicted temperature response of the physics-based model to a step input power of (a) 1.1 W starting from an initial temperature of 46 °C and (b) 5 W starting from an initial temperature of 34 °C. The solid lines show the temperature response for different samples of the parameters (μ_1, μ_2) used to train the E-RLC, and the black stars and blue squares correspond to the fitted aluminum and glass thermal dynamics, respectively.

4.2. Improving controller performance

We use the setpoint profile in Figure 5 to test the three RLCs across a range of substrate parameters and ambient temperatures. Our goal is to quantify the *in silico* setpoint tracking performance under a range of conditions in order to test the extent to which the dynamics randomization in training can improve the control performance. Figures 6 and 7 show color surfaces of the MAE and CI for the three RLCs under different realizations for the parameters (μ_1, μ_2) as well as three different values for the ambient temperature T_∞. Each point in these plots is obtained by performing a closed-loop setpoint tracking simulation corrupted with measurement noise, where the substrate parameters μ_1 and μ_2 are shown on the axes. Each point represents the performance of the RLC agent for different substrate dynamics *in silico*. In each plot, the parameter combinations corresponding to experimentally obtained values for the glass, aluminum, and polyimide substrates are denoted by the light blue, black, and green stars, respectively (see Table 1). Note that, given the physical bounds on the input power P, it may not be practical to achieve adequate setpoint tracking for any substrate type (i.e., any realization of the substrate parameters). That is, at a given ambient temperature, some or all of the setpoints may be unreachable because of the actuation limitations of the input power. The unachievable temperature setpoints result in a high MAE and a low CI, indicating that the power input saturates at the bounds while the temperature setpoint is not achieved. This manifests itself as the triangle-shaped regions in each of the plots in Figures 6 and 7.

Figure 6 indicates that the G-RLC provides (perhaps surprisingly) high-performance *in silico* when tested against a wide range of parameters. However, as μ_1 increases while the ratio μ_2/μ_1 is kept constant (i.e., along the plot diagonal), we observe that the setpoint tracking performance deteriorates (i.e., MAE increases) using the G-RLC. By accounting for measurement noise and a small degree of parameter uncertainty while training the GU-RLC, the control performance can be somewhat

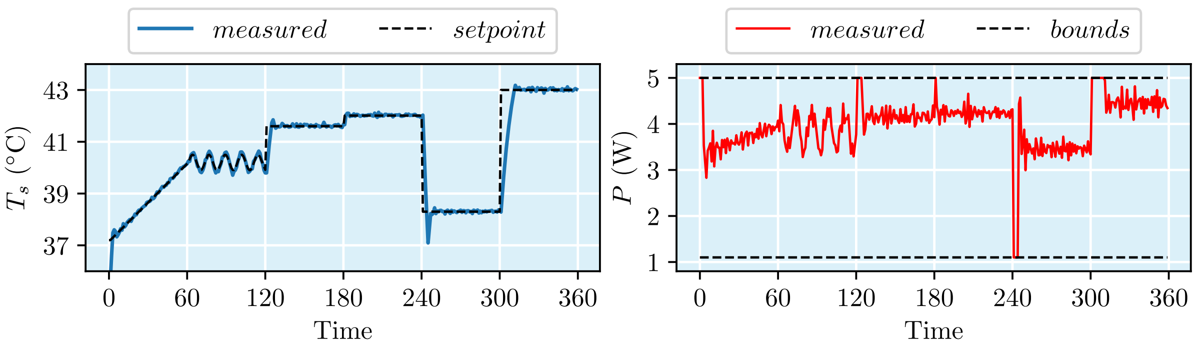

Figure 5: The closed-loop control performance of the G-RLC is tested *in silico* on the model of the borosilicate glass substrate in the absence of measurement noise. The setpoint tracking performance (left) achieves a MAE = 0.16 and the power input (right) achieves a CI = 69.2.

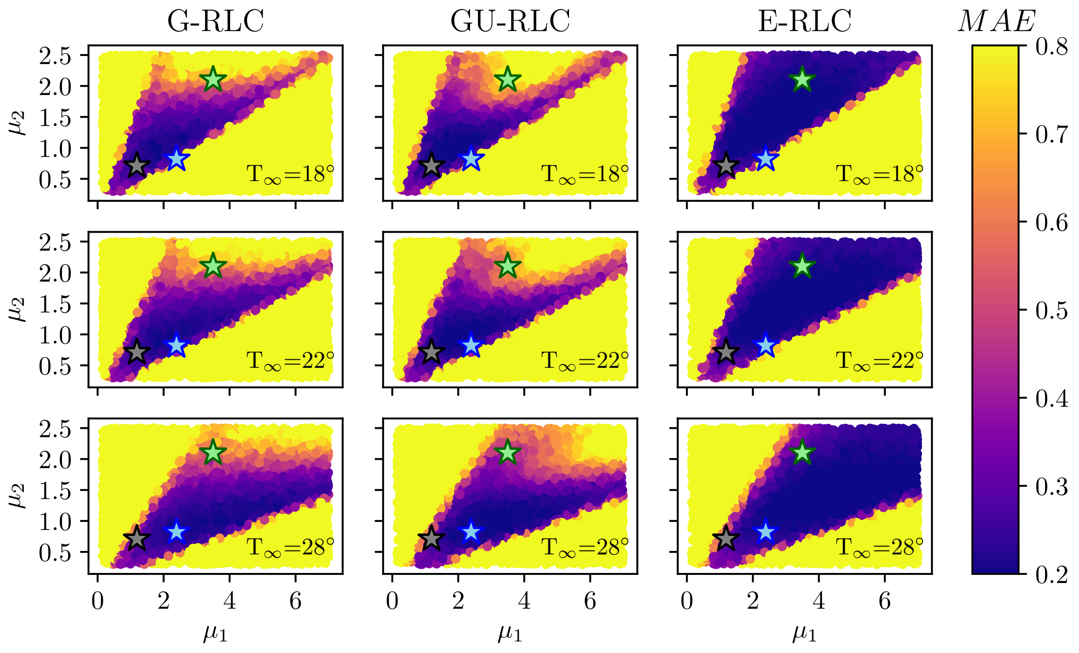

Figure 6: *In silico* evaluation of the control performance of the three RLCs. Each subplot shows the MAE of the closed-loop temperature setpoint tracking simulations for different realizations of the substrate parameters (μ_1, μ_2) at three different ambient temperatures T_∞. The model predictions are corrupted with measurement noise. Black, blue, and green stars represent, respectively, the model parameters fitted for the aluminum, borosilicate glass, and polyimide substrates.

improved. However, the most significant improvement is achieved with the E-RLC, where the control performance becomes significantly more robust to changes in the substrate properties that may lead to extremely fast temperature dynamics and/or high sensitivity to changes in process inputs (i.e., large process gains). Note that for $\mu_1/\mu_2 >> 1$ and $\mu_1/\mu_2 << 1$, poor control performance (very high MAE) is observed for all of the RLCs since these substrate parameters correspond to system dynamics in which the the power input bounds prevent effective tracking of the prescribed setpoints. Moreover, the reachable temperature setpoints are a function of the ambient temperature T_∞. For example, it may be difficult to track high temperature setpoints on colder days (i.e., lower T_∞).

