Model-free Tracking Control of an Optical Fiber Drawing Process using Deep Reinforcement Learning

by

Sangwoon Kim

B.S., Seoul National University (2018)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

© Massachusetts Institute of Technology 2020. All rights reserved.

Author .....
Department of Mechanical Engineering
May 15, 2020

Certified by .....
Brian W. Anthony
Principal Research Scientist, Department of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Nicolas Hadjiconstantinou
Graduate Officer

Model-free Tracking Control of an Optical Fiber Drawing Process using Deep Reinforcement Learning

by

Sangwoon Kim

Submitted to the Department of Mechanical Engineering
on May 15, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

A deep reinforcement learning (DRL) approach for tracking control of an optical fiber drawing process is developed and evaluated. The DRL-based control is capable of regulating the fiber diameter to track either steady or varying reference trajectories in the presence of stochasticity and non-linear delayed dynamics of the system. With about 3.5 hours of real-time training, it outperformed other control models such as open-loop control, proportional-integral (PI) control, and quadratic dynamic matrix control (QDMC) in terms of diameter error. It does not require analytical or numerical model of the system dynamics unlike model-based approaches such as linear-quadratic regulator (LQR) or model predictive control (MPC). It can also track reference trajectories that it has never experienced in the training process.[1]

Thesis Supervisor: Brian W. Anthony
Title: Principal Research Scientist, Department of Mechanical Engineering

[1]Part of this thesis is based on [1] with permission

Acknowledgments

This thesis would have never been possible without the support of many individuals along the way. I would like to take the opportunity to appreciate every one of them.

First of all, I would like to thank my parents, Woongchul Kim and Eunsook Hong, and my sisters, Jia Kim and Jiyun Kim, for supporting me throughout my entire lifetime. Your love and care gave me the power to overcome hardships and difficulties. Thank you for always being there and guiding me in the right direction.

Dr. Brian Anthony, my thesis advisor, helped me to come up with novel ideas and make breakthroughs when my research is at a stalemate. The precious discussion with you made me become a better researcher. Thank you for your guidance and for being always supportive.

I want to thank every labmate, especially David Donghyun Kim and Shirley Lu. The active discussion with you on the fiber machine has been the strongest driving force for my research. Without the interaction, my research idea would have never been realized.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Tracking Control of Manufacturing Processes

The control of a manufacturing process is instrumental in achieving consistent quality of materials or products at the desired rate. The design of an appropriate control system is necessary to achieve the desired outcomes. One important requirement for such a control system is to regulate output variables to track the desired reference trajectories. The reference trajectories can be either steady or dynamically varying. For example, in chemical processes, concentration or flowrate of the chemicals should follow the target reference trajectories. In thermal processes, temperature and airflow should be controlled to track certain trajectories. Conventional proportional-integral-derivative (PID) controller is widely used for the tracking control. Advanced approaches like linear-quadratic regulator (LQR) or model predictive control (MPC) are also often used.

In PID control, the controller takes measurements from sensors as input and computes the control action based on the error between the measurement and the target output. It is one of the most widespread controllers in industrial practice due to its simple structure and ease of implementation. Since it leverages only the difference between the actual value and the target value, it does not require a model of the system. It works well in many control problems, especially when the desired setpoint is steady. However, it requires manual tuning whenever the setpoint changes.

Control Method	Predictive	Analytical Model
PID	no	not required
LQR	no	required
MPC	yes	required
Model-free RL	yes	not required

Table 1.1: Comparison between PID, LQR, MPC, and Model-free RL

Also, since it does not look forward but only sees the error at the present and the past, it yields significant delay when the desired reference dynamically varies faster than its response bandwidth. These limitations of PID control can be improved with the model-based approaches like LQR or MPC.

While PID controller focuses on reducing the error between the output and the desired reference, LQR represents the system as a state-space model and optimizes to stabilize the system to the desired state. This generally enables a more stable control for LQR over PID controllers. While P, I, D parameters need to be tuned in PID controller, LQR optimizes the control by minimizing a cost function. Therefore, the operators only need to define the cost function based on their control objective and do not need to retune the parameters when the setpoint changes. However, a model of the system is necessary to establish the state-space model and the performance of the controller is sensitive to model accuracy. Also, since LQR is interested in stabilizing the system to a certain desired state, it cannot predictively control the system when the future reference trajectory is dynamically varying.

Unlike PID and LQR, MPC can predictively track the future reference trajectory. Using the system model, MPC predicts the output trajectory during a certain time-horizon and computes the error between the predicted trajectory and the reference trajectory. Then it computes the optimal control plan to minimize the error and exerts the first control action of the control plan. The above process of predicting and optimizing is repeated at every timestep. Thereby, MPC can minimize the expected future error between the actual and the reference trajectory. Since the prediction is based on a system model, accurate modeling of the system by either analytic or empirical analysis is necessary. Therefore, when the accurate model of the system is

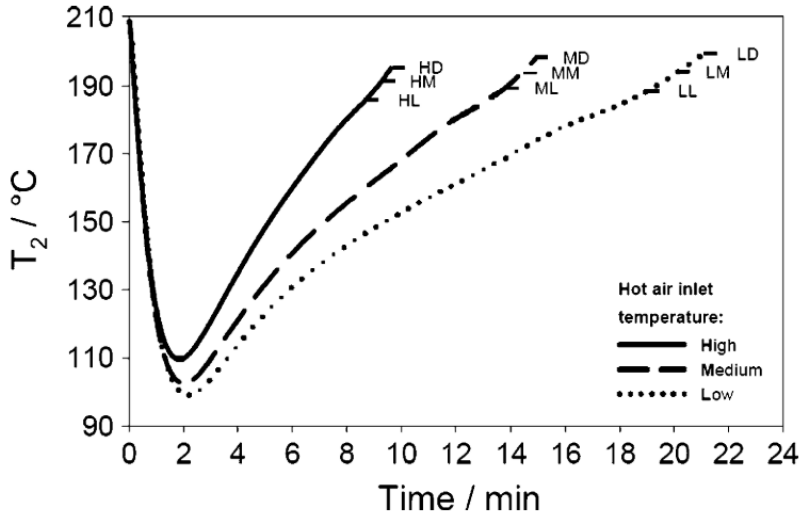


Figure 1-1: A typical time-temperature profile during the coffee roasting process (Adapted from Wieland et al.[2] with permission)

unavailable due to the high complexity of the system, it is hard to implement MPC.

As described above, PID and LQR are not predictive, and LQR and MPC are hard to implement when there is no accurate model of the system. Therefore, when the model of the system is hard to be achieved, PID, LQR and MPC cannot be used to regulate the output variables to a dynamically varying reference trajectory. For example, in a coffee roasting process, the quality of the coffee is defined by the temperature trajectory during the roasting process. As shown in Figure 1-1, the temperature follows a typical type of profile during the roasting process. Industry coffee roasters usually rely on a PID controller or rule of thumb to regulate the temperature to track the reference trajectory. However, since there is a time delay between the burner level change and temperature change, the tracking error occurs when a PID controller is used. Model-based methods such as LQR and MPC are also hard to be used because the process involves complex heat and mass transfer. The complexity of the process makes it hard to establish an analytical or numerical model. In this work, we use deep reinforcement learning (DRL) to develop a model-free and predictive control system for manufacturing processes that require varying reference target trajectory and involve complex non-linear delayed dynamics. We develop and deploy the control system on the optical fiber drawing process, focusing on regulating

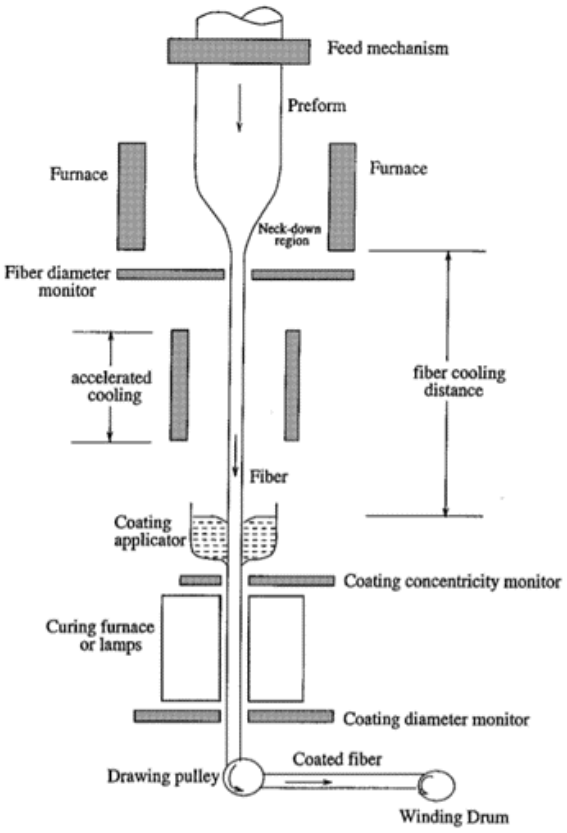


Figure 1-2: A schematics of the fiber drawing process (Adapted from Choudhury et al.[3])

to track time-varying reference trajectories for fiber diameter.

