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

Matthew Witman*, Dogan Gidon*, David Graves, Berend Smit, Ali Mesbah

Abstract. Applications of atmospheric pressure plasma jets (APPJs) present challenging controls problems due to the complexity of the plasma/substrate interactions. APPJs are particularly sensitive to changes in the electrical and thermal properties of treated substrates exhibiting large variations in thermal dynamics. The increasingly popular reinforcement learning (RL) based methods have great potential to aid regulation of thermal properties of APPJs across different substrates. Using only simulated data generated by a simple lumped-parameter energy balance, we train a robust reinforcement controller to perform temperature setpoint tracking by exposing it to randomized dynamics that capture the diverse temperature responses of different substrates. This approach is an efficient, safe, and flexible means for training an effective RL controller which, when transferred to the live environment, is able to perform effective temperature control over a wide variety of substrates without further fine-tuning.

1. Introduction

Atmospheric pressure plasma jets (APPJs) are unique tools which permit the treatment of heat and pressure sensitive substrates [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, among others [5–8]. However, APPJs suffer from considerable operational variability. APPJ dynamics are sensitive to disturbances such as changes in jet-tip-to-substrate separation distance [9, 10] and changes in substrate characteristics [11–13]. Moreover, jet dynamics can change drastically from run to run, even when the APPJ is operated at practically identical operating conditions [14]. These variations can result in thermal damage to sensitive substrates [15] and, in general, complicates the delivery of safe and therapeutically effective treatment. All these complexities necessitate model-based feedback control methods for reliable APPJ operation [16–19].

A significant hindrance to the flexible operation of APPJs, even in the presence of feedback control, is the wide variations in the coupled APPJ-substrate dynamics. For

example, APPJs tend to spread over dielectric substrates due to charge accumulation on the surface. On the other hand, so-called "discharge re-strike" phenomena observed on conductive substrates result in a significantly higher power deposition in the discharge as well as considerable changes in electric fields and species densities delivered to the targets [11, 12]. Similarly, the electron emissivity of the treated substrates can significantly influence the discharge-substrate interactions giving rise to, for example, different mode behaviors (e.g., α-γ transition in RF APPJs [20]). Yet, substrates with heterogeneous characteristics often arise in materials processing and medical applications. For example, healthy tissue and wounds generally have different thermal and electrical properties. Furthermore, skin electrical conductivity can change from patient to patient or even from point to point on the same patient. It can be challenging to develop model-based control strategies to accommodate such a broad range of dynamics. One potential solution to address this challenge is the use of data-driven machine learning (ML) methods that rely on universal function approximators called artificial neural networks (ANNs).

Deep Reinforcement learning (RL) is emerging area of research in ML whereby an ANN is trained to take optimal actions to maximize a reward (or minimize a penalty) through continuous feedback during training [21]. RL has found remarkable success in a range of applications including complex gameplay, robotic control, autonomous navigation, and chemical process control, among others [22–29]. 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 in a simulated domain has multiple advantages. Firstly, the RL agent's exploration during the learning process can raise safety concerns if deployed in a live environment (i.e. thermal damage to the substrate). Secondly, many deep RL algorithms 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 than generating simulated data. Finally, simulations permit the generation of data under conditions which may be difficult and/or expensive to probe experimentally. These concerns 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.

However, transferring the simulated performance of RL agents to a real-world environment can be challenging given the "reality gap", or the mismatch between system dynamics in the simulated and real-world environments. Building a high fidelity model or more complex simulators which can capture every aspect of a real environment is often prohibitively expensive or inefficient. Therefore methods that enrich the data sets on which RL agents are trained, have been proposed to address the reality gap. These methods can account for uncertain and difficult-to-model aspects of the real environments and enable successful sim-to-real transfer of RL agents without the need for any live data [30–32]. The method we focus on in this work is "dynamics randomization" which involves training an RL agent using a relatively simple model to describe the system dynamics [33]. The parameters of the model are randomized during

the training process in order to account for the mismatch between the simulated and real environments.

In the present work, we demonstrate the control of thermal effects of a kHz-excited APPJ in helium across multiple substrates with distinctly different dynamics using sim-to-real transfer of an RL agent trained purely in the simulation domain using dynamics randomization. We train three RL agent using data generated from a simple lumped-parameter model of the substrate energy balance with multiple parameter realizations. This both captures the variations in thermal dynamics of different substrates and the operational variability of APPJs. Each RL agent is exposed to a different degree of dynamics randomization, i.e, to a different set of model parameter realizations. We first test the RL agents in a simulation environment to asses their performance in temperature setpoint tracking across a broad model parameter space. The RL agent with the highest degree of dynamics randomization (i.e., trained with the most diverse set of model parameters) outperforms the others. We then experimentally compare the RL agents' ability to track temperature setpoints on glass, aluminum and polyimide substrates. We further demonstrate that the highest performing RL agent is also capable of rejecting step disturbances in jet-tip-to-substrate separation distance. Thus the sim-to-real transfer learning using dynamics randomization provides a successful method to address the challenges of controlling the uncertainties and variability in plasma-substrate dynamics.

2. Methods: Experimental setup

2.1. Atmospheric pressure plasma jet

The APPJ device used for experiments is shown in Figure 1. The APPJ consists of a quartz dielectric tube (ID = 3 mm, OD = 4 mm) and a powered copper ring electrode placed 1 cm from the tube nozzle. An aluminum plate acts as the ground and doubles as a conductive substrate. The treated substrates are placed on the aluminum plate under the plasma plume 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 PI controller, manipulating 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 spatial distribution of substrate temperature via a radiometric infra-red thermal camera (Lepton FLIR 3.5) pointed towards to the incident point of the APPJ. An example temperature measurement is presented in Figure 1c. For the purposes of this work we focus only the peak value of this distribution. The measurements and actuation are coordinated via a Wi-Fi enabled single board controller (Raspberry Pi 3). The sampling time is fixed at 1.3 s. The RL algorithms are implemented on a remote computer and measurement and actuation information are

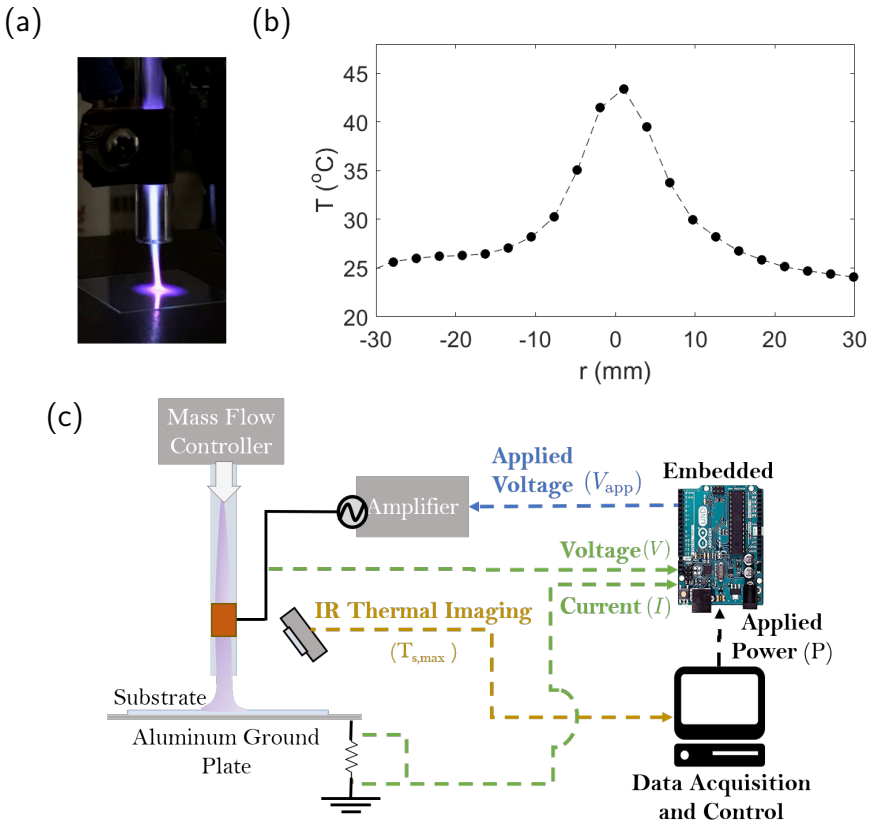


Figure 1: (a) Visual appearance of the APPJ, (b) an example temperature distribution on the glass substrate, and (c) the diagram of the APPJ setup used for experiments. The applied voltage V_{p2p} is manipulated to maintain the desired applied power P, computed based voltage and current measurement in via an embedded PI control. Data acquisition and control determines the desired P based on measured substrate temperature T_s computed by the RL agent.

exchanged via TCP/IP protocol over Wi-Fi.