To investigate the performance improvement of the E-RLC, Figure 7 shows the CI that corresponds to the same setpoint tracking simulations as in Figure 6. The regions in Figure 6 for which MAE is drastically reduced with the E-RLC (i.e., better control performance) also show a large reduction in CI in Figure 7. The results suggest that the E-RLC yields comparable or better performance (i.e., lower MAE) with less control effort (i.e., lower CI) than the G-RLC. This implies that the E-RLC has learned a control policy that improves the setpoint tracking performance while simultaneously decreasing

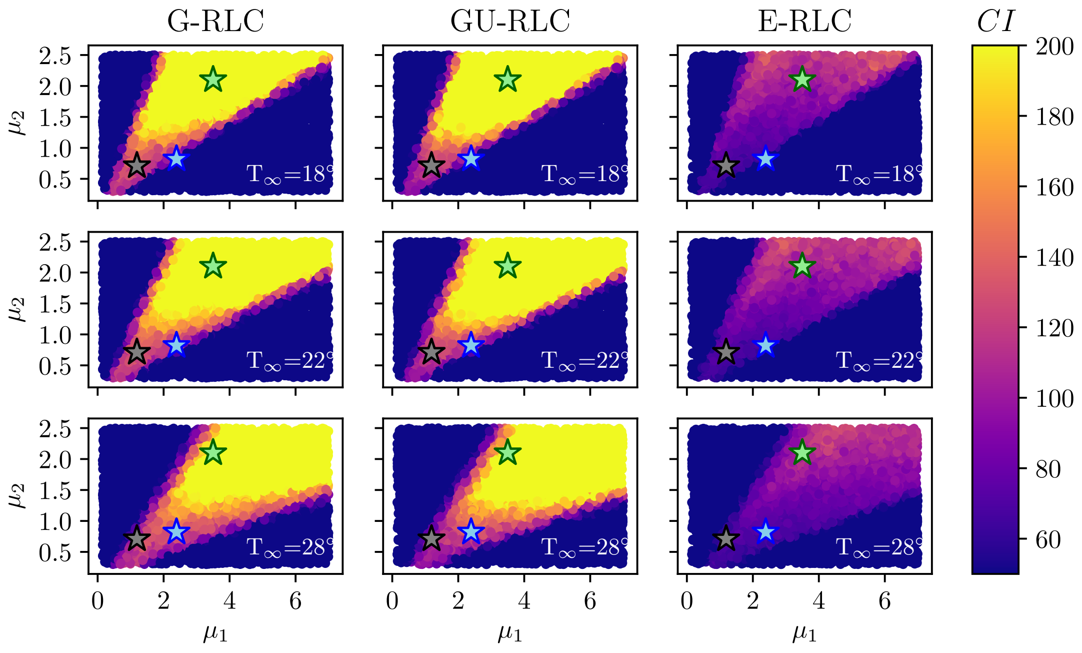

Figure 7: *In silico* evaluation of the cumulative input change of the three RLCs. Each subplot shows the CI of the closed-loop temperature setpoint tracking simulations for different realizations of the substrate parameters $(\mu_1,\ \mu_2)$ at three different ambient temperatures T_∞. The model predictions are corrupted with measurement noise. Black, blue, and green stars represent, respectively, the model parameters fitted for the aluminum, borosilicate glass, and polyimide substrates.

Table 3: Performance metrics for the G-RLC and E-RLC tested via *in silico* simulations on the borosilicate glass, aluminum, and polyimide substrates.

	MAE		CI	
Substrate	G-RLC	E-RLC	G-RLC	E-RLC
Glass	0.24	0.24	126.1	69.1
Aluminum	0.25	0.23	136.3	72.5
Polyimide	0.56	0.21	415.6	109.2

the control effort across a broad range of substrate dynamics. The performance metrics of the G-RLC and E-RLC for the closed-loop simulations performed using the experimentally determined parameters for glass, metal, and polyimide substrates are given in Table 3, and the time-course input power and substrate temperature profiles are shown in Appendix C. Table 3 indicates that the E-RLC can result in a particularly significant performance improvement on the polyimide substrate with a 2.5-fold decrease in MAE and a 4-fold decrease in CI.