1.2 Optical Fiber Drawing Process

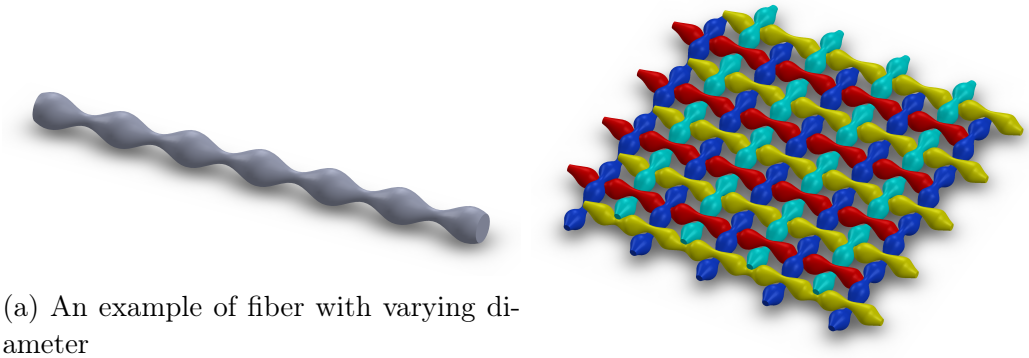
Figure 1-2 shows a schematics of the typical optical fiber drawing process. A preform rod with a large diameter is fed into the heater at the upstream. The preform is heated and pulled to the downstream. This causes the material to form the neck-down profile and transform it into a small diameter fiber. The fiber then goes through the cooling process and routed on the pulley to a winding drum.

A number of studies have modeled the neck-down shape and temperature distribution. Early models of the neck-down shape and temperature distribution were validated experimentally [4]. The initial models were extended for non-isothermal conditions [5]. The modern approach using iterative methods to improve the model

were introduced [3]. Models were improved by incorporating the physics and properties of the drawing process including the gas flow and iris opening sizes [6, 7]. There was also research done to simplify the model [8], along with research to enable high-speed drawing as well [9, 10]. The fiber process models were further augmented to include stochastic characteristics [11] and to evaluate parameters critical to stabilizing the fiber diameter [12].

In addition to the numerical model of the process, control of the fiber diameter has been studied. While control of the industrial fiber draw tower heavily relies on classical PID feedback control, more advanced model-based control approaches were studied. Most of them focused on maintaining a steady diameter. Mulpur and Thompson developed a modal diameter control method based on simulation [13]. They assumed isothermal temperature profiles and utilized a modal control method. They also developed nonlinear control on the optical fiber diameter [14]. State-space modeling of the optical fiber drawing process coupled with linear-quadratic Gaussian optimal controllers was investigated [15, 16]. Improved models of the neck-down profile and control of the draw tension enabled high-speed production [17]. Reduced-order models were coupled with robust control methods [18]. This long history of modeling and controlling the optical fiber manufacturing process focuses on maintaining the diameter at a fixed set point.

When the reference diameter trajectory varies over time, conventional PID controllers or model-based controllers mentioned above are either hard to implement or yield non-optimal performance. The PID controllers compute control actions based on only the present and past error between actual and the target diameter. Therefore, it results in a significant error when the target reference changes faster than the bandwidth of the PID controllers. In the model-based methods, new state models at new setpoints are required when the setpoints change [15]. A dynamic model of the transition between different setpoints is also required. However, the model for the dynamic transition is hard to be achieved because it requires considering complex non-linear dynamics caused by mass flow, rheology, fluid dynamics, heat transfer, vibration, etc. In this work, we use DRL to allow the controller to continuously track



(a) An example of fiber with varying diameter

(b) An example of fabric weaved with the varying diameter fiber

Figure 1-3: Possible application of a fiber with varying diameter

a varying reference diameter without requiring a CFD model for the setpoint changes on.

We expect that the capability of producing fiber with varying diameter will enable us to embed various physical or optical properties to the fiber. Furthermore, the fabric made out of such fiber may have special properties as well, which we can exploit for a certain purpose. For example, if the fibers in Figure 1-3a are weaved into the fabric, it will form an interlocking structure, as shown in Figure 1-3b. This fabric is expected to be resistive to slipping between fibers while having a relatively low bending rigidity. Thus, if we have a control system capable of supporting varying geometries, we can customize the properties of the produced fiber or fabric.

1.3 Deep Reinforcment Learning Approaches for Control

The availability of computation power in recent years has triggered a strong interest in control methods that utilize machine learning, especially model-free DRL. Model-free DRL based algorithms have outperformed the humans in playing the game of Go [19] and Atari [20]. It is successful in the simulation of physical tasks [21], heat exchanging process control [22], and AUV control [23]. It also performs well in real-

world applications such as robot manipulation [24], high precision assembly tasks [25] and flying quadrotor [26]. One of the strengths that enabled the success of model-free DRL is that it does not require an accurate model as a prerequisite. Even when the model is inaccurate or unknown, the DRL agent can learn and optimize the control policy by interacting with the system. Without prior knowledge about the system, the DRL agent estimates the best control action by learning from trial and error.

This thesis discusses how the model-free DRL algorithm can be used to optimally and predictively control the fiber drawing process without an analytical or numerical model of the system. We focus on regulating the diameter to the reference trajectories, either steady or varying. We apply the DRL framework to the desktop fiber drawing system, which has been developed in Device Realization Laboratory at MIT [27]. Compared to industrial optical fiber draw towers, which are typically a few stories tall, it is significantly smaller so that it fits on a tabletop. The associated cost to build and operate the system is relatively small.

Chapter 2

Background

2.1 Reinforcement Learning

Reinforcement learning (RL) is a programming method that trains the control agents to maximize rewards or minimize the penalty. As shown in Figure 2-1, the RL agent interacts with the environment at each time step. It receives the observation and reward from the environment, then computes the action based on its policy. The environment is affected by the action and the new reward that corresponds to the new state of the environment is computed. The cycle is repeated until the task is over. In this cycle, the action that resulted in high reward is 'reinforced'. In other words, the agent tends to prefer the action that is similar to the reinforced action. As a result, the agent converges to the optimal policy, which takes the optimal action to maximize the expected future reward. In the manufacturing process, a controller and a manufacturing process can be considered as the agent and the environment; the controller receives the observations through sensor readings and computes the control inputs.

2.1.1 State-Action Value function Q(s,a)

The state-action value function, also called the Q-function, represents the expected future reward when taking a certain action at a certain state, then thereafter following

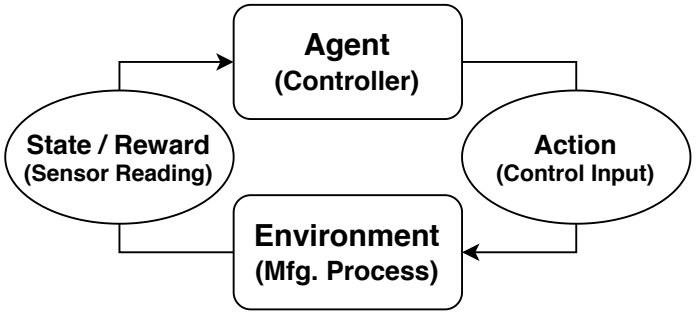


Figure 2-1: Reinforcement Learning Schematics

the agent's policy,

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{\mu}[R_t | S_t = s_t, A_t = a_t], \tag{2.1}$$

where R_t and μ represents the expected future reward and the agent's policy, respectively. The expected future reward R_t, also called return, is often discounted with a discount factor $\gamma \in [0, 1)$,

$$R_{t_0} = r_{t_0} + \gamma r_{t_0+1} + \gamma^2 r_{t_0+2} + ... = \sum_{t=t_0}^{end} \gamma^{t-t_0} r_t, \tag{2.2}$$

where r_t is the reward at time t. The discount factor γ models the notion that a state and an action have decreased relation with the state and reward that are farther separated in time. Therefore, the reward is discounted at each time step by multiplying the discount factor γ. As a result, when optimizing the return, the agent is biased on the more recent time steps.