2.2. Lumped-parameter model of thermal dynamics of substrates

The complexity of APPJ-substrate interactions and disparity of the timescales of physical phenomena make physics-based models of APPJs challenging to develop and time-consuming to solve. However, the dynamics of the thermal response of substrates to APPJs can be reasonably described by lumped-parameter models based on volume averaged mass and energy balances [17]. Here, we conduct a volume-averaged energy balance on the substrate in contact with the APPJ to describe the dynamics of the maximum substrate temperature T_s,

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

Table 1: Fitted lumped model parameters for three different substrates; borosilicate glass, aluminum plate, and polyimide tape.

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

Here, ρ, c_p and k are the density and heat capacity and thermal conductivity of borosilicate glass, our 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; η represents the fraction of power dissipated on the substrate; and T_∞ is the ambient temperature. $\bar{\mu}_1$ and $\bar{\mu}_2$ are the two lumped parameters fitted based on experimental data. We normalize these two parameters to the order of 1; $\bar{\mu}_1 = 38\mu_1$ and $\bar{\mu}_2 = 0.003\mu_2$ to facilitate analysis. We consider three substrates with varying dynamics: bare borosilicate glass coverslip, grounded aluminum plate, and borosilicate glass coverslip with polyimide tape on the surface. These model parameters μ_1 and μ_2 values fitted based based on dynamics observed in each substrate are summarized in Table 1. This lumped-parameter model is used to generate *in silico* training data for the reinforcement learning controller which is detailed in the following section.

3. Methods: Reinforcement Learning controller design

We aim to design an RL agent which dictates the APPJ input power P for achieving optimal setpoint tracking of substrate temperature under a range of substrate dynamics and subject to the inherent variability of APPJ operation. We generally refer to an agent that has been trained for this purpose as a reinforcement learning controller

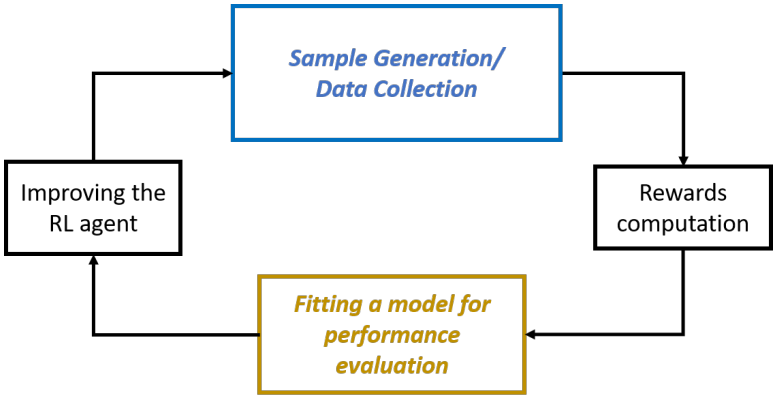


Figure 2: A simplified block diagram describing the key components of reinforcement learning

(RLC). Reinforcement learning generally relies on four key steps, shown in Figure 2, to train an agent: (i) generating simulated data and/or collecting live data to be used for training, (ii) evaluating the reward for the generated samples, based on the goal of the application, (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 section serves to review basic RL concepts and some key training specifics that highlight how to train a successful RLC for APPJ operation. More details on the RLC training can be found in the Appendix and the references contained within.

3.1. States, actions, and rewards for APPJ operation

Designing an RL agent requires defining the environment in which it will operate. This necessitates specifying the state of the environment 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. In our application, 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 temperature setpoint. Thus, the states for our application ($s_t \in \mathbb{R}^{2m}$) are the deviations from temperature setpoint and applied power for past m time steps. For this work, we choose $m = 3$. Employing the history of the operation as states allow the RLC to implicitly build its own process model [34]. The action to take 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, and so actions dictated by the RLC falling outside the bounds of $1.1\ W < P_t < 5.0\ W$ are truncated. Although the model ((1) is linear, the bounds on P introduces a nonlinearity which should be captured by the RL training process.

The reward function aims to quantify the goal of the operation. In this case, the reward function is designed to ensure the RLC maintains the temperature at the desired setpoint, T_s^{sp},

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

Here ϵ denotes a tolerance level, which corresponds to a setpoint tracking difference that we deem to be as good as the optimal value of 0. This reward value is designed to account for the fact that under realistic (i.e., experimental) conditions, process and measurement noise 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 for ϵ is chosen based on the observed noise level in the measurements (set to 0.1 in this work), but it can be modified based on the scale of acceptable deviations from T_s^{sp} for different applications.

3.2. Reinforcement learning control of APPJ operation

The RLC takes the mathematical form of what is known as a policy. This policy, $\pi(a|s)$, is a probability distribution over the action space conditioned on the states. In other words, when querying the policy, i.e. feeding s_t to the policy, the ANN outputs the parameters of a conditional distribution, from which the implemented action a_t is sampled. While the policy specifies how the RLC takes actions, in general, a state transition model $p(s_{t+1}|s_t, a_t)$ specifies a probabilistic description of how the environment's state evolves in response to the RLC's actions. The transition model is determined by the dynamics of the environment, which for APPJ operation is simply modeled by the substrate dynamics given by Eqn. 1. Nominally, this model generates deterministic transitions. Note that this model is simplistic and may deviate significantly from the true dynamics of the live APPJ environment, $p^*(s_{t+1}|s_t, a_t)$.

3.3. Goal of reinforcement learning control

Using the policy and the transition model, we can propagate the APPJ environment forward in time to generate simulated trajectories or "roll-outs" of time length T, denoted by a string 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 dynamics of APPJ and our RLC's policy. The objective during the learning procedure is to find some optimal policy, π^{opt} that maximizes the expected return of the agent, $J(\pi)$,

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

where,

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

In other words, we seek to maximize the expectation value of the rewards accumulated across the roll-outs of a given policy, and this quantity can be computed by sampling trajectories in the APPJ environment by Monte Carlo simulations.

3.4. Actor-critic algorithm

In this work, we use a variant of the policy gradient formulation [35, 36] in reinforcement learning, and dynamics randomization for sample generation [33] (see next section), to achieve effective substrate temperature control with the APPJ across a range of substrate dynamics. Specifically, we use the actor-critic algorithm which relies on using two ANNs, termed the *actor* and *critic* networks (Fig. 3), that are trained in tandem [24, 37]. The actor neural network, parameterized by weights θ, defines our policy $\pi_\theta(a|s)$.

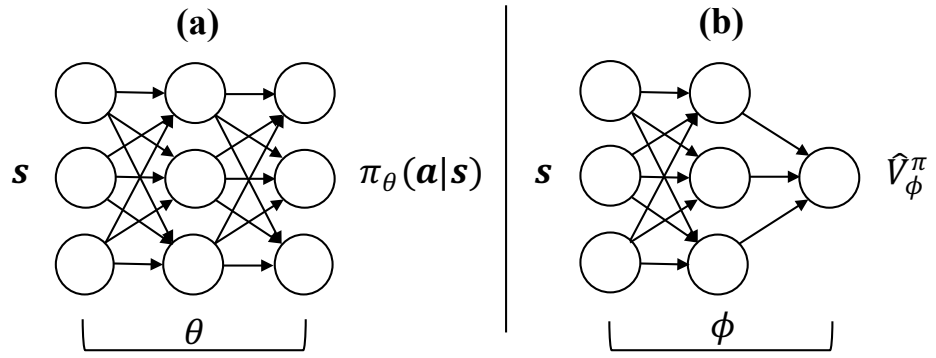


Figure 3: Generic schematic of an (a) actor policy and (b) a critic network.