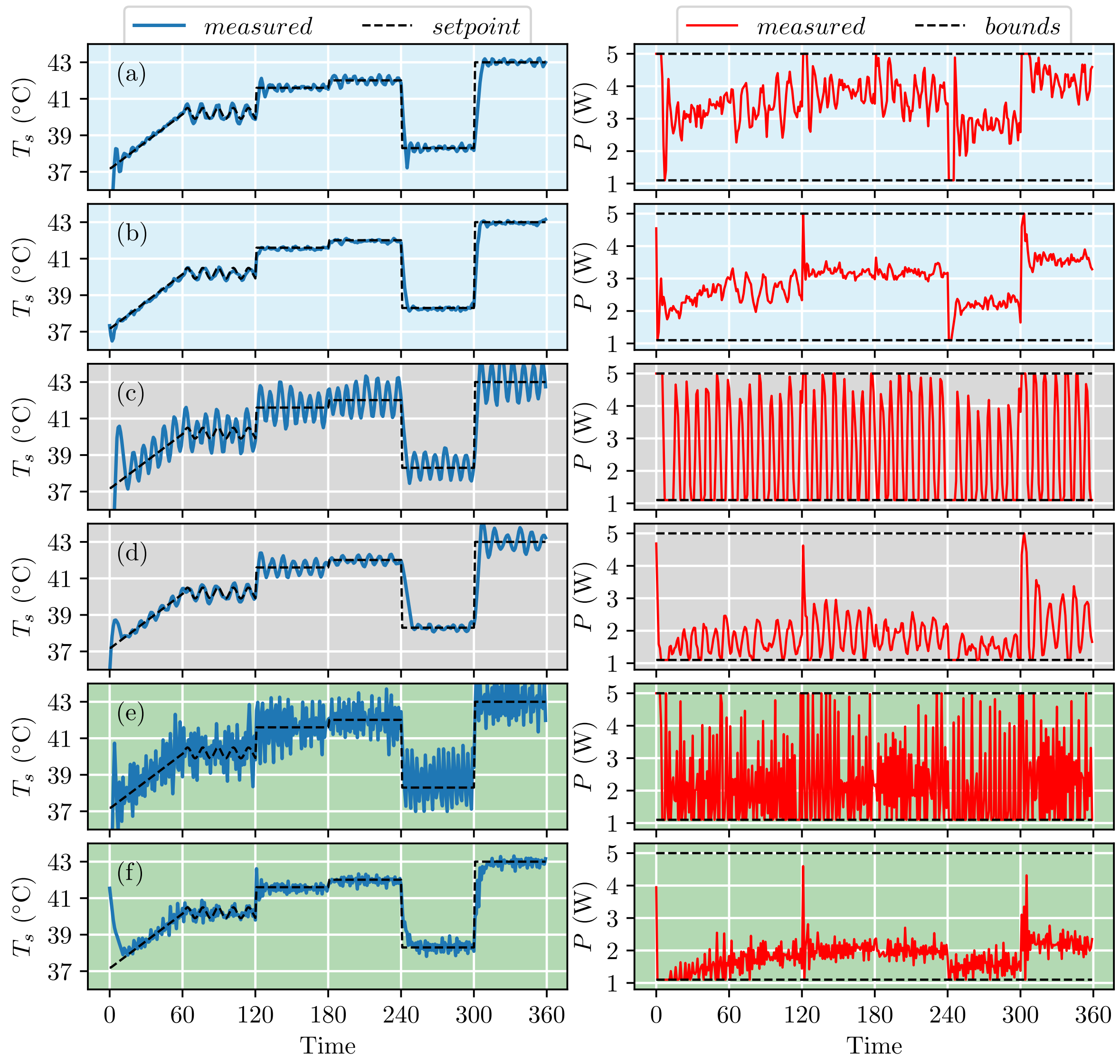

Figure 8: The temperature setpoint tracking performance of the G-RLC (a,c,e) and E-RLC (b,d,f) in real-time control experiments of the APPJ treatment of the borosilicate glass (a,b), aluminum (c,d), and polyimide (e,f) substrates. The E-RLC significantly outperforms the G-RLC in every case in terms of the setpoint tracking error (left column) and control effort (right column).

5. Results: Real-time experiments

5.1. Temperature control on different substrates

We perform real-time control experiments using both the G-RLC and E-RLC on the APPJ setup described in section 2.1. We test the RLCs on three different substrates: borosilicate glass, aluminum, and polyimide. Results of the closed-loop control experiments are shown in Figure 8. As can be seen, the control performance of the E-RLC is notably better than the control performance of the G-RLC over all

Table 4: Performance metrics for the G-RLC and E-RLC tested via real-time control experiments of the APPJ for treatment of the glass, aluminum, and polyimide substrates.

	MAE		CI	
Substrate	G-RLC	E-RLC	G-RLC	E-RLC
Glass	0.19	0.12	98.3	48.6
Aluminum	0.64	0.30	240.1	73.0
Polyimide	0.81	0.24	499.6	129.3

substrates. While the G-RLC performs well on the borosilicate glass substrate (Figure 8a), the E-RLC performs better with a 33% reduction in MAE and a 50% reduction in CI (see Figure 8b and Table 4). The presence of unmodeled dynamics in the experimental setup as well as the uncertainty associated with the estimated parameters μ_1 and μ_2 result in performance deterioration of the G-RLC compared to simulation studies. Small oscillations in the input and small setpoint overshoots are observed when the G-RLC encounters dynamics that are not *exactly* what the agent was trained on. In contrast, the E-RLC provides excellent setpoint tracking performance with low control effort, comparable to the base case shown in Figure 5. Achieving acceptable control performance with G-RLC over aluminum and polyimide substrates is even more challenging since the temperature dynamics of both substrates are considerably different than that of glass. Both substrates exhibit thermal dynamics that are highly sensitive to changes in the input power (i.e., large process gain), with relatively slow settling times for aluminum and relatively fast settling times for polyimide. For both substrates, the G-RLC struggles to maintain the setpoint (Figure 8c and 8e). The input oscillations and setpoint overshoots are more pronounced over these substrates as the mismatch between the training data of the G-RLC and the actual system behavior is greater. Large oscillations in the power input indicate the increased control effort of the G-RLC. In contrast, the E-RLC can follow the temperature setpoint more closely with considerably less control effort (Figure 8d and 8f). The performance of the G-RLC and E-RLC on the three substrates is summarized in Table 4. The E-RLC demonstrates significant improvement in setpoint tracking on all three substrates with much less control effort.

5.2. Disturbance rejection

Lastly, to test the capability of the E-RLC for effective substrate temperature control beyond the setting on which it was trained (i.e., responding to setpoint changes), we introduce a disturbance to the APPJ operation. Changes in the jet tip-to-substrate separation distance present a relevant disturbance to APPJ applications. Such disturbances can arise due to the substrate topology or variations in the separation distance during hand-held APPJ treatments, which in turn can have a significant effect

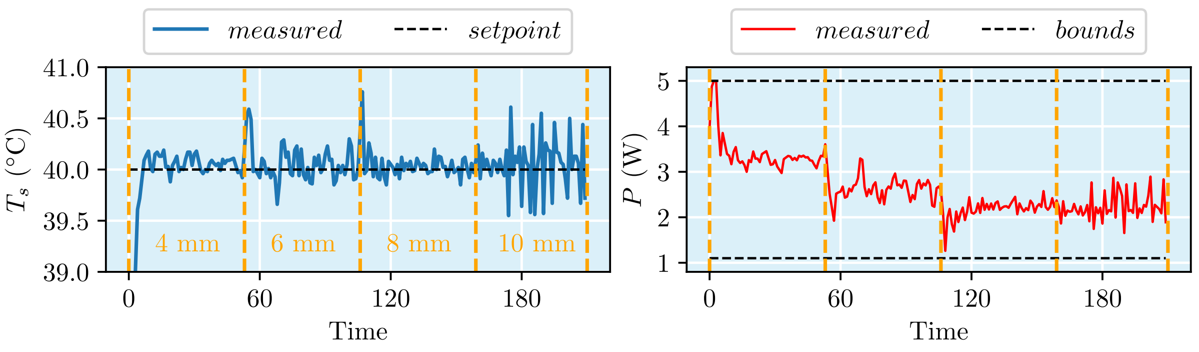

Figure 9: The temperature setpoint tracking performance of the E-RLC in real-time control experiments of the APPJ for treatment of the glass substrate when the jet tip-to-substrate separation distance undergoes step changes from 4 mm to 6 mm, 6 mm to 8 mm, and 8 mm to 10 mm.

on thermal dynamics of the substrate [12, 13, 15]. Here, we investigate the performance of the E-RLC in effectively maintaining a constant substrate temperature when the separation distance is abruptly changed. Figure 9 shows the performance of the E-RLC in controlling the substrate temperature as the separation distance is increased by 2 mm in a step-wise manner. As can be seen, the E-RLC can quickly recover and maintain effective setpoint tracking after such a disturbance. Note that the scale of the y-axis in Figure 9 is $\pm$ 1 ^{o}C, suggesting that the excursions of the substrate temperature are on the order of only 0.5 ^{o}C. At high separation distances (>10 mm), the substrate temperature and the power input start to fluctuate. This can be attributed to the fact that at this separation distance the plasma is on the verge of decoupling from the substrate and thus the plasma starts exhibiting unstable behavior. Within this operating region, the available actuation of the APPJ is not adequate for effective regulation of the thermal behavior of the substrate.