2.1.2 Bellman Equation and Temporal Difference Method

A large number of computations are typically required to calculate the Q-value with the Monte-Carlo method by trying the entire trajectory multiple times. Therefore, the temporal difference method with the Bellman equation (2.5) is widely used to solve this issue by bootstrapping the Q-value estimation between consecutive time

step,

$$
\begin{aligned}
Q^{\mu}(s_t, a_t) &= \mathbb{E}_{\mu}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... | S_t = s_t, A_t = a_t] && (2.3)\\
&= \mathbb{E}_{\mu}[r_t + \gamma(r_{t+1} + \gamma r_{t+2} + ...) | S_t = s_t, A_t = a_t] && (2.4)\\
&= \mathbb{E}_{\mu}[r_t + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1})) | S_t = s_t, A_t = a_t] && (2.5)
\end{aligned}
$$

This Bellman equation compares the Q-values with a single time step difference. The error between the left and right side of the equation is called the temporal difference (TD) error. By iterating the Bellman equation, the Q-value can be estimated with significantly less computation. The simplest temporal difference update can be expressed as below (2.6):

$$
Q^{\mu}(s_t, a_t) \leftarrow Q^{\mu}(s_t, a_t) + \alpha[r_t + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1})) - Q^{\mu}(s_t, a_t)] \qquad (2.6)
$$

where α is the step-size of the update.

2.1.3 Actor-Critic Approach

The actor-critic approach is often used for reinforcement learning [28]. The approach approximates both policy and value function with the actor and the critic. An actor acts as the agent observing a state from the environment and computing actions accordingly. The critic evaluates the actor's action by estimating value function. A critic takes the observations and the actions as the inputs and computes the value estimation. Based on the critic's evaluation, the actor is updated along the direction that increases the value estimation. Simultaneously, the critic is also updated by minimizing the TD error in the Bellman equation. Consequently, the critic converges near the true value function and the actor is optimized to maximize the value function. In deep reinforcement learning, multilayer perceptrons are often used as function approximators for the actor and the critic (e.g. deep deterministic policy gradient (DDPG) [21]). Recurrent Neural Networks (RNN) are used to consider the history of the observations and actions (e.g. recurrent deterministic policy gradient (RDPG)

[29]). There are some cases where multiple critics are used (e.g. twin delayed DDPG (TD3) [30]).

Deep Deterministic Policy Gradient (DDPG)

Algorithm 1 Deep Deterministic Policy Gradient

Initialize critic networks Q_θ, and actor network π_ϕ with random parameters θ, ϕ
Initialize target networks $\theta' \leftarrow \theta, \phi' \leftarrow \phi$
Initialize empty replay buffer R
for episode = 1 : M **do**
  Sample the initial state s_0
  for timestep = 1 : T **do**
    Observe state s_t
    execute action $a_t = \pi_\phi(s_t) + \epsilon$, ϵ: exploration noise
    Observe state s_{t+1} and reward r_t
    store transition (s_t, a_t, r_t, s_{t+1}) to the replay buffer R
    Sample a mini-batch of N transitions from R: $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$
    Compute target values for each transition:
    $\widetilde{a}_{t+1}^i \leftarrow \pi_{\phi'}(s_{t+1}^i)$
    $y^i \leftarrow r_t^i + \gamma Q_{\theta'}(s_{t+1}^i, \widetilde{a}_{t+1}^i)$
    Compute critic update:
    $\Delta\theta = \frac{1}{N}\sum_i (y^i - Q_\theta(s_t^i, a_t^i)) \frac{\partial Q_\theta(s_t^i, a_t^i)}{\partial \theta}$
    Compute actor update:
    $\Delta\phi = \frac{1}{N}\sum_i \frac{\partial Q_\theta(s_t^i, \pi_\phi(s_t^i))}{\partial a} \frac{\partial \pi_\phi(s_t^i)}{\partial \phi}$
    Update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i', \phi' \leftarrow \tau\phi + (1-\tau)\phi'$
  end for
end for

DDPG is one of the most widely used actor-critic methods. It approximates the actor and the critic with multilayer perceptrons. While the actor interacts with the environment by exerting actions and receiving observations, the critic evaluates the actor's action with the Q-value estimation. Target actor and the target critic network are also used to facilitate the stable convergence. The pseudo-code of this algorithm is outlined in Algorithm 1.

First, the actor and the critic networks are initialized by randomly assigning the values of each parameter. The target actor and the target critic are initialized by

copying parameter values from the actor and the critic. The empty replay buffer is also initialized. A replay buffer is a storage unit that stores previous actions and observations. After the initialization, the agent starts to interact with the environment. At each timestep of each episode, the actor computes the "greedy" action. Then the exploration noise is added to the greedy action before it is exerted to the environment. The action with the exploration noise is executed and then the actor observes a new state and a reward. These states, actions, and rewards are stored in the replay buffer for the later use in the training.

The training starts after a reasonable amount of states, actions, and rewards data are collected in the replay buffer. First, a mini-batch of transition data is sampled from the replay buffer. Then, the target value of each transition is computed from the target networks and compared with Q-value estimation of the critic to compute the TD error. The mean squared error of TD error is used as the loss and the critic is updated by minimizing the loss. The actor is updated along the direction of Q-value estimation increment. Lastly, the target networks are updated by interpolating between old parameters of the target actor and the target critic networks and new parameters of the actor and the critic networks.

2.1.4 Partial Observability

Partially Observed Markov Decision Process (POMDP)

Full observability means that the observation can represent the current state completely. Therefore, if the state is fully observed, then the probability distribution of the next state only depends on the current observation and the current action. This types of model is called a Markov decision process (MDP). On the other hand, partial observability means that the observation can represent only partial components of the full state. Therefore, the probability distribution of the next state cannot be determined based on the current observation and the current action. This model is a partially observed Markov decision process (POMDP).

In the fiber drawing process discussed in Chapter 3, sensors measure the diameter

of the fiber, temperature of the heating chamber, speed of the spool motor, and feed rate of the extruder. However, this is not a full observation mainly due to delayed dynamics. The diameter response to the feed rate or the spool speed change is delayed by the time it takes for material to flow through the system. Therefore, the history of observations and actions are needed to predict the future states accurately. One solution to this problem is using RNNs. RNNs pass activation values to consecutive time step so the inputs at the previous time steps are considered when computing outputs. On the other hand, a non-recurrent network computes its outputs from scratch at each time step so only the current input is considered.

Long Short Term Memory (LSTM)

A neural network is generally updated using knowledge of the gradient. In a typical feed-forward network, the gradient is backpropagated from the output layer to the input layer. In RNNs, the gradient is also backpropagated through time (BPTT) to consider the previous inputs while updating. LSTM is a type of RNN that enables BPTT to reach farther time steps without vanishing gradient by using gate mechanism [31]. Therefore, LSTM is widely used in domains such as robot manipulation [32], self-driving cars [33] and language modeling [34].

Recurrent Deterministic Policy Gradient (RDPG)

RDPG is a modified version of DDPG. While DDPG uses non-recurrent multilayer perceptrons to represent the actor and the critic, the RDPG uses RNN to represent the networks. By using RNN, it can consider the history of interaction between the agent and the environment. Therefore, RDPG generally works better than DDPG in controlling POMDP, where the state of the system is not fully observed. Algorithm 2 summarizes the pseudocode of the RDPG. It is similar to DDPG except that it uses RNN and BPTT to update the networks.