The output layer of the actor network is typically a multivariate normal distribution with diagonal covariance matrix, but since $a_t \in \mathbb{R}$ in this application, our actor simply outputs the parameters of a single Gaussian distribution which is then sampled to yield an action. The second neural network, the critic network parameterized by weights ϕ, is used to approximate the value function, $\hat{V}_\phi^\pi(s)$. This value function is the expected value of future rewards from being at s under the current policy, or more colloquially, how valuable it is to be at a given state.

The weights θ and ϕ are trained to maximize the reinforcement learning objective (Eqn. 7) by the actor-critic algorithm, the details of which are presented in the Appendix. Briefly, the weights of the actor and the critic are both initialized randomly at the beginning of training. Thus both the actor and critic commence with poor performance and try to improve in their respective tasks (approximating the optimal decision making policy and the value function). The actions computed by the actor policy start by computing sub-optimal actions as it "explores" different behaviors. During each training iteration, simulated roll-outs are collected (see next section), rewarded, and a gradient step is 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 predicted action distributions shrinks as the policy becomes optimized to take actions that accumulate higher rewards. The deep ANNs for both the actor and the critic have two hidden layers with 64 nodes per layer. While more advanced variants of model-free RL were not needed to complete the objectives of this work, an interesting opportunity in the future is to see if schemes such as Generalized Advantage Estimate [38], Proximal Policy Optimization [39], or Soft Actor-Critic algorithms [40] may be necessary to succeed in more complex control problems in APPJs. The implementation of the algorithm and the plasma model environment are provided at `https://github.com/mwitman1/PlasmaRL`.

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 Eqn. 7 can be computed and

used to update the weights of the actor network (see Appendix for details). Each training epoch consists of collecting N simulated roll-outs, each of time length T, from the current policy (Eqn. 5) using MC sampling where the transition dynamics are given by the lumped parameter model described in Eqn. 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. Each roll-out consists of the response of the closed-loop (actor policy and the APPJ model) system to a single setpoint change. At the beginning of each roll-out a new setpoint is randomly chosen from the uniform distribution, $T_{sp} \sim \mathcal{U}[34, 46]$. Since the state vector contains previous setpoint deviations, it is recomputed with the new setpoint temperature at the start of each new roll-out.

The parameters of the lumped-parameter model (μ_1, μ_2) must also be specified in order to generate the training roll-outs. These lumped model parameters can be fit for a particular substrate to experimental data (see Table 1). However, this model remains a rough approximation (low-fidelity model) of the very complex physical processes occurring at the plasma/substrate interface. This makes training with dynamics randomization over (μ_1, μ_2), critical. With dynamics randomization [33], the reinforcement learning objective is now to maximize the expected rewards when the model parameters are no longer constant but sampled from a 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. In addition to the dynamics randomization, sample from a Gaussian white noise σ_T distribution and add it to the temperature dynamics to represent some uncertainty in the measurements. We hypothesize these efforts will permit successful sim-to-real transfer despite our low fidelity model. It will also allow us to create a single RL agent that is capable of maintaining good thermal control performance for various substrates in an experimental context since we can choose ρ_μ in a way that is representative of the range of dynamics expected for different substrates.

We train 3 RLCs in this work, each with its own ρ_μ as summarized in Table 2. The Glass RLC (G-RLC) is trained only using a single set of parameters which correspond to the values fitted for the borosilicate glass substrate (see Table 1) and no model noise. This is the base case with no dynamics randomization. The Glass Uncertainty RLC (GU-RLC) is trained drawing from a normally distributed parameter set centered around the parameters fitted to the borosilicate glass substrate and in the presence of additive Gaussian white noise, σ_T, to the model. Finally, the Ensemble RLC (E-RLC) is trained by discrete uniform sampling from an ensemble of 12 (μ_1, μ_2) parameter sets, which are chosen to be a representative selection of possible substrate dynamics, combined with the additive Gaussian white noise. This type of dynamics randomization allows us to explicitly define the range of temperature dynamics that we want our RLC to operate on. Some example temperature responses of the system with the different

Table 2: The parameter sampling scheme used to train the G-RLC, the GU-RLC, and the E-RLC.

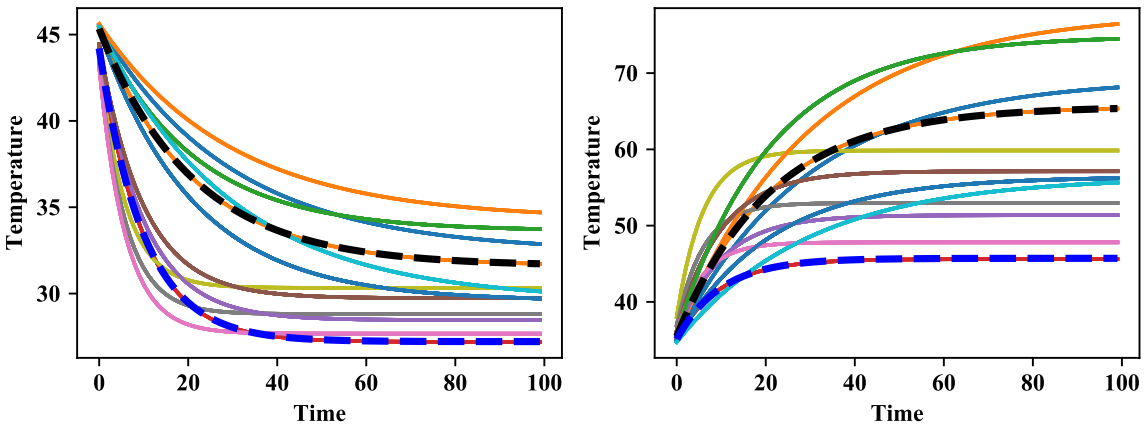
	μ_1	μ_2	σ_T
G-RLC	2.38	0.82	0.0
GU-RLC	$\mathcal{N}(2.38, 0.2)$	$\mathcal{N}(0.82, 0.05)$	0.11
E-RLC	$(\mu_1, \mu_2) \sim$ $\{(0.80, 0.40), (0.80, 0.55), (0.80, 0.65),$ $(1.12, 0.56), (1.12, 0.71), (1.12, 0.86),$ $(2.39, 0.82), (2.39, 1.02), (2.39, 1.22),$ $(4.00, 1.50), (4.00, 1.80), (4.00, 2.20)\}$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

parameters sampled during the training of the E-RLC are shown in Fig. 4.

4. Performance evaluation

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

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



(a) Cooling dynamics (b) Heating dynamics

Figure 4: Temperature response of the *in silico* energy balance 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. Each line corresponds to the response for one of the uniformly sampled (μ_1, μ_2) parameter sets used to train the E-RLC, and the thick dashed black and blue lines correspond to the fitted aluminum and glass dynamics, respectively.

and the cumulative input change as

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

which quantifies the control effort. We choose $b = 60$, neglecting the first 60 time steps in the trajectory in order to make the quantitative performance comparison between two different setpoint tracking experiments independent of the system's starting temperature. This helps minimize the effects of unavoidable day-to-day experimental variations in ambient temperature. High-performance operation corresponds to a low MAE and low CI. The setpoint tracking experiment upon which we evaluate these metrics can be seen in the following Results section.

5. 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 treated substrates it might encounter. Before validating our approach in real-time experiments, we can quantify its performance by evaluating the three RLC agents in a simulation environment on the *in silico* model.

5.1. Setpoint tracking

We first test the G-RLC *in silico* on the model of the borosilicate glass substrate (with $\sigma_T = 0$) via the closed-loop setpoint tracking simulation shown in Fig. 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. For this base case, then the performance of G-RLC is quantified by a $MAE = 0.16$ and a $CI = 69.2$.