6. Discussion

This paper aimed at addressing a major challenge in feedback control of APPJs for treatment of complex substrates: how to account for variations in the plasma dynamics and/or plasma-substrate interactions using learning-based control. We demonstrated how simulated training data can be used to obtain a high-performing reinforcement learning controller for controlling the thermal effects of a kHz-excited APPJ in He on a variety of substrates with greatly different thermal and electrical properties. It was shown that the broad range of dynamics of the plasma-substrate interactions can be effectively accommodated by using random realizations (i.e., dynamics randomization) for parameters of a physics-based model used to generate the training data. Without using dynamics randomization, *in silico* and real-time control experiments showed that the RLC can be ineffective when it encounters a substrate that has significantly different dynamics than that the agent was trained on. In contrast, by using dynamics randomization, we achieved a significantly improved control performance over the parameter range tested *in silico* and over three substrates (i.e., glass, aluminum, and polyimide) in real-time experiments. This demonstrates the importance of dynamics randomization for successful sim-to-real transfer of an RLC.

Our *in silico* training procedure and subsequent successful transfer to the APPJ is important for several reasons. Firstly, the *in silico* training is orders of magnitude faster compared to using data collected from experiments. A maximum return for the RLC policy is achieved after approximately 500 training epochs with 10,000 time steps per epoch. Since our APPJ sampling frequency is fixed at 1.3 s^{-1}, a training procedure based on experimental data would take $\sim$75 days of non-stop APPJ operation. In contrast, *in silico* training was completed in approximately 1 hour and 50 minutes (0.075 days). Future research will investigate if more sample efficient RL algorithms would enable training of RL agents with limited experimental data. An additional challenge in using experimental data is that the facile selection of an ensemble of model parameters *in silico* cannot be easily replicated in an experimental setting. To perform training with dynamics randomization of real experiments (i.e., *in vivo* training), one would need extensive experiments to find a representative set of substrates that sufficiently spans the space of model dynamics one wishes to train over. It can be inconvenient, if not impractical, to change the experimental conditions to cover the entire model parameter space that the RLC may need to operate within during the APPJ operation. Finally, a key requirement of APPJ treatments is safety, particularly in a medical context where the substrate is a patient receiving treatment. Training RL agents *in vivo* can inflict damage on the target substrate, especially in early iterations, making this approach prohibitive. By avoiding any *in vivo* training, we can ensure high-performing RL agents can be designed and tested *in silico* before applying the controller in the real world.

An attractive feature of the proposed sim-to-real transfer approach is the use of a simple model of the substrate temperature dynamics in conjunction with dynamics randomization. The speed-up in training allowed by using a computationally inexpensive model was appreciable and dynamics randomization allowed accommodating a range of experimentally observed dynamics. However, there may be limitations to using such simple models. Particularly, we restricted our model structure to be linear and focused on a single output variable, the substrate temperature. On the other hand, APPJs can exhibit a range of nonlinear behaviors and may have multiple (e.g., chemical, electrical, thermal) effects on target substrates. In this work, mildly nonlinear temperature dynamics were observed on the aluminum substrate, which resulted in a somewhat oscillatory temperature response even with E-RLC strategy (see Figure 8d). Thus, future work will involve using more complex, nonlinear and multivariable models of the APPJ to generate simulation training data. The proposed sim-to-real transfer approach can be readily applied using models of arbitrary complexity. However, the trade-off between model complexity and training time must be taken into account. It may be challenging to train RL agents using data generated with models with a large number of uncertain parameters, as the required training time grows with parameter space that is sampled during training.

There remain many interesting extensions and improvements of the proposed sim-to-real transfer approach in order to explore the potential role of RL in the regulation of cumulative and spatially varying plasma effects, i.e., dose delivery. In most APPJ applications, one wishes to administer a predetermined dose as accurately as possible over a two-dimensional surface in a fixed period of time. The E-RLC agent developed in this work can be useful as a lower level controller for fast disturbance rejection, thereby allowing effective temperature regulation across substrates with different electrical and thermal dynamics [44]. Moreover, RL can be useful in analyzing and integrating more complex sensory information such as optical emission spectra and current waveforms for diagnostics applications [45], and incorporate further manipulated inputs for control problems. An interesting future research will be to see if more recent techniques such as Generalized Advantage Estimate [46], Proximal Policy Optimization [47], or Soft Actor-Critic algorithms [48] are necessary to succeed in these more complex APPJ control problems. The proof-of-concept demonstration in this work indicates the vast potential of reinforcement learning in complementing and, in some instances, replacing classical and advanced control strategies for effective treatment delivery with APPJs.

Acknowledgments

This work is partially supported by NSF grant 1839527. MW and BS were partly supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 666983, MaGic).

Appendix A. Physics-based model predictions versus experiment

The parameters of the physics-based model are summarized in Table A1. The model is validated against experimental measurements; Figure A1 shows the measured and predicted temperatures of the glass substrate as the applied power is increased in a step-wise manner. The model appears to provide a reasonable approximation of the observed substrate temperature dynamics.