Algorithm 2 Recurrent Deterministic Policy Gradient

Initialize critic networks Q_θ, and actor network π_ϕ with random parameters θ, ϕ
Initialize target networks $\theta' \leftarrow \theta, \phi' \leftarrow \phi$
Initialize empty replay buffer R
for episode = 1 : M **do**
  Initialize the environment
  Initialize empty history h_0
  for timestep = 1 : T **do**
    Observe observation o_t
    $h_t \leftarrow h_{t-1}, a_{t-1}, o_t$, append observation and previous action to the history
    execute action $a_t = \pi_\phi(h_t) + \epsilon$, ϵ: exploration noise
  end for
  Store the full history $(o_1, a_1, r_1, ..., o_T, a_T, r_T)$ in replay buffer R
  Sample a mini-batch of N episodes from R: $(o_1^i, a_1^i, r_1^i, ..., o_T^i, a_T^i, r_T^i)$
  Construct history slices: $h_t^i = (o_1^i, a_1^i, ..., o_t^i)$
  Compute target values for each transition:
  $\widetilde{a}_{t+1}^i \leftarrow \pi_{\phi'}(h_{t+1}^i)$
  $y^i \leftarrow r_t^i + \gamma Q_{\theta'}(h_{t+1}^i, \widetilde{a}_{t+1}^i)$
  Compute critic update (BPTT):
  $\Delta\theta = \frac{1}{NT}\sum_i (y^i - Q_\theta(h_t^i, a_t^i)) \frac{\partial Q_\theta(h_t^i, a_t^i)}{\partial \theta}$
  Compute actor update (BPTT):
  $\Delta\phi = \frac{1}{NT}\sum_i \frac{\partial Q_\theta(t_t^i, \pi_\phi(h_t^i))}{\partial a} \frac{\partial \pi_\phi(h_t^i)}{\partial \phi}$
  Update target networks:
  $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i', \phi' \leftarrow \tau\phi + (1-\tau)\phi'$
end for

Chapter 3

Mechanical System Design

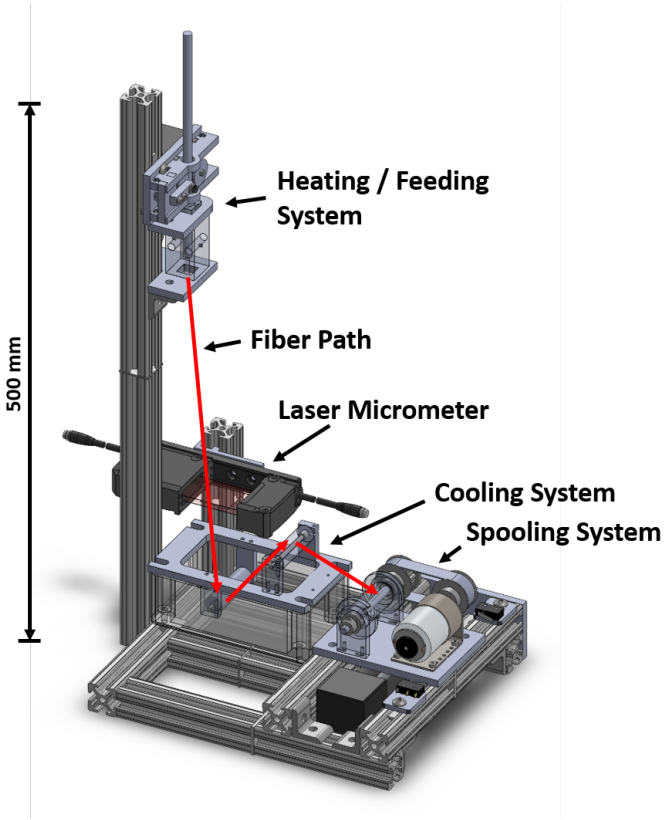
3.1 Design Overview

Figure 3-1a shows a CAD overview of the desktop fiber manufacturing system [27]. The DRL-based control system was developed, trained, and evaluated on this physical system (Figure 3-1b). It miniaturizes a typical industrial fiber draw tower. It is composed of several sub-systems: an extruder system, a cooling system, and a spool system. The preform rod of 7.11 mm diameter is first fed into the extruder system, where it is heated over its glass-transition temperature and becomes deformable. As it is heated, it is pulled to the downstream as a thin fiber by going through the neck-down profile. The fiber then enters the cooling system and is immersed in coolant. Lastly, the fiber exits the coolant and is wrapped around the pulley in the spooling system.

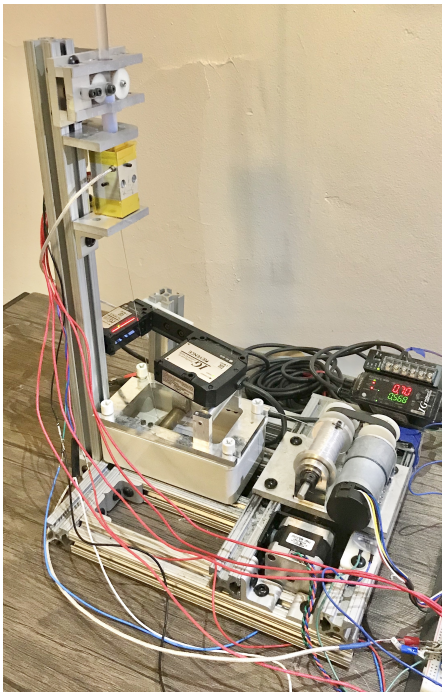
3.2 Heating and Feeding System

The extruder system is shown in Figure 3-1c. The extruder system is composed of a heating chamber and a feeding actuator. The heating chamber has a sensor and heating elements to control the temperature. The feeding actuator feeds the preform into the heating chamber at a controlled speed.

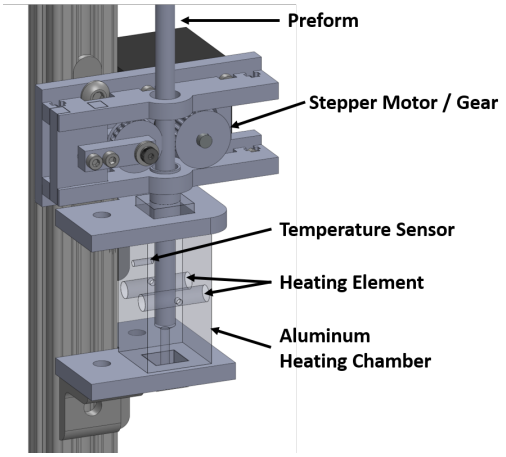
The heating chamber has two cartridge heaters each operating at 40W. It also has



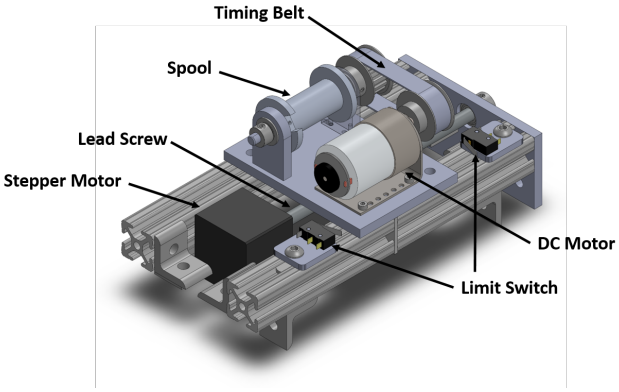
(a) Overview



(b) Physical system



(c) Heating and feeding system



(d) Spooling system

Figure 3-1: Desktop fiber manufacturing system

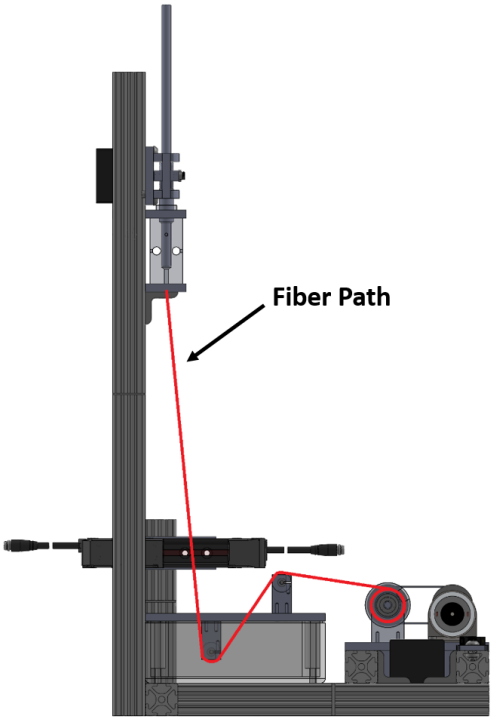


Figure 3-2: Path of the fiber

a resistance temperature detector (RTD) to measure the temperature. In the center of the heating chamber, a hole slightly larger than the preform diameter is machined. The temperature is kept constant throughout the process with a manually tuned PID controller.