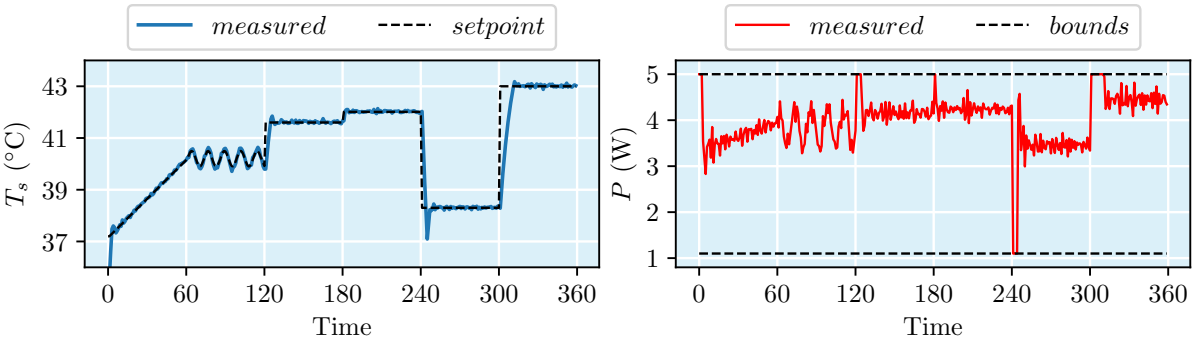


Figure 5: Control performance of the G-RLC is tested *in silico* on the model of the borosilicate glass substrate with $\sigma_T = 0.0$. The setpoint tracking performance (left) achieves an $MAE = 0.16$ and the power input (right) achieves $CI = 69.2$.

5.2. Improving controller performance

We use the setpoint profile in Fig. 5 to test the three RLC agents across a range of parameters and ambient temperatures for the dynamic model of the substrate temperature. 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 of randomization training improves control performance. Figures 6 and 7, show color surfaces of the MAE and CI for the three RLCs under different parameter set realizations (μ_1, μ_2, T_∞). Each point in these plots is obtained by conducting a closed-loop setpoint tracking simulation with $\sigma_T = 0.1$, where the parameters of the controlled system model are modified according to the values shown on the axes. In other words, each point represents the performance of the RLC agent on different substrate dynamics in *in silico*. In each plot, the parameter combinations corresponding to experimentally obtained values for glass, aluminum, and polyimide substrates are denoted by the light blue, black, and green stars (see Table 1). Note here, that given the limitations on the input P it is possible to track the desired setpoint profile in Fig. 5 only for certain combinations of dynamics at a given ambient temperature T_∞. The condition for unreachable setpoints corresponds to a high MAE and a low CI indicating that the input saturates at the bounds, yet the setpoint in temperature is not achieved. This manifests itself as the triangle-shaped regions in each of the plots in Figures 6 and 7.

Fig. 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 with the G-RLC. By training with additive noise and a small degree of parameter uncertainty, the control performance can be improved with the GU-RLC. However, the most significant improvement is achieved with the E-RLC. In other words, the RLC becomes especially more robust to changes in substrate dynamics that might have extremely fast temperature dynamics and/or large gains. Note that for $\mu_1/\mu_2 >> 1$ and $\mu_1/\mu_2 << 1$, poor performance (very high MAE) is observed for all of the RLCs since these parameters correspond to system dynamics in which the limit to the input power prevents effective tracking of the prescribed setpoints. Moreover, the reachable setpoints are a function of the ambient temperature T_∞. For example, on colder days (i.e., lower T_∞) it may be difficult to track high-temperature setpoints even when maximum P is applied.

To understand the systematic driver of the performance increase of the E-RLC, we plot in Fig. 7 the CI that corresponds to the same setpoint tracking simulations shown in Fig. 6. The region of drastic MAE improvement in Fig. 6 is also characterized by a large reduction in CI in Fig. 7. Even in the parameter space where a low MAE is observed in Fig. 6, there is a significant reduction in the CI with E-RLC. The E-RLC appears to have therefore learned a highly desirable control behavior whereby its setpoint tracking performance is improved while simultaneously decreasing the control effort across a broad range of dynamics. In order to predict the control performance

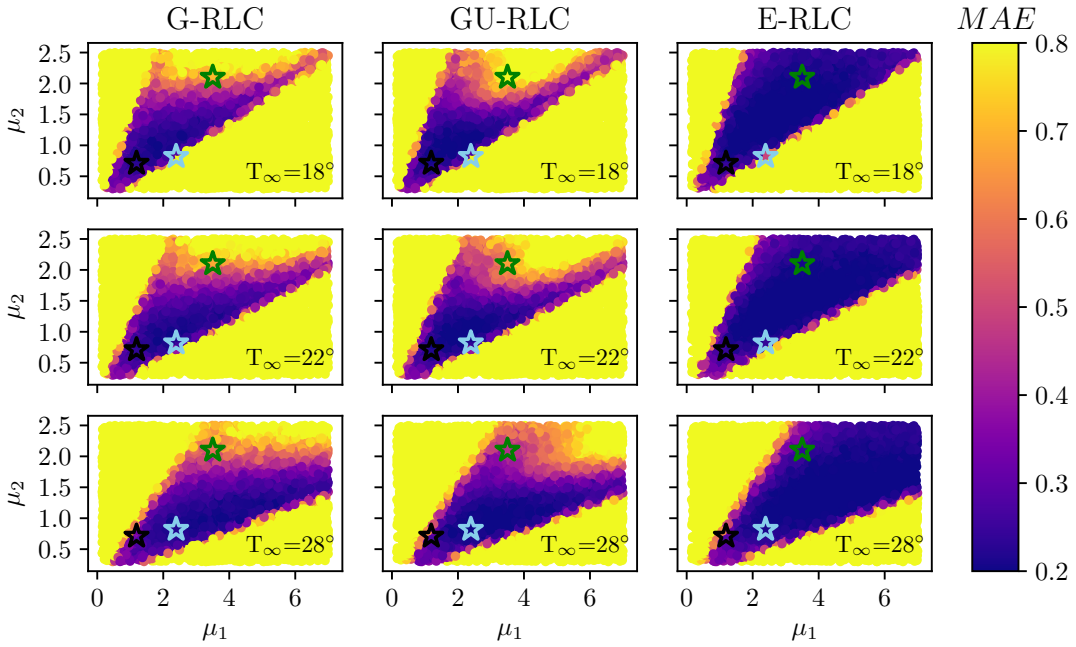


Figure 6: *In silico* evaluation of the performance of the 3 RLC controllers investigated in this work. Each subplot shows the MAE in the setpoint tracking experiment (with $\sigma_T = 0.1$) where the columns correspond to the different RLC agents and rows correspond to different ambient temperatures. Black, blue, and green stars respectively represent the model parameters fitted for aluminum, borosilicate glass, and polyimide substrates.

Table 3: Quantitative performance metrics for the G-RLC and E-RLC tested via *in silico* simulations on 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

on the experimentally available substrates, the quantitative performance metrics of the simulations performed using the corresponding model parameters are shown in Table 1. The time courses for these setpoint tracking simulations are visualized in Appendix D. Results summarized in Table 1 indicate that the E-RLC can provide a particularly significant performance increase on the polyimide substrate with a predicted 2.5-fold decrease in MAE and a 4-fold decrease in CI.

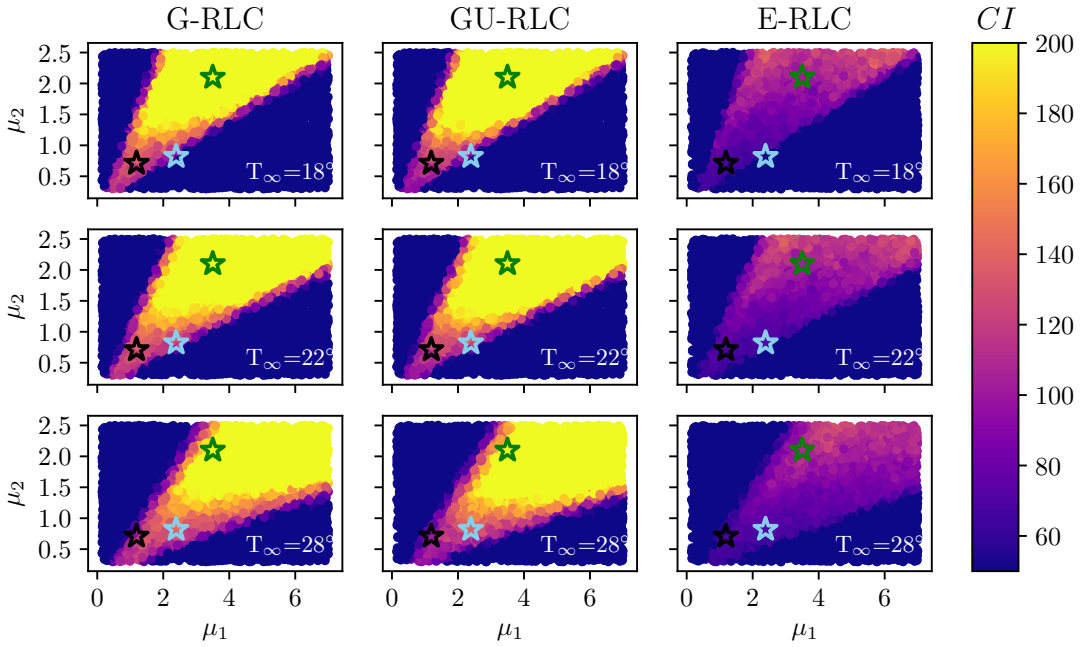


Figure 7: The cumulative input change, CI, for the control experiments shown in Fig. 6