Table A1: Parameters of the physics-based model of the thermal dynamic of the substrate

Parameter	Value
ρ	2.8×10^3 kgm^{-3}
c_p	795 Jkg^{-1}K^{-1}
d	0.2 mm
r	1.5 mm
η	0.4
k	1.43 Wm^{-2}K^{-1}
$\bar{\mu}_1$	$38\mu_1$
$\bar{\mu}_2$	$0.003\mu_2$

Appendix B. Details of the actor-critic algorithm

Some additional details of the actor-critic algorithm not presented in the main text are included in this section. Several actor-critic variants of policy gradient exist [38, 25],

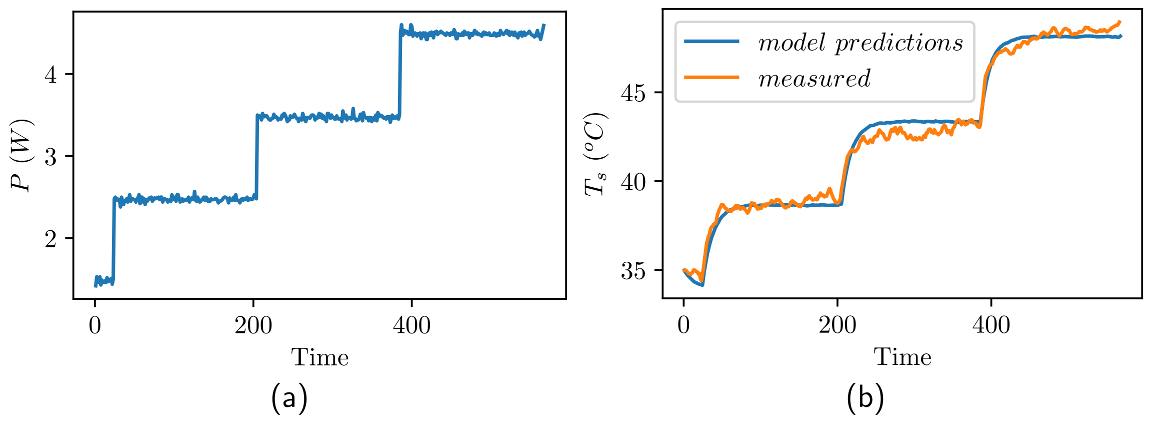

Figure A1: Experimental validation of the physics-based model on a glass substrate. (a) Applied APPJ power and (b) predicted and measured values of the substrate temperature.

which commonly seek to estimate the gradient of the expected rewards with respect to the policy. The form of the gradient common to these methods is as follows [46],

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} R_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{B.1}$$

where the form of R_t varies between the different methods. For example, for the most basic version of policy gradient, $R_t = \sum_{t=0}^{T-1} r_t$. The problem is that sampling this quantity produces a very high variance rewards signal. Many policy gradient methods seek to reduce this variance, and Ref. [46] provides an excellent summary. The aforementioned references provide relevant derivations; in this work, we use the gamma-discounted TD-residual, $R_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$, which reduces the variance in the rewards signal (the discount factor, $\gamma \in [0, 1.0]$, is introduced to discount future rewards). Specifically, it uses the value function, defined as the expected future rewards under the current policy from being in s_t.

$$V^\pi(s_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t:T-1}}\left[\sum_{\Delta=0}^{T-1} r_{t+\Delta}\right] \tag{B.2}$$

Now, combining (B.1) with the TD-residual and estimating the expectation value over N different MC roll-outs, we can rewrite the gradient as follows:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T-1} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left(r_{i,t} + \gamma V^\pi(s_{i,t+1}) - V^\pi(s_{i,t})\right) \tag{B.3}$$

Once (B.3) has been computed, we can maximize the expected rewards by taking a gradient step in θ. This equation also provides the link between computing the policy gradient and the two network actor-critic structure summarized in the main text. In the two ANN design, the policy is specified by the outputs of the actor network, $\pi_\theta(a|s)$, and the value function is approximated by the critic network, $\hat{V}^\pi_\phi(s_t)$. The actor and critic can now be trained in tandem via Algorithm 1 where the corresponding block diagram is shown in Figure B1. Note that Figure B1 is the actor-critic specific realization of the more general reinforcement block flow diagram shown in Figure 2.

1. Monte Carlo roll-outs:

The details of roll-out collection using domain randomization were discussed in the main text. Briefly, we collect $N = 100$ simulated roll-outs of $T = 100$ time steps. At the beginning of each roll-out a new setpoint is randomly chosen from the uniform distribution, $T_{sp} \sim \mathcal{U}[34, 46]$ and trajectories are propagated forward in time based on the transition model and control policy. This can be expressed succinctly as collecting $N \times T$ tuples of (s_t, a_t, s_{t+1}, r_t) where the model transitions $p(s_{t+1}|s_t, a_t, \mu)$ depend on the model parameters chosen to be sampled, $\mu \sim \rho_\mu$, in a given roll-out.

Algorithm 1 Online, two network actor-critic

```
 1: for each actor training iteration do
 2:     policy roll-outs: collect $(s_t, a_t, s_{t+1}, r_t)$ for all $N \times T$ time steps
 3:     for number of critic target updates do
 4:         compute critic targets: $z_t = r_t + \gamma \hat{V}^{\pi}_{\phi}(s_{t+1})$
 5:         for gradient steps per target updates do
 6:             compute critic loss: $\mathcal{L}(\phi)$
 7:             update critic: $\phi \leftarrow \phi + \alpha \nabla_{\phi} \mathcal{L}(\phi)$
 8:     estimate TD residual: $\hat{R}_t = r_t + \gamma \hat{V}^{\pi}_{\phi}(s_{t+1}) - \hat{V}^{\pi}_{\phi}(s_t)$
 9:     compute cost gradient: $\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \hat{R}_t$
10:     update actor: $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$
```

2. Critic training:

The goal of the critic network is to approximate the value function with weights ϕ, $\hat{V}^{\pi}_{\phi}(s_t)$. Once a batch of simulated data has been collected and the rewards computed under the current iteration of the policy, the value function must be approximated, i.e. fit to the data that has been collected. When using the TD-residual, this must be done in an iterative manner. The difference between the critic network's predictions for each state, $\hat{V}^{\pi}_{\phi}(s_t)$, and its target values, z_t, as shown in Algorithm 1, are formulated as the