The feeding actuator is composed of a stepper motor and an idler. The feed rate is controlled by the stepper motor speed. The stepper motor takes 3,200 steps per one revolution and the diameter of the roller attached to the stepper motor is 18.9 mm. Therefore, the feed speed of the preform rod can be calculated as below,

$$v_{\text{preform}} = \frac{D}{2}\dot{\theta} = \frac{18.9 \text{ mm}}{2} \times \frac{2\pi \text{ rad/rev}}{3{,}200 \text{ step/rev}} \times f \text{ [step/sec]} \quad (3.1)$$

$$= 1.85\text{e-}2 \text{ mm/step} \times f \text{ [step/sec]}, \quad (3.2)$$

where D, $\dot{\theta}$, and f are diameter of the roller, angular velocity of the motor, and frequency of the step. As the feed rate increases, the fiber diameter increases given a

fixed spool velocity.

3.3 Cooling and Spooling System

After the fiber comes out of the extruder system, the fiber goes through the cooling and spooling system. The overall path of the fiber is shown in Figure 3-1a with a red arrow. First, the fiber will go through the laser micrometer for diameter measurement before it enters the coolant. After the fiber comes out of the cooling system, the fiber enters the spool system.

The detailed design of the spool system is shown in Figure 3-1d. The main function of the spool system is to collect the fiber and to provide speed feedback to control the fiber diameter. The spool is rotated by a DC motor with encoder attachment. The spool and the DC motor is mounted to the stage that is actuated by a lead screw and a stepper motor. The stage movement along the lead screw allows the fiber to be spread out evenly on the spool. The limit switches limit the range of the linear motion of the stage to ensure the fiber does not go off the spool's ends. The diameter of the spool is 20 mm. As the spool spins faster, the fiber goes under tension and diameter reduces given a fixed feed rate.

Chapter 4

Learning Algorithm

4.1 Overview

The learning algorithm inspired by DDPG [21], RDPG [29], and other variations [32, 35] is used for developing and training the controller. Figure 4-1 shows the overview of the learning method and the pseudocode is elaborated in Algorithm 3. Four LSTM networks compose the overall model: actor, critic, target actor and target critic. The networks are manipulated in the three sub-processes: initialization, control thread, and train thread. The control thread and the train thread run simultaneously throughout the entire process.

In the control thread, the sensors attached to the system measure the state and the actor computes the action accordingly. The reward is computed using the reward function. These observations, actions and rewards are then stored in the history memory **H**. In the train thread, a mini-batch of data sampled from the history memory is fed into a critic and the critic computes the Q-value as an output. The Q-value is then compared with the target value computed by target networks and the critic is updated by minimizing the difference between the Q-value and the target value. Finally, the actor is updated by maximizing the critic's evaluation (Q-value).

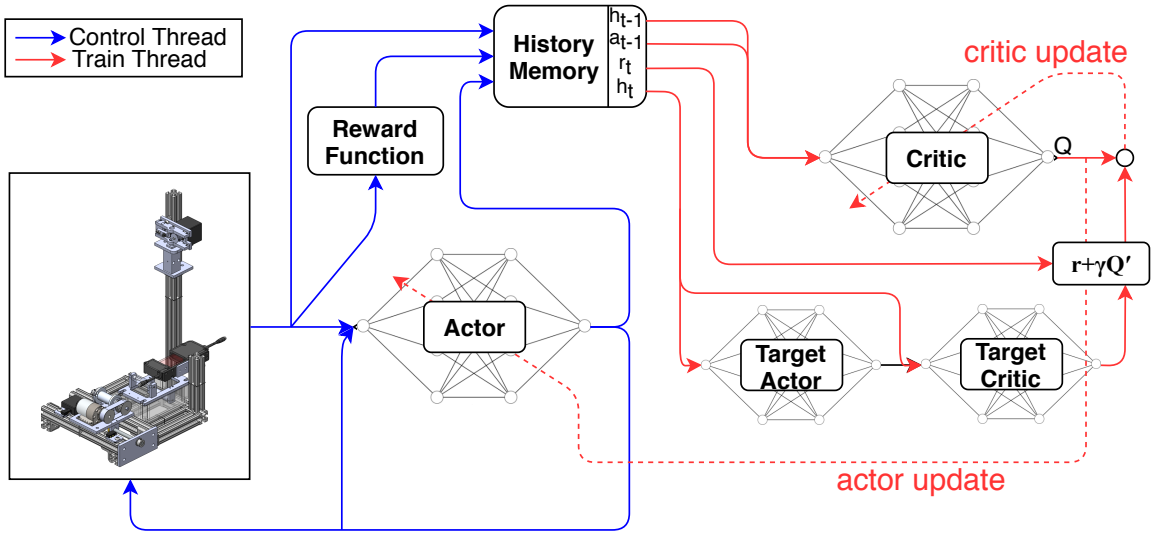


Figure 4-1: Learning algorithm overview

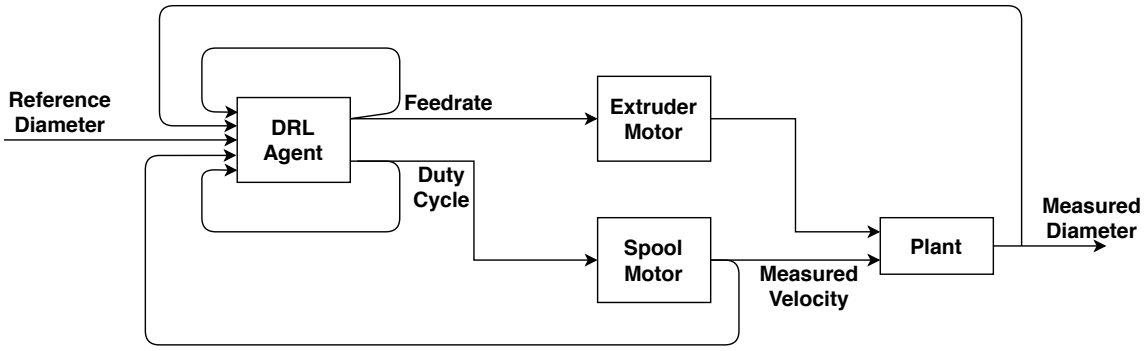


Figure 4-2: Block diagram of the DRL control system

4.2 POMDP Formulation

The system should be formulated as POMDP or MDP to implement the algorithm. As mentioned in Section 2.1.4, sensor measurements of the fiber drawing system only observes the partial component of the system, mainly due to the delayed dynamics. Therefore, we formulate the process as POMDP by defining observation, action, and reward.

4.2.1 Observation

Observations are the numerical values that help the actor and the critic to estimate the optimal action and the true Q-value. It includes physical measurements from sensors or other numerical values that are necessary for the agent to learn the optimal policy. Our formulation of observation includes the below components:

The Spool Angular Velocity is measured by a encoder attached to the spool motor. The motor has a gearbox ratio of 131.25 : 1 and the encoder has a resolution of 64 counts per revolution of the motor input shaft, which corresponds to 8,400 counts per revolution of the output shaft. The angular velocity is calculated by dividing number of counts by time interval of the sampling. The sampling rate is set to 4 Hz.

The Fiber Diameter is measured by a laser micrometer, which is mounted between the extruder and the cooling bath. After the fiber come out from the extruder and becomes thinner by going through the neck-down profile, the micrometer measures its diameter right before the fiber enters the coolant. The sensor is capable of measuring minimum diameter of 200 μm and maximum diameter of 28 mm. The repeatability of measurement is 5 μm. The linearity is $\pm 0.1\%$ of full scale (28 μm).