6. Results: Real-time experiments

6.1. High performance control on different substrates

We can hypothesize that the high performance of the E-RLC that was demonstrated *in silico* indicates that, not only should good control performance be achieved with live APPJ operation, but improved performance should be achieved with the E-RLC over the G-RLC. To test this hypothesis, we perform real-time control experiments using both the G-RLC and E-RLC on APPJ setup described in section 2.1. We test the RLCs on three different substrates: borosilicate glass, aluminum, and polyimide. Experimental results obtained for setpoint experiments are shown in Fig. 8.

The results demonstrate that the control performance of the E-RLC is notably better than the control performance of the G-RLC over the three investigated substrates. While the G-RLC performs well on the borosilicate glass substrate (Fig. 8a), whose dynamics are similar to what it was trained on, the E-RLC (Fig. 8b) performs quantitatively better with a 33% reduction in the MAE (see Table 4). The presence of unmodeled phenomena in the experimental setup, not captured by the simple lumped-parameter model, as well as the uncertainty associated with the fitting procedure for μ_1 and μ_2 parameters, results in deterioration in the performance of the G-RLC compared to simulation studies. Small amplitude oscillatory input patterns and small setpoint overshoot behavior are observed when the G-RLC encounters dynamics that are not *exactly* what the agent was trained on. Meanwhile, the E-RLC provides excellent

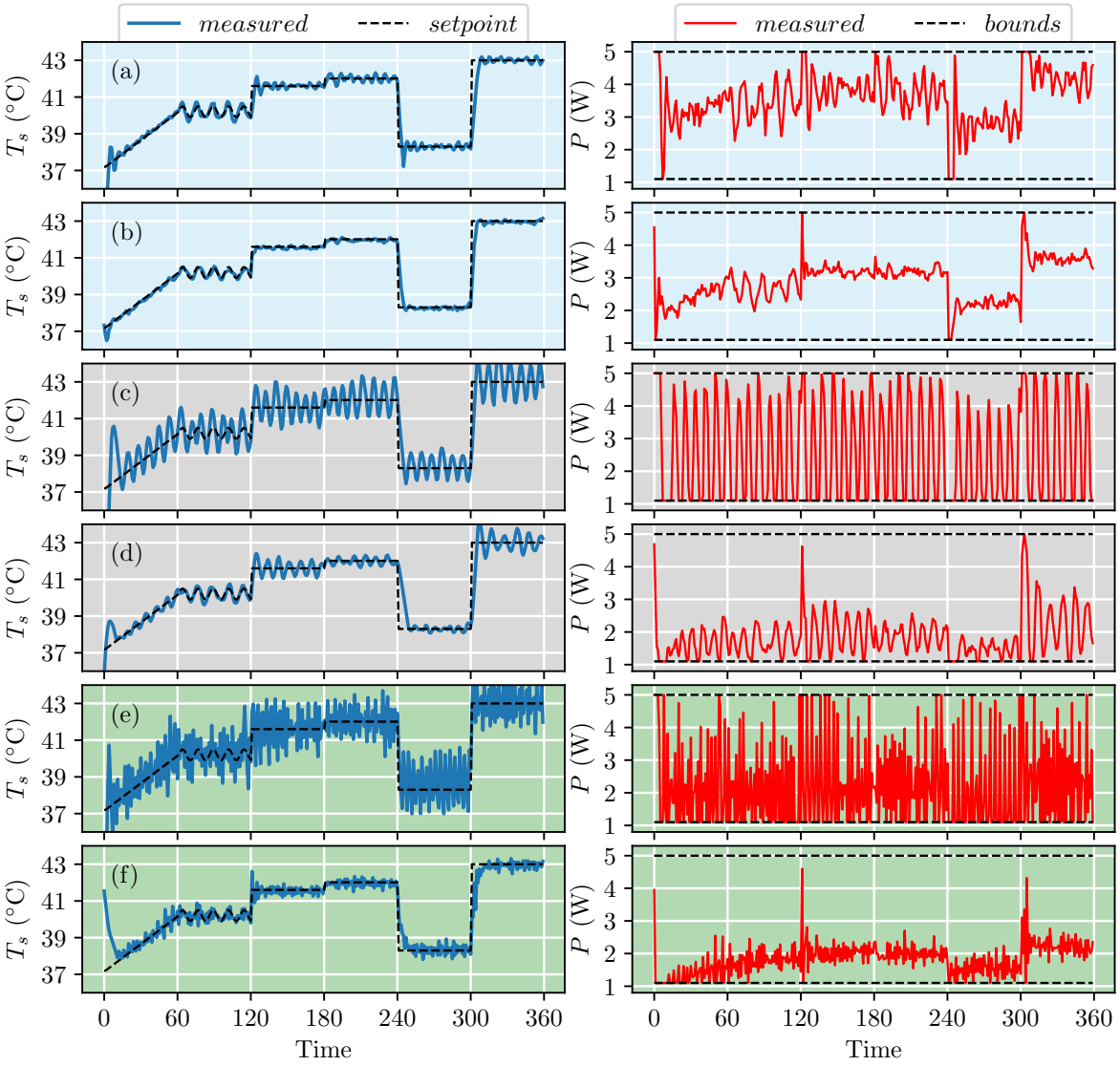


Figure 8: The setpoint tracking control performance of the G-RLC (a,c,e) and E-RLC (b,d,f) recorded in real-time experiments treating a borosilicate glass (a,b), aluminum (c,d), and polyimide (d,f) substrate. The E-RLC's control performance is significantly better for each individual substrate compared to the G-RLC as quantified by its setpoint tracking error (left column) and control effort (right column).

setpoint tracking performance with low control effort, comparable to the base case in 5. Moving to aluminum and polyimide substrates presents even more difficulty since the temperature dynamics of both substrates can be characterized by very large process gains with relatively slow dynamics for aluminum and relatively fast dynamics for polyimide. For both substrates, the G-RLC (Fig. 8c,e) struggles to maintain the setpoint, as indicated by large amplitude oscillations in the input profile, while the E-RLC performs much more robust setpoint tracking (Fig. 8d,f). Quantitative evaluation of the control performance from Fig. 8 is summarized in Table 4. The E-

Table 4: Quantitative performance metrics for the G-RLC and E-RLC tested with the live APPJ on 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

RLC demonstrates significant quantitative improvement for setpoint tracking on all three substrates while also drastically reducing its control effort.

6.2. Disturbance rejection

Lastly, in order to test the capability of the E-RLC for temperature control beyond the setting on which it was trained (i.e., responding to setpoint changes), we introduce a disturbance to the APJ operation. Changes in the separation distance (the distance between the jet tip and substrate) present a relevant disturbance for applications of APPJs. Changes in substrate topology or variations during hand-held treatment can impact the separation distance which in turn has a considerable effect on thermal dynamics [9, 10, 16]. If to be useful in real-world applications, the E-RLC should be able to effectively maintain constant temperature setpoint tracking when the separation distance suddenly changes. Fig. 9 shows the performance of E-RLC in regulating substrate temperature as the separation distance is increased by 2 mm in a step-wise manner. These results indicate that the E-RLC is able to quickly recover and maintain effective setpoint tracking after such a disturbance. Note in Figure 9 the scale of the y-axis only shows $\pm$ 1 ^{o}C; the excursions of the substrate temperature are indeed are on the order of only 0.5 ^{o}C. At very high separation distances (>10 mm), a fluctuating response is observed both in temperature and the applied power. This is due to the fact that at this separation distance, the discharge is on the verge of de-coupling from the substrate and exhibits unstable behavior. In this region, the available actuation is not sufficient to effectively regulate thermal behavior.