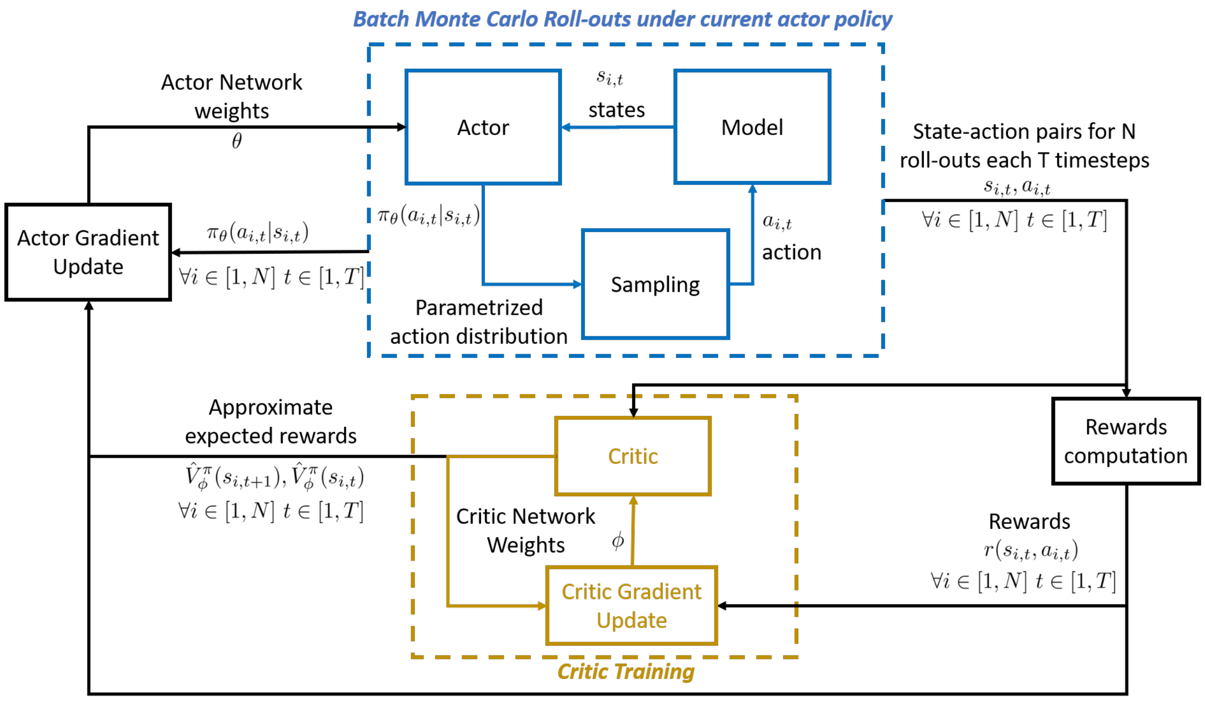
Batch Monte Carlo Roll-outs under current actor policy
Actor Network weights θ
Actor
$s_{i,t}$ states
Model
State-action pairs for N roll-outs each T timesteps
$s_{i,t}, a_{i,t}$
$\forall i \in [1, N]$ $t \in [1, T]$
Actor Gradient Update
$\pi_\theta(a_{i,t}|s_{i,t})$
$\forall i \in [1, N]$ $t \in [1, T]$
$\pi_\theta(a_{i,t}|s_{i,t})$
$a_{i,t}$ action
Sampling
Parametrized action distribution
Approximate expected rewards
$\hat{V}^\pi_\phi(s_{i,t+1}), \hat{V}^\pi_\phi(s_{i,t})$
$\forall i \in [1, N]$ $t \in [1, T]$
Critic
Critic Network Weights
ϕ
Critic Gradient Update
Critic Training
Rewards computation
Rewards
$r(s_{i,t}, a_{i,t})$
$\forall i \in [1, N]$ $t \in [1, T]$

Figure B1: A block diagram representation of the actor-critic algorithm for training the RLC

critic network's loss and minimized by optimizing the weights of the critic network:

$$\mathcal{L}(\phi) = \frac{1}{N}\sum_{t} ||z_t - \hat{V}^{\pi}_{\phi}(s_t)||^2 \tag{B.4}$$

However, once this loss has been minimized, the critic's predictions are no longer self-consistent with the new values of ϕ. Hence the target values must be recomputed and the minimization process repeated. Within each actor training step, we update the critic target values ten times, and we update the critic network with ten gradient steps per target update.

3. Actor update

Once the critic network has been optimized for the current batch of simulated data, the policy gradient can be computed via (B.3) and a gradient step computed in θ (with learning rate α) to improve the control policy for the next iteration of data collection.

4 Learning curves

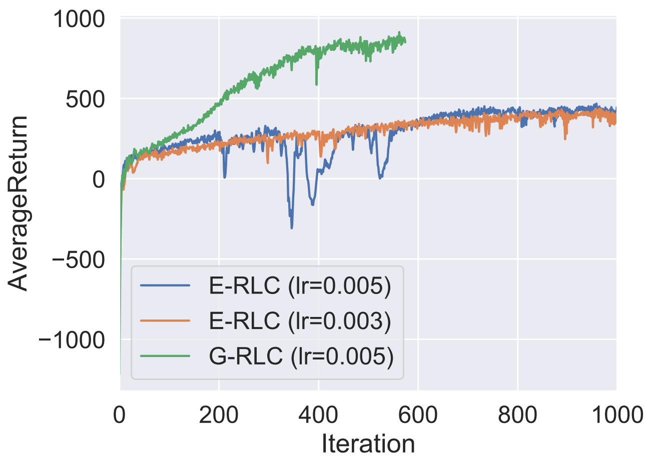

Figure B2: Training curves for the G-RLC with a learning rate of 0.005 and the E-RLC with both a learning rate 0.005 and 0.003.

The learning curves for the G-RLC with a learning rate of 0.005 and the E-RLC with both a learning rate of 0.005 and 0.003 are shown in Figure B2. The average return per MC roll-out is plotted versus actor training iteration. The G-RLC quickly learns a control policy that practically achieves the maximum possible return based on the rewards structure described in the manuscript ($r_{max}(s_t, a_t) = 10$ for $T = 100$ time steps per MC roll-out). The G-RLC training takes longer to plateau and achieves a lower average return than the E-RLC, but this should not be interpreted as a shortcoming. The E-RLC training incorporates measurement noise, which inherently makes the setpoint tracking more difficult. The E-RLC must also operate on a wide range of different model dynamics. In each training iteration, some fraction of the E-RLC roll-outs are performed with very slow substrate dynamics (see Figure 4), and the limitations of these slow dynamics combined with a bounded action space reduce the minimum setpoint tracking error that can theoretically be achieved. We also highlight the sensitivity of the training to the learning rate, as demonstrated by the more stable E-RLC training when a learning rate of 0.003 is used.

Appendix C. Visualization of *in silico* set point tracking experiments

In silico setpoint tracking simulations of the 3 RLC controllers are shown in Figure C1, whose quantitative performance metrics were presented in Table 3. The largest performance gain with the E-RLC occurs on the polyimide substrate model (Figure C1f). Notably from Figure 7, the CI is also significantly reduced with the E-RLC in the vicinity of borosilicate glass and aluminum parameters. The setpoint tracking performance on an aluminum substrate is plotted in Figure C1c-d, which show how the E-RLC demonstrates better control performance than the G-RLC. Not only does the E-RLC achieve slightly lower MAE, the CI is also significantly reduced as the E-RLC becomes better at controlling the temperature without exhibiting the more oscillatory input profiles obtained under the G-RLC. Regardless, it is important to note that the G-RLC performs surprisingly well across a range of model parameters, given the large difference in temperature dynamics between borosilicate glass and aluminum substrates. However, this moderate transferability of the G-RLC only applies to *in silico* results as is shown in the feedback control experiments.