The Cumulative Sum of Extruder Feedrate represents how much fiber has produced during the current production run. This is important information because the system is not time-invariant due to the accumulation of drawn fiber on the spool. As the fiber is drawn and wrapped around the spool, the effective radius of the spool increases with respect to time. Therefore, the linear velocity of the fiber increases when the spool's angular velocity is a constant value. Consequently, if the stacking fiber on the spool is not considered and the spool is run with a constant angular velocity, the linear velocity increases and results in a thinner fiber. We assume that there is a very strong relationship between the cumulative summation value and the effective radius of the spool, and therefore include the summation value in the

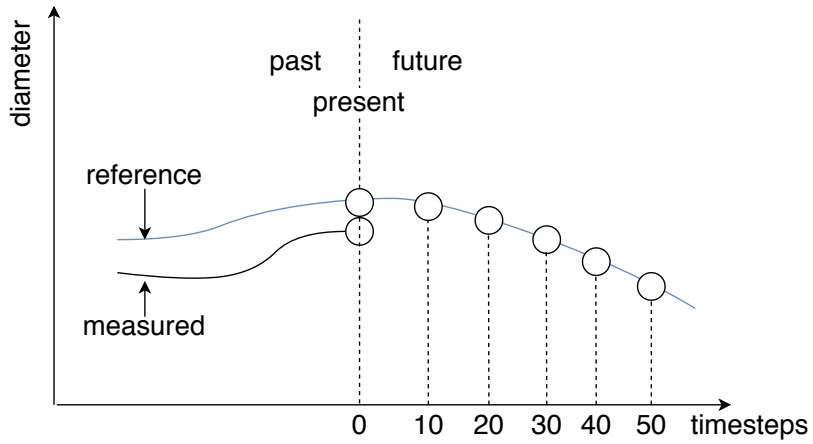


Figure 4-3: An example of a reference and a measured diameter trajectories. Not only current measured diameter and reference diameter, but also future reference diameter of 10, 20, 30, 40, 50 time steps ahead are included in the state representation

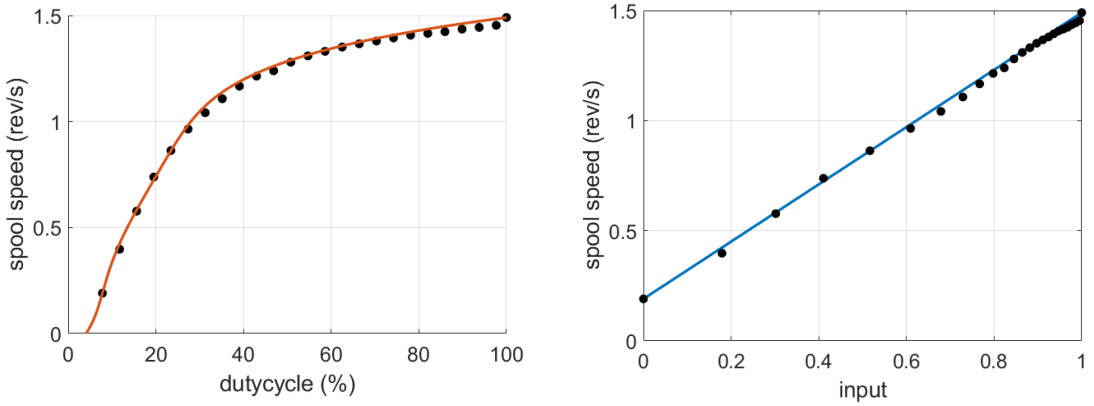
observation.

The Target Diameter is the diameter of the desired reference trajectory of diameter. The target diameter at the present time step and of several future target diameters are included (10, 20, 30, 40, 50 time steps ahead). This is because the agent requires the future reference in order to predictively regulate the output to track the reference. In other words, the observation looks as far as 50 time steps (12.5 seconds) ahead. Therefore, the actor and the critic can estimate the optimal action and the Q-value precisely unless the dynamic response time to control input longer than 12.5 seconds. In the desktop fiber drawing system, majority of the delayed dynamics responses happen in less than 10 seconds.

4.2.2 Action

Actions are the control inputs that are computed by the actor network and executed on the environment. In the desktop fiber drawing system, the actions include the command spool input and the command extruder input.

The Spool Input Command determines the PWM duty cycle input to the spool motor. The duty cycle is constrained to have a minimum value of 7.8% and a maxi-



(a) Motor's duty cycle vs. speed relation before linear mapping

(b) Action input vs. speed relation after linearly mapping action input to the speed

Figure 4-4: Action-speed linear mapping

mum value of 100%. The spool input value is normalized that the value is 0 when the duty cycle is minimum and 1 when the duty cycle is maximum. The relation between the spool motor's duty cycle and the spool speed is shown in Figure 4-4a. The slope is steeper at the lower duty cycle and more flat at the higher duty cycle. In other words, at the low velocity, the velocity is very sensitive to the variation of the duty cycle. Consequently, if the spool input is mapped just linearly with the duty cycle, then it will be hard to control the velocity precisely. Therefore, we do polynomial regression on Figure 4-4a and convert the spool input so that it has a linear relation with the velocity, as shown in Figure 4-4b. This way, by reducing a degree of nonlinearity, it enables more precise control at a low velocity.

The Extruder Input Command determines the frequency of the stepper motor that pushes the preform into the heating element. The stepper motor frequency is constrained to have a minimum value of 5 Hz and a maximum value of 30 Hz, each corresponds to 9.28e-2 mm/s (5.57 mm/min) and 5.57e-1 mm/s (33.4 mm/min). The extruder input is normalized in the same way as the spool input so that the value is 0 and 1 when the frequency is minimum and maximum. Contrast to the spool input, since the preform feed-rate is proportional to the frequency, the extruder input is linearly mapped with the stepper motor's frequency.

4.2.3 Reward Function Design

A careful design of the reward function is required in order to ensure the performance of the algorithm. The reward function is designed as below (4.1).

$$r_t = -|d_t - \zeta_t| + \alpha f_t + C, \tag{4.1}$$

where α and C are positive scalars, and d and ζ are the measured and the reference diameter in 100 μm. The consection terms of the reward function represents the diameter error, extruder feed rate, and the offset.

Diameter Error

The first term represents the difference between the reference diameter and the measured diameter at each step. Since the control objective is to regulate the diameter to the reference diameter as close as possible, the reward decreases as the difference increases.

Extruder Feedrate

The second term is proportional to the feed rate of the material and thus represents the mass production rate of the fiber. The α is set to 0.106 s/mm, which makes this term to have an order of approximately ten times smaller than the first term. This term is needed to ensure the uniqueness of the input action combination. There are two input actions (the spool and the extruder input), which regulate only one output measurement (diameter). Therefore, there could be several input action combinations that yield a similar diameter. For example, a combination of a high spool input and a high extruder input can lead to a similar diameter as when a low spool input and a low extruder input is used. However, by adding the second term, the model chooses the combination that maximizes the production rate when there are several other options with similar diameter output.

Offset

The offset term is used to facilitate the learning. If there is no offset term, the reward will be negative at most times. This will lead the model to think that the actions in the operable boundary are worse than the actions that are outside of the operable region, especially at the early stage of the learning. In this case, the action can be trapped near the operable boundary. The offset term is set to 1 so that it has the similar scale with other terms.