7. Discussion

In this work, we have demonstrated how dynamics randomization and additive noise in simulated training data can be used to obtain a high-performing RLC across a variety of substrates that exhibit greatly different temperature dynamics. The broad range of system dynamics is accommodated with a simple model and a range of parameters allowing for improved control performance. Without using these transfer learning techniques, we have shown *in silico* how the RLC is particularly ineffective when it

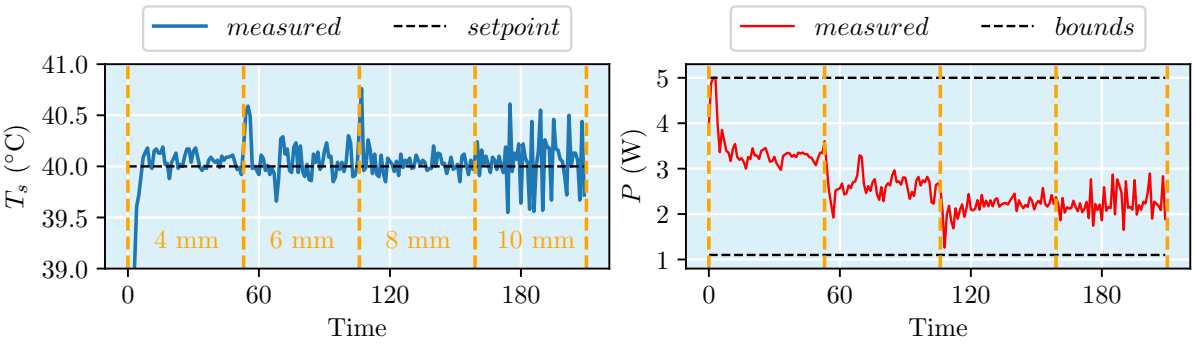


Figure 9: The E-RLC performs constant temperature setpoint tracking while the jet/substrate separation distance is perturbed at regular time intervals from 4 mm, 6 mm, 8 mm, to 10 mm.

encounters a substrate which responds much faster than the dynamics the agent was trained on. The E-RLC also displays more robust control performance on all substrates we tested both *in silico* and in real-time experiments. This demonstrates both the power and the necessity of domain randomization for sim-to-real transfer of an RLC for APPJ operation. This has allowed us to address two of the main challenges in APPJ control: operational variability and widely differing dynamics of substrates' plasma interactions.

Our purely *in silico* training procedure and subsequent successful transfer to *in vivo* operation is important for several reasons. Firstly, the *in silico* training is orders of magnitude faster compared to using data collected from real-time 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 purely live data training procedure would take ∼75 days of non-stop APPJ operation, whereas the *in silico* training is completed on the order of 0.075 days. An interesting future project will be to work with more sample efficient RL algorithms to see if the training can be accomplished only with live data; however, using only live training to create a robust RLC would quickly face another barrier, even if sample efficiency were not an issue. In addition to alleviating efficiency concerns by using *in silico* training, the facile selection of an ensemble of model parameters *in silico* is not so easily replicated in the live environment. To perform *in vivo* training with dynamics randomization, one would need time-consuming manual experimentation to find a representative set of substrates (or tweak other parameters of the experimental setup) that sufficiently spans the space of model dynamics one wishes to train over. In a best-case scenario it will be inconvenient (and in the worst case impossible) to tweak the *in vivo* experimental conditions to cover the entire model parameter space that the RLC may need to operate within during run time. Finally, a key requirement of APPJ treatment is safety, particularly in a medical context where the substrate is a patient receiving treatment. Training RL agents with real-time experiments can result in thermal damage on the treated substrate, especially in early iterations, making this approach prohibitive. By avoiding any *in vivo* training, we can ensure stable and high-

performing *in silico* performance before applying the RL controller in the real world.

An attractive feature of our presented approach is the use of a low-fidelity model of the substrate temperature dynamics in conjunction with dynamics randomization. While the speed-up in training allowed with a low-fidelity model is appreciable and dynamics randomization, in our case, allows accommodating a range of experimentally observed dynamics, there may be limitations to using low-fidelity models. Particularly, we restricted our model structure to be linear, whereas APPJs can exhibit a range nonlinear behaviors. Indeed, mildly nonlinear temperature dynamics are observed on the aluminum substrate, which resulted in a somewhat oscillatory temperature response even with E-RLC strategy. Thus future work may involve using more complex APPJ models to generate training data. The methods we present in this paper can readily generalize to 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 this project to explore the potential role of RL in the control and dose delivery of plasma medicine. For example, one task that is highly important to the plasma medicine community is that of optimal dose delivery. In this problem, 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 in this context as lower level controllers for fast disturbance rejection and allowing effective regulation of across substrates with varying thermal dynamics. Moreover, RL methods can be useful in analyzing and integrating more complex sensory information such as optical emission spectra and current waveforms for diagnostics applications [41], and incorporate further manipulated inputs for control problems. The comparatively simple demonstration in this work indicates the vast potential of RL strategies in complementing and, in some instances, replacing classical and advanced control strategies for effective treatment delivery with APPJs.

Acknowledgments

For Berend: Do we acknowledge EFRC here for funding and/or something else?

For Dogan/David/Ali: please add funding sources as necessary

References

(1) Winter, J.; Brandenburg, R.; Weltmann, K.-D. *Plasma Sources Science and Technology* **2015**, *24*, 064001.

(2) Laroussi, M.; Leipold, F. *International Journal of Mass Spectrometry* **2004**, *233*, 81–86.

(3) Morent, R. *The Open Plasma Physics Journal* **2013**, *7*, 6.

(4) Siow, K. S.; Britcher, L.; Kumar, S.; Griesser, H. J. *Plasma Processes and Polymers* **2006**, *3*, 392–418.

(5) Von Woedtke, T.; Metelmann, H.-R.; Weltmann, K.-D. *Contributions to Plasma Physics* **2014**, *54*, 104–117.

(6) Metelmann, H. R.; Seebauer, C.; Rutkowski, R.; Schuster, M.; Bekeschus, S.; Metelmann, P. *Contributions to Plasma Physics* **2018**, *58*, 1–5.

(7) Laroussi, M.; Kong, M.; Morfill, G.; Stolz, W., *Plasma Medicine*; Cambridge University Press: 2012.

(8) Weltmann, K. D.; von Woedtke, T. *Plasma Physics and Controlled Fusion* **2016**, *59*, 014031.

(9) Breden, D.; Raja, L. L. *Plasma Sources Science and Technology* **2014**, *23*, 065020.

(10) Wu, S.; Wand, Z.; Huang, Q.; Lu, X.; Pan, Y. *IEEE Transactions on Plasma Science* **2011**, *39*, 1489–1495.

(11) Norberg, S. A.; Johnsen, E.; Kushner, M. J. *Journal of Applied Physics* **2015**, *118*, 1–13.

(12) Klarenaar, B. L.; Guaitella, O.; Engeln, R.; Sobota, A. *Plasma Sources Science and Technology* **2018**, *27*.

(13) Ji, L.; Yan, W.; Xia, Y.; Liu, D. *Journal of Applied Physics* **2018**, *123*.

(14) Shin, J.; Raja, L. L. *J. Phys. D. Appl. Phys.* **2007**, *40*, 3145–3154.

(15) Dobrynin, D.; Wu, A.; Kalghatgi, S.; Park, S.; Shainsky, N.; Wasko, K.; Dumani, E.; Ownbey, R.; Joshi, S.; Sensenig, R.; Brooks, A. D. *Plasma Medicine* **2011**, *1*, 93–108.

(16) Gidon, D.; Graves, D. B.; Mesbah, A. *Proc. Am. Control Conf.* **2016**, *2016-July*, 4889–4894.

(17) Gidon, D.; Graves, D. B.; Mesbah, A. *Plasma Sources Sci. Technol.* **2017**, *26*, 085005.

(18) Gidon, D.; Curtis, B.; Paulson, J. A.; Graves, D. B.; Mesbah, A. *IEEE Trans. Radiat. Plasma Med. Sci.* **2017**, *2*, 1–1.