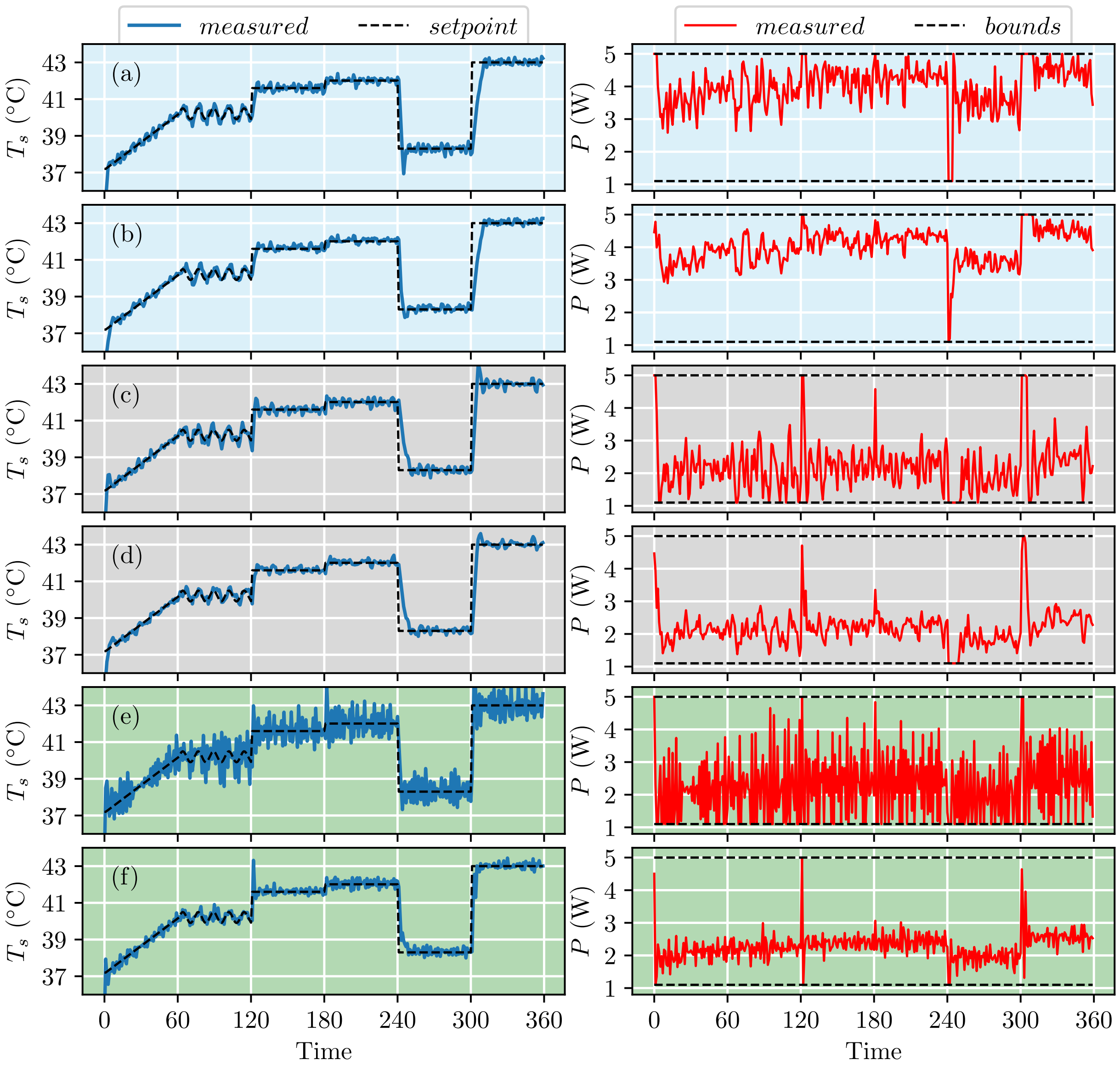

Figure C1: The temperature setpoint tracking performance of the G-RLC (a,c,e) and E-RLC (b,d,f) in closed-loop simulations of the APPJ treating a borosilicate glass (a,b), aluminum (c,d), and polyimide (d,f) substrate. The performance of the E-RLC is significantly better for each substrate compared to that of the G-RLC in terms of setpoint tracking error (left column) and control effort (right column).

References

[1] J. Winter, R. Brandenburg, and K.-D. Weltmann, "Atmospheric pressure plasma jets: an overview of devices and new directions," *Plasma Sources Science and Technology*, vol. 24, no. 6, p. 064001, 2015.

[2] M. Laroussi and F. Leipold, "Evaluation of the roles of reactive species, heat, and UV radiation in the inactivation of bacterial cells by air plasmas at atmospheric pressure," *International Journal of Mass Spectrometry*, vol. 233, no. 1-3, pp. 81–86, 2004.

[3] R. Morent, "Editorial: Atmospheric Pressure Plasma Polymerization," *The Open Plasma Physics Journal*, vol. 7, no. 1, p. 6, 2013.

[4] K. S. Siow, L. Britcher, S. Kumar, and H. J. Griesser, "Plasma Methods for the Generation of Chemically Reactive Surfaces for Biomolecule Immobilization and Cell Colonization - A Review," *Plasma Processes and Polymers*, vol. 3, no. 6-7, pp. 392–418, 2006.

[5] T. von Woedtke, H.-R. Metelmann, and K.-D. Weltmann, "Clinical plasma medicine: State and perspectives of in vivo application of cold atmospheric plasma," *Contributions to Plasma Physics*, vol. 54, no. 2, pp. 104–117, 2014.

[6] H. R. Metelmann, C. Seebauer, R. Rutkowski, M. Schuster, S. Bekeschus, and P. Metelmann, "Treating cancer with cold physical plasma: On the way to evidence-based medicine," *Contributions to Plasma Physics*, vol. 58, no. 5, pp. 1–5, 2018.

[7] M. Laroussi, M. Kong, G. Morfill, and W. Stolz, *Plasma Medicine*. Cambridge University Press, 2012.

[8] K. D. Weltmann and T. von Woedtke, "Plasma medicine - current state of research and medical application," *Plasma Physics and Controlled Fusion*, vol. 59, no. 1, p. 014031, 2016.

[9] S. A. Norberg, E. Johnsen, and M. J. Kushner, "Helium atmospheric pressure plasma jets touching dielectric and metal surfaces," *Journal of Applied Physics*, vol. 118, no. 1, pp. 1–13, 2015.