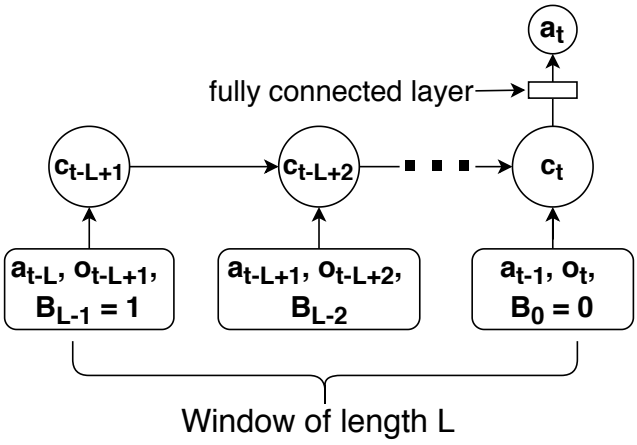
4.3 Network Architecture

4.3.1 Actor Network

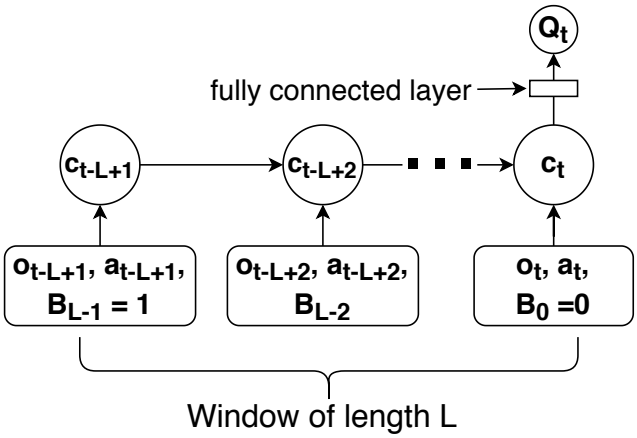
The actor network structure is shown in Figure 4-5a. The structure of the target actor is identical to that of the actor. Each of the circles in the figure represents the LSTM network. The number of layers of the LSTM networks is set to 5 and the number of nodes in each layer is set to 512. The networks recur through L time steps. L is the window length and the window is a span of time where the networks take inputs. The outputs of the networks are determined by the inputs that are within the window length. At each time step, the action taken at one time step before and the following observation is fed into the actor. The action output is computed by passing the activation values of the last recursion through a fully connected layer.

4.3.2 Critic Network

The critic network structure is shown in Figure 4-5b. The target critic has the same structure as the critic network. The critic network takes similar form as the actor network but the input and output elements are different. The number of layers and nodes of the LSTM network are set to 5 and 512. At each time step, an observation at each time step and the following action is fed into the critic and the Q-value output is computed by passing the activation values of the last recursion through a fully connected layer.



(a) Actor network



(b) Critic network

Figure 4-5: Network architecture

4.3.3 Window Length (L)

In the original RDPG paper [29], which our model is inspired by, the activation values of the LSTM network are propagated from the beginning to the end of each episode. The gradients are back-propagated to the beginning of the episode, and the updates are done between each episode rather than within the episodes. One problem with this method is that computation time increases as the episode gets longer since the gradient must back-propagate through the entire episode. Therefore, the computational requirement can become a bottleneck if we want to train the model in real-time. In the case of the fiber drawing system, each episode is thousands of time steps long (tens of minutes). Therefore, the computation time becomes long for

each training iteration, which makes it hard to train the model in real-time.

Thus, we consider only the time span that significantly affects the state of the system, rather than the entire episode. We set the length of the window, through which the networks look into the system (Figure 4-5). The networks do the computations for control and updates only within the window. The window size should be long enough to capture the delayed dynamics of the system. One of the longest delayed dynamics in the fiber drawing system is the delay between the feed rate change and the response in diameter. When we apply a step change to the feed rate, it takes about ∼10 seconds (∼40 time steps for given sample rate) for the response to show up in the diameter. Therefore, the window length should be at least ∼40 time steps to capture the delayed dynamics.

In the original RDPG, the model is trained by minimizing the Bellman equation's temporal difference (TD) error based on Q-value computed at every time step of the episode. However, as we set a finite window length, outputs at the early part of the window will be less accurate than that of the later part. This is because the result is computed based on less information. For example, the output at the first time step of the window is based on only one observation and one action. Therefore, we only consider the last Q-value computed within the window, as depicted in Figure 4-5b.

4.3.4 When-Label (B)

To facilitate the learning, when-labels are augmented to the inputs. When-labels have scalar values between 0 and 100 so that it has a similar scale with other inputs. It indicates how far ago from the present did each observation and action happened. It forms an arithmetic sequence, where the most recent inputs have a when-label value of 0 and the oldest inputs within the window size have a value of 100 (Figure 4-5). Without when-label, the LSTM network processes the inputs in the same way, no matter when the input data is produced. In contrast, the network can process the inputs more efficiently if it knows when the input data arrived.

4.4 Initialization

The first step of the algorithm is initializing each network. The parameters of the actor and the critic are initialized using the Glorot initialization [36]. Then, the parameters of the actor and the critic are copied to the target actor and the target critic. Next, the empty history memory $\mathbf{H}$ is initialized. Lastly, the history buffer h of window length L is initialized. The history buffer is a buffer that contains the L most recent observations and actions.

4.5 Control Thread

The control thread is where the actor receives the observation from the system and computes the input action. Exploration noise is added to the actor's output action (greedy action) in order to find a better action in the action space.

4.5.1 Data generation and Storage

First, the actor receives the observation and the reward is computed by a reward function in (4.1). Next, the observation and the most recent action are appended to the history buffer h. The actor then takes the history buffer as the input and computes a greedy action. An exploration noise is added to the greedy action to explore the action space. Lastly, the action with the exploration noise is exerted on the fiber drawing system as the control input. The reward, observation, and action are added to the history memory $\mathbf{H}$ at each time step.

4.5.2 Exploration Strategy

An Ornstein-Uhlenbeck (OU) process [37] with a decay factor β is used for the exploration noise. It can be expressed as below.

$$dx_t = -\theta\, x_t\, dt + \sigma_t\, dW_t \tag{4.2}$$

$$\sigma_t = \beta\, \sigma_{t-1} \tag{4.3}$$

where θ is a parameter that determines the degree of attraction that pulls variable to zero, W express the Wiener process, and σ is a volatility of the process. The volatility is decreased by the factor of β at each time step. Since the exploration noise is attracted to zero, the noise has a mean value of zero. The process is temporally correlated so it is effective when learning on physical systems, which is also temporally correlated.

4.6 Train Thread

The train thread run in parallel with the control thread. It is composed of three steps: batch sampling from history memory (**H**), temporal difference method implementation, and soft target network updates.

4.6.1 Batch Sampling from History Memory (H)

First, N samples of memory slice, which have a length of $L+2$, are sampled from the history memory **H**. Each sample does not include the beginning or the end of the episode to ensure that the memory slices have the same length. The random sampling decouples temporal correlation between samples and makes the algorithm to avoid bias. In contrast, if only recent samples are used, then the algorithm will be biased to only recent history.

4.6.2 Temporal Difference Method Implementation

Next, using the target networks, we compute the target value y^i as described in Algorithm 3. The target value y^i is used as the right hand side term of (2.5). Then, the mean square value of the TD error (MSE) becomes,

$$\text{MSE} = \frac{1}{N}\sum_i (y^i - Q_\theta(\widetilde{h}^i_{t-1}, a^i_{t-1}))^2. \tag{4.4}$$

Therefore, the critic gradient that decreases MSE can be computed with BPTT,

$$\Delta\theta = \frac{1}{N}\sum_{i}(y^i - Q_\theta(\widetilde{h}^i_{t-1}, a^i_{t-1}))\frac{\partial Q_\theta(\widetilde{h}^i_{t-1}, a^i_{t-1})}{\partial\theta}. \tag{4.5}$$

By applying this gradient, the critic is updated. The Adam optimizer [38] is used as the gradient descent optimizer. After updating the critic, the actor can also be updated by applying gradient that increases the Q-value. The chain rule is used to compute the gradient,

$$\Delta\phi = \frac{1}{N}\sum_{i}\mathcal{C}\left(\frac{\partial Q_\theta(\widetilde{h}^i_{t-1}, \pi_\phi(h^i_{t-1}))}{\partial a}\right)\frac{\partial\pi_\phi(h^i_{t-1})}{\partial\phi}, \tag{4.6}$$

where $\mathcal{C}(\cdot)$ is a gradient transformation inspired by [35], which bounds actions between the maximum and the minimum.

Gradient Transformation

The gradient transformation $\mathcal{C}(\cdot)$ is:

$$\mathcal{C}(\nabla_a) = \begin{cases} \nabla_a \cdot (a_{max} - a)/(a_{max} - a_{min}), \\ \qquad \text{if } \nabla_a \text{ suggests increasing } a \text{ and } a > a_{max} \\ \nabla_a \cdot (a - a_{min})/(a_{max} - a_{min}) \\ \qquad \text{if } \nabla_a \text{ suggests decreasing } a \text{ and } a < a_{min} \\ \nabla_a, \quad \text{otherwise} \end{cases}, \tag{4.7}$$

where a_{max} and a_{min} are the maximum and the minimum of the operable action. If the gradient ∇_a suggests increasing action and the action computed by actor exceeded the maximum action, the transform reverses the direction of gradient so that the action can revert to the operable action space. Thus, the action can be bounded within the operable action space.