(19) Gidon, D.; Graves, D. B.; Mesbah, A. *Plasma Sources Science and Technology* **2019**, *28*.

(20) Balcon, N.; Hagelaar, G. J. M.; Boeuf, J. P. *IEEE Transactions on Plasma Science* **2008**, *36*, 2782–2787.

(21) Sutton, R. S.; Barto, A. G., *Reinforcement Learning: An Introduction*; The MIT Press: Cambridge, Massachusetts, 2015.

(22) Mnih, V. et al. *Nature* **2015**, *518*, 529–533.

(23) Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. **2015**, *2*, 1–127.

(24) Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; Zhokhov, P. *Openai baselines, GitHub repository* **2017**.

(25) Hoskins, J.; Himmelblau, D. *Comput. Chem. Eng.* **1992**, *16*, 241–251.

(26) Anderson, C. W.; Hittle, D. C.; Katz, A. D.; Kretchmar, R. *Artif. Intell. Eng.* **1997**, *11*, 421–429.

(27) Govindhasamy, J.; McLoone, S.; Irwin, G. In *2004 2nd Int. IEEE Conf. 'Intelligent Syst. Proc. (IEEE Cat. No.04EX791)*, IEEE: 2004; Vol. 1, pp 316–321.

(28) Syafiie, S.; Tadeo, F.; Martinez, E. In *Reinf. Learn.* January; I-Tech Education and Publishing: 2008.

(29) Spielberg, S.; Gopaluni, R.; Loewen, P. In *6th Int. Symp. Adv. Control Ind. Process.* 2017.

(30) Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. In *2017 IEEE/RSJ Int. Conf. Intell. Robot. Syst.* IEEE: 2017, pp 23–30.

(31) Rajeswaran, A.; Ghotra, S.; Levine, S.; Ravindran, B. *CoRR* **2016**, *abs/1610.01283*.

(32) Sadeghi, F.; Levine, S. *CoRR* **2016**, *abs/1611.04201*.

(33) Peng, X. B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. *CoRR* **2017**, *abs/1710.06537*.

(34) Spielberg, N. A.; Brown, M.; Kapania, N. R.; Kegelman, J. C.; Gerdes, J. C. *Science Robotics* **2019**, *4*, DOI: 10.1126/scirobotics.aaw1975.

(35) Sutton, R. S.; McAllester, D.; Singh, S.; Mansour, Y. In *Proc. 12th Int. Conf. Neural Inf. Process. Syst.* 1999, pp 1057–1063.

(36) Konda, V. R.; Tsitsiklis, J. N. In *Adv. Neural Inf. Process. Syst.* 2000.

(37) Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. In *International conference on machine learning*, 2016, pp 1928–1937.

(38) Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; Abbeel, P. In *ICLR*, 2016.

(39) Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. *CoRR* **2017**, *abs/1707.06347*.

(40) Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. *CoRR* **2018**, *abs/1801.01290*.

(41) Gidon, D.; Pei, X.; Bonzanini, A. D.; Graves, D.; Mesbah, A. *IEEE Transactions in Radition Science and Plasma Medicine*.

Appendix A. Details of the actor-critic algorithm

The mathematical details of the actor-critic algorithm are presented in this section; however, it is assumed the reader has read the methods in the main manuscript. Several actor-critic variants of policy gradient exist [21, 35], 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 [38],

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

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. [38] 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.

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

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

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

Once Eqn. A.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 shown in Fig. A1. Note that Fig. A1 is the actor-critic specific realization of the more general reinforcement block flow diagram shown in Fig. 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

Algorithm 1 Online, two network actor-critic

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

distribution, $T_{sp} \sim \mathcal{U}[34, 46]$ and trajectories are propagated forward in time based on the transition model and actor policy. This can be expressed succinctly as collecting $N \times T$ tuples of (s_t, a_t, s_{t+1}, r) 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.

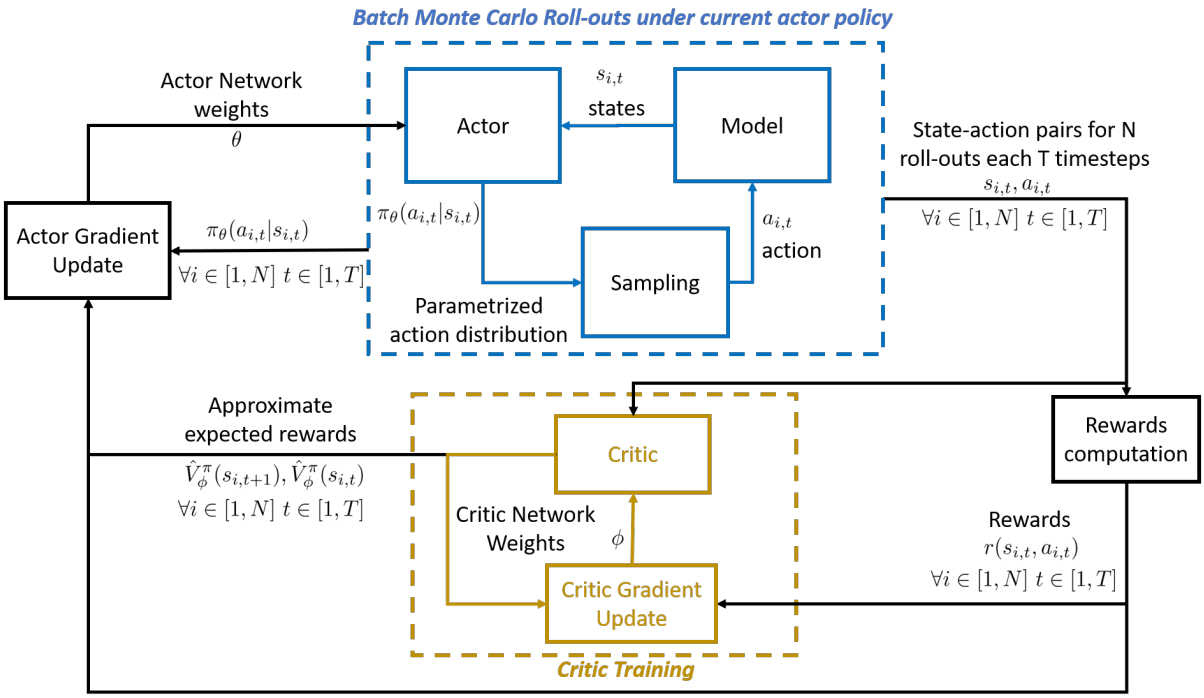


Figure A1: A block diagram representation of the actor-critic algorithm for training the RLC

2. Critic training:

The goal of the critic network is to approximate the value function with weights ϕ, $\hat{V}_{\phi}^{\pi}(s_t)$. 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, $\hat{V}_{\phi}^{\pi}(s_t)$, and its target values, z_t as shown in Algorithm 1, are formulated as the critic network's loss and minimized by optimizing the weights of the critic network:

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_{i} ||z_i - \hat{V}_{\phi}^{\pi}(s_i)||^2 \tag{A.4}$$

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 policy data, the policy gradient can be computed via Eqn. A.3 and a gradient step computed in θ (with learning rate α) to improve the policy for the next iteration of data collection.

Appendix B. Learning curves

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 Fig. B1. The average return per MC roll-out is plotted versus actor training iteration. The G-RLC quickly learns a 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 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 dynamics (see manuscript), and the limitations of these slow dynamics combined with a bounded action space reduce the minimum setpoint tracking error that can theoretically be achieved. This is also visually demonstrated in the next section. 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.

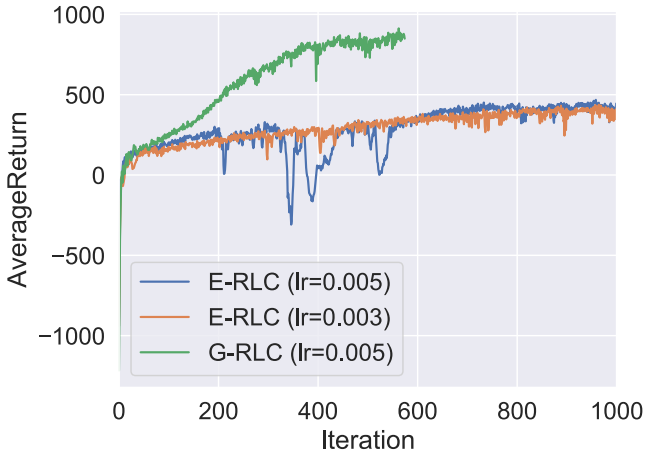


Figure B1: Training curves for the G-RLC with a learning rate of 0.005 and the E-RLC with both a learning rate 0.005 and 0.003.