[10] B. L. Klarenaar, O. Guaitella, R. Engeln, and A. Sobota, "How dielectric, metallic and liquid targets influence the evolution of electron properties in a pulsed He jet measured by Thomson and Raman scattering," *Plasma Sources Science and Technology*, vol. 27, no. 8, 2018.

[11] L. Ji, W. Yan, Y. Xia, and D. Liu, "The effect of target materials on the propagation of atmospheric-pressure plasma jets," *Journal of Applied Physics*, vol. 123, no. 18, 2018.

[12] D. Breden and L. L. Raja, "Computational study of the interaction of cold atmospheric helium plasma jets with surfaces," *Plasma Sources Science and Technology*, vol. 23, no. 6, p. 065020, 2014.

[13] S. Wu, Z. Wand, Q. Huang, X. Lu, and Y. Pan, "Study on a room-temperature air plasma for biomedical application," *IEEE Transactions on Plasma Science*, vol. 39, no. 6, pp. 1489–1495, 2011.

[14] J. Shin and L. L. Raja, "Run-to-run variations, asymmetric pulses, and long time-scale transient phenomena in dielectric-barrier atmospheric pressure glow discharges," *J. Phys. D. Appl. Phys.*, vol. 40, no. 10, pp. 3145–3154, 2007.

[15] D. Gidon, D. B. Graves, and A. Mesbah, "Effective dose delivery in atmospheric pressure plasma jets for plasma medicine: A model predictive control approach," *Plasma Sources Sci. Technol.*, vol. 26, no. 8, p. 085005, 2017.

[16] D. Gidon, B. Curtis, J. A. Paulson, D. B. Graves, and A. Mesbah, "Model-Based Feedback Control of a kHz-Excited Atmospheric Pressure Plasma Jet," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 2, no. 2, pp. 129–137, 2018.

[17] D. Gidon, D. B. Graves, and A. Mesbah, "Spatial thermal dose delivery in atmospheric pressure plasma jets," *Plasma Sources Science and Technology*, vol. 28, p. 025006, 2019.

[18] Y. Lyu, L. Lin, E. Gjika, T. Lee, and M. Keidar, "Mathematical modeling and control for cancer treatment with cold atmospheric plasma jet," *Journal of Physics D: Applied Physics*, vol. 52, no. 18, p. 185202, 2019.

[19] M. Keidar, D. Yan, and J. H. Sherman, "Adaptive plasmas and recent progress in plasma

application in cancer therapy," in *Cold Plasma Cancer Therapy*, 2053-2571, pp. 7–1 to 7–9, Morgan & Claypool Publishers, 2019.

[20] N. Balcon, G. J. M. Hagelaar, and J. P. Boeuf, "Numerical model of an argon atmospheric pressure RF discharge," *IEEE Transactions on Plasma Science*, vol. 36, no. 5, pp. 2782–2787, 2008.

[21] D. Spether, M. Scharpf, J. Hennenlotter, C. Schwentner, A. Neugebauer, D. Nüßle, K. Fischer, H. Zappe, A. Stenzl, F. Fend, A. Seifert, and M. Enderle, "Real-time tissue differentiation based on optical emission spectroscopy for guided electrosurgical tumor resection," *Biomedical Optics Express*, vol. 6, no. 4, p. 1419, 2015.

[22] H.-R. Metelmann, T. von Woedtke, and K.-D. Weltmann, eds., *Comprehensive Clinical Plasma Medicine: Cold Physical Plasma for Medical Application.* Springer, 2018.

[23] S. U. Kalghatgi, G. Fridman, M. Cooper, G. Nagaraj, M. Peddinghaus, M. Balasubramanian, V. N. Vasilets, A. F. Gutsol, A. Fridman, and G. Friedman, "Mechanism of blood coagulation by nonthermal atmospheric pressure dielectric barrier discharge plasma," *IEEE Transactions on plasma science*, vol. 35, no. 5, pp. 1559–1566, 2007.

[24] A. Mesbah and D. B. Graves, "Machine learning for modeling, diagnostics, and control of non-equilibrium plasmas," *Journal of Applied Physics*, vol. 52, p. 30LT02, 2019.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts: The MIT Press, 2015.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," vol. 2, no. 1, pp. 1–127, 2015.

[28] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," *Openai baselines, GitHub repository*, 2017.

[29] J. Hoskins and D. Himmelblau, "Process control via artificial neural networks and reinforcement learning," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 241–251, 1992.

[30] C. W. Anderson, D. C. Hittle, A. D. Katz, and R. Kretchmar, "Synthesis of reinforcement learning, neural networks and PI control applied to a simulated heating coil," *Artif. Intell. Eng.*, vol. 11, no. 4, pp. 421–429, 1997.

[31] J. Govindhasamy, S. McLoone, and G. Irwin, "Reinforcement learning for process identification, control and optimisation," in *2004 2nd Int. IEEE Conf. 'Intelligent Syst. Proc. (IEEE Cat. No.04EX791)*, vol. 1, pp. 316–321, IEEE, 2004.

[32] S. Syafiie, F. Tadeo, and E. Martinez, "Model-Free Learning Control of Chemical Processes," in *Reinf. Learn.*, no. January, I-Tech Education and Publishing, 2008.

[33] S. Spielberg, R. Gopaluni, and P. Loewen, "Deep Reinforcement Learning Approaches for Process Control," in *6th Int. Symp. Adv. Control Ind. Process.*, 2017.

[34] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 23–30, IEEE, 2017.

[35] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran, "Epopt: Learning robust neural network policies using model ensembles," *CoRR*, vol. abs/1610.01283, 2016.

[36] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," *CoRR*, vol. abs/1611.04201, 2016.

[37] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *CoRR*, vol. abs/1710.06537, 2017.

[38] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *Proc. 12th Int. Conf. Neural Inf. Process. Syst.*, pp. 1057–1063, 1999.

[39] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network

vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, 2019.

[40] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," in *Adv. Neural Inf. Process. Syst.*, 2000.

[41] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[42] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

[43] M. Witman, "PlasmaRL." `https://github.com/mwitman1/PlasmaRL`, 2019.

[44] D. Gidon, D. B. Graves, and A. Mesbah, "Predictive control of 2D spatial thermal dose delivery in atmospheric pressure plasma jets," *Plasma Sources Science and Technology*, vol. In Press, 2019.

[45] D. Gidon, X. Pei, A. D. Bonzanini, D. Graves, and A. Mesbah, "Machine Learning for Real-time Diagnostics of Cold Atmospheric Plasma Sources," *IEEE Transactions in Radition Science and Plasma Medicine*, vol. In Press, 2019.

[46] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *ICLR*, 2016.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[48] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018.