4.6.3 Soft Target Network Updates

Lastly, the target actor and the target critic is updated by applying the soft update,

$$(\theta_i', \phi') \leftarrow (\tau\theta_i + (1-\tau)\theta_i', \tau\phi + (1-\tau)\phi'), \tag{4.8}$$

where τ is a very small positive scalar value. This soft updates of the target networks enable the stable convergence of the model [21].

Algorithm 3 Learning Algorithm

Initialize critic networks Q_θ, and actor network π_ϕ with random parameters θ, ϕ
Initialize target networks $\theta' \leftarrow \theta, \phi' \leftarrow \phi$
Initialize empty history memory H
for production run = 1 : M **do**
  Initialize empty history buffer h_0 of length L
  // ***Control Thread***
  while spool is not full **do**
    Observe observation o_t and reward r_t
    $h_t \leftarrow h_{t-1}, a_{t-1}, o_t$, append observation and previous action to the history buffer
    if $t > L$ **then**
      discard the oldest observation o_{t-L} and action a_{t-L} from h_t
    end if
    select action $a_t = \pi_\phi(h_t) + \epsilon$, ϵ: exploration noise (OU process)
    append r_t, o_t, a_t to the history memory H
  end while
  // ***Train Thread***
  while spool is not full **do**
    Sample a mini-batch of N sequences from H:
    $(r^i_{t-L-1}, o^i_{t-L-1}, a^i_{t-L-1}, ..., r^i_t, o^i_t, a^i_t)$
    Construct history buffers:
    $h^i_t = (a^i_{t-L}, o^i_{t-L+1}, ..., a^i_{t-1}, o^i_t)$
    $\widetilde{h}^i_t = (o^i_{t-L+1}, ..., a^i_{t-1}, o^i_t)$
    Compute target values for each sequence:
    $\widetilde{a}^i_t \leftarrow \pi_{\phi'}(h^i_t)$
    $y^i \leftarrow r^i_t + \gamma Q_{\theta'}(\widetilde{h}^i_t, \widetilde{a}^i_t)$
    Compute critic update (using BPTT):
    $$\Delta\theta = \frac{1}{N}\sum_i (y^i - Q_\theta(\widetilde{h}^i_{t-1}, a^i_{t-1}))\frac{\partial Q_\theta(\widetilde{h}^i_{t-1}, a^i_{t-1})}{\partial\theta}$$
    Compute actor update (using BPTT):
    $$\Delta\phi = \frac{1}{N}\sum_i \mathcal{C}\left(\frac{\partial Q_\theta(\widetilde{h}^i_{t-1}, \pi_\phi(h^i_{t-1}))}{\partial a}\right)\frac{\partial \pi_\phi(h^i_{t-1})}{\partial\phi}$$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i, \phi' \leftarrow \tau\phi + (1-\tau)\phi'$
  end while
end for

Chapter 5

Implementation and Baselines

5.1 Hardware Setup

The temperature of the heating chamber was set to 80°C, where the fiber drawing is stable with minimal diameter fluctuation. The temperature was controlled with PI controller. The stage speed was set to 9.28 mm/s. Adtech W220-3824 glue-sticks composed of ethylene-vinyl acetate and room temperature water were used as the material and coolant. For the neural network computation, Nvidia's RTX 2080 was used. Sensor measurements and computation results were received and transmitted to PJRC Teensy 3.5 board, an Arduino-based microcontroller. The Teensy 3.5 then controlled the motors and drivers based on the computation results. The frequency of the control and sensing was set to 4 Hz.

5.2 Hyperparameters

The hyperparameters of the algorithms were set to the values in Table 5.1. The mini-batch size N was chosen to allow the stable computation. If the mini-batch size is too big, it requires too much computation power and may result in unstable computation. Learning rate were carefully tuned to ensure the stable and fast convergence of the parameters. The convergence is slow when the learning rate is small. On the other hand, if the learning rate is to big, the convergence can be unstable and the parameters

Parameter	Value
Minibatch size (N)	32
Actor learning rate	1e-6
Critic learning rate	5e-6
Soft update factor (τ)	0.05
History memory ($\mathbf{H}$) size	75,000
Discount factor (γ)	0.99
OU volatility / speed / decay rate (β)	10 / 0.1 / 0.999925
window length (L)	50

Table 5.1: Hyperparameters for model training

can fluctuate or blow up.

5.3 Training Target Diameter Trajectory Design

To train the model so that it can track the arbitrary step change, a training target trajectory that includes random step changes was used for training (Figure 5-1). The interval of each step is 120 time steps (30 seconds), and each step's diameter is randomly sampled between 300 μm and 600 μm.

5.4 Baseline Control Methods

Several baseline control methods were used to compare the performance of the DRL algorithm with baseline methods. Open-loop control based on mass conservation model and a classical PI feedback control were used as baselines.

5.4.1 Open-loop Control

Unlike closed-loop feedback control, an open-loop control computes spool's angular velocity input from a model and does not feedback measured diameter to compensate error. Only the angular velocity is controlled by integral control. We used mass conservation model to compute the inputs for the open-loop control.

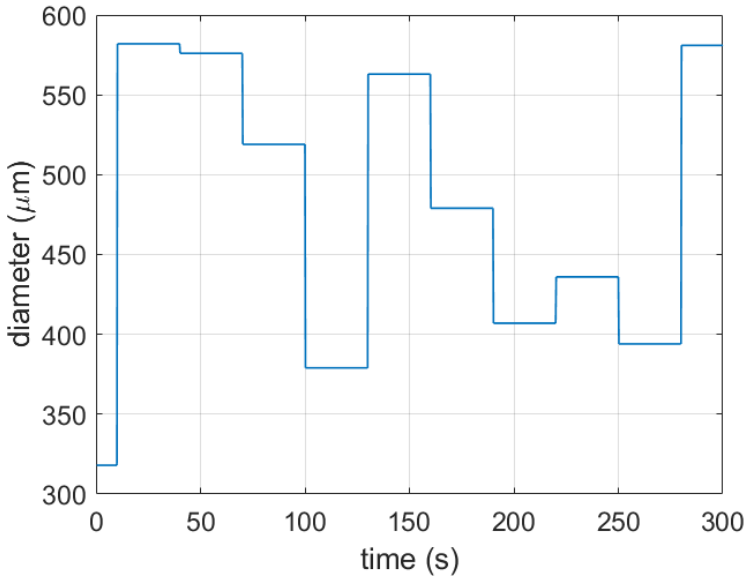


Figure 5-1: An example of training reference trajectory. The reference takes random step every 30 seconds. The maximum and minimum of the random reference is 600 μm and 300 μm.

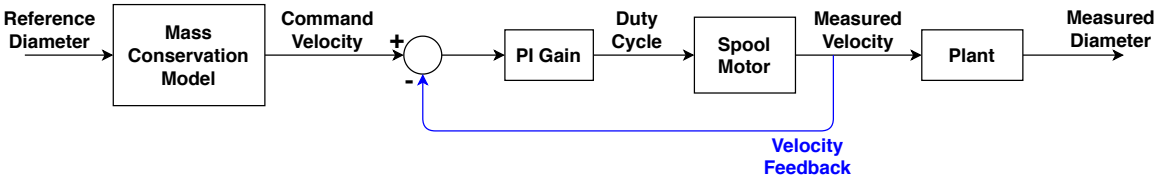


Figure 5-2: Open-loop control with mass conservation model

Mass Conservation Model

The mass conservation model is a model that is based on the assumption that the mass flow rate of the raw material is the same as the mass flow rate of the drawn fiber. Assuming the constant density, the mass conservation model can be expressed as:

$$v_{\text{preform}} A_{\text{preform}} = v_{\text{fiber}} A_{\text{fiber}} = r_{\text{spool}} \omega_{\text{spool}} A_{\text{fiber}}, \tag{5.1}$$

where v, A, r, ω are linear speed, cross-sectional area, radius and angular speed. This model assumes a constant r_{spool}, which means that it does not consider the increase of the effective radius due to the fiber stacking up on the spool. v_{preform} can be computed