Table C1: Parameters of the lumped-parameter 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}

Appendix C. Lumped-parameter model predictions versus experiment

Fig. C1 shows the predicted temperature at the next time step from the lumped-parameter model, T^m_{t+1}, with glass fitted parameters versus the actual temperature at the next time step, T_{t+1}, for the set point tracking experiment from Fig. C1a (the real-time glass control experiment). Fig. C1a plots the error in the model predictions versus the jet power at every time step with the color code corresponding to the change in power from the previous time step. This demonstrates that the largest errors in model prediction typically correspond when the jet is operating at large input changes. Fig. C1b shows the error in model predictions in histogram form with $\sigma_T = 0.17$ for this particular experiment.

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

In silico setpoint tracking simulations of the 3 RLC controllers are shown in Fig. D1, whose quantitative performance metrics were presented in Table 3. The largest performance gain with the E-RLC occurs on the polyimide substrate model (Fig. D1f). Notably from Fig. 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 Fig. D1c-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 will be demonstrated in the next section.

Appendix E. Implementation availability

The code used to train and execute the RLC's shown in this work are provided at `https://github.com/mwitman1/PlasmaRL` which extends the code provided by the instructors of CS294-112 at UC Berkeley (`http://rail.eecs.berkeley.edu/deeprlcourse/`). To train the E-RLC, use the command:

```
    python train_ac_f18_plasma.py PlasmaModel -ep 100 -epext 1 --discount
0.99 -n 3000 -e 1 -l 2 -s 64 -b 10000 -lr 0.005 --exp_name anyname -ntu 10
```

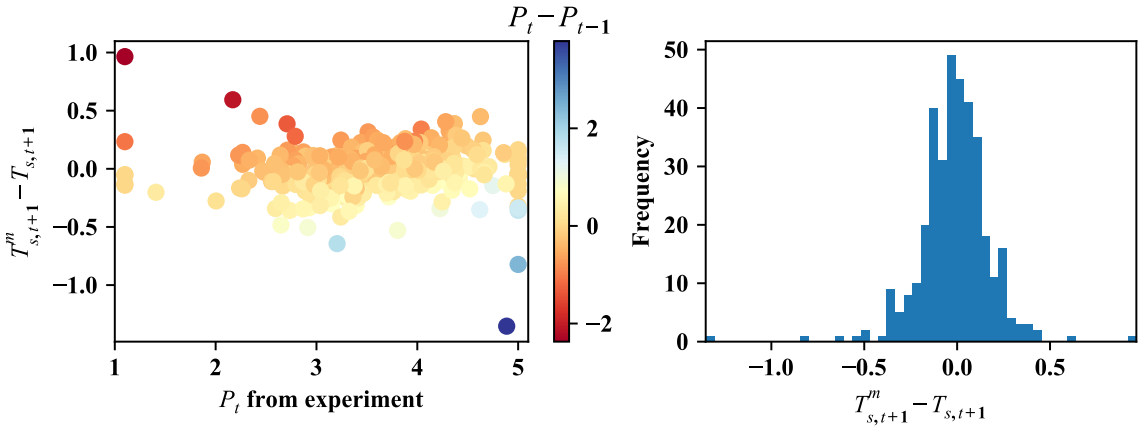


Figure C1: (Left) Deviations of the lumped-parameter model prediction from the actual temperature for the real time glass temperature control experiment from Fig. 6a. Deviations are plotted as a function of the jet power at each time step, and the color-coding corresponds to the change in the applied input.

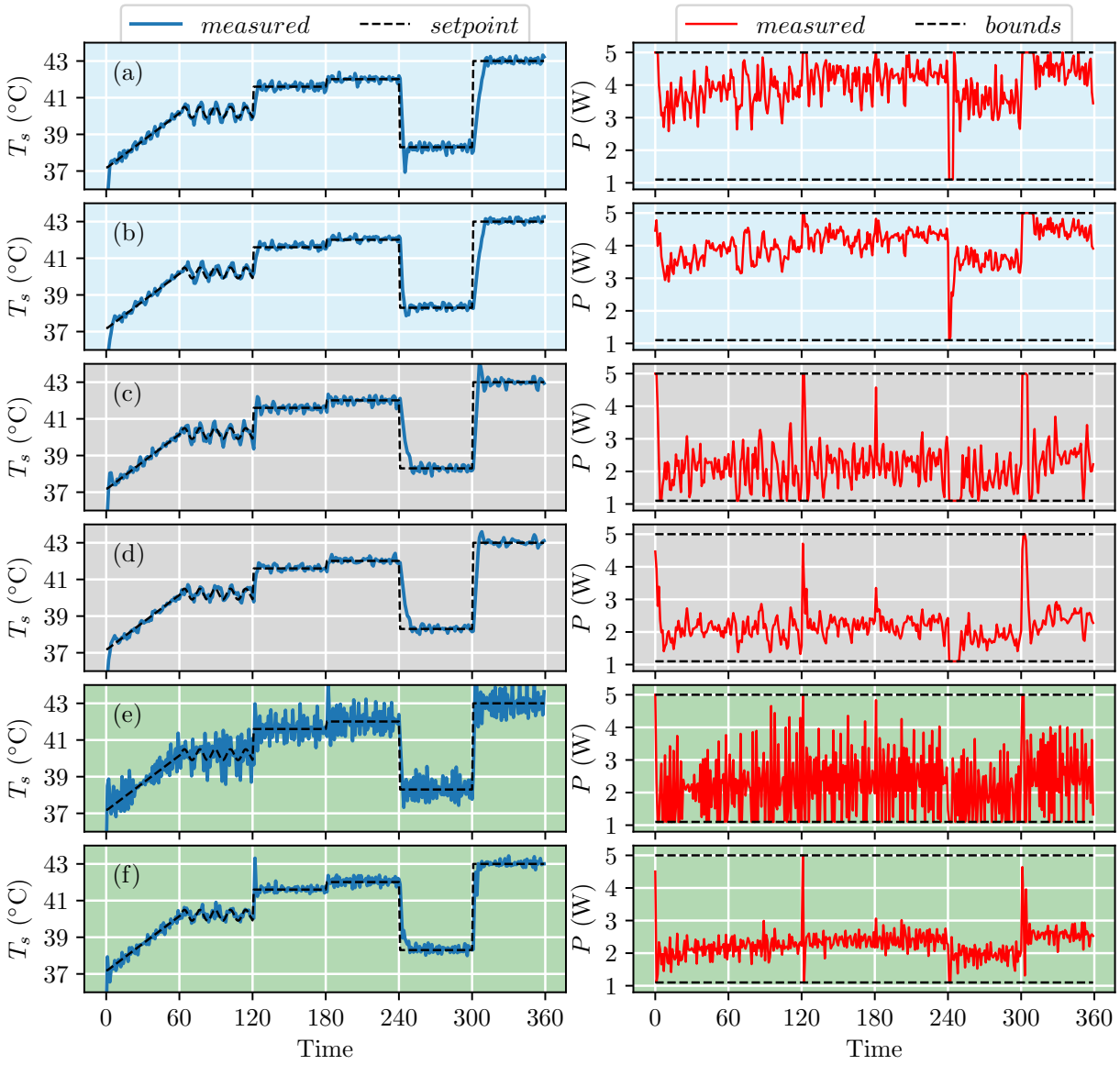


Figure D1: The setpoint tracking control performance of the G-RLC (a,c,e) and E-RLC (b,d,f) are tested on the *in silico* model for the parameters fitted to borosilicate glass (a,b), aluminum (c,d), and polyimide (e,f) substrates. The E-RLC's MAE is reduced for the set point tracking experiments (left column) and the corresponding power inputs (right column) have a reduced CI compared to the G-RLC (drastically so for polyimide).

`-ngsptu 10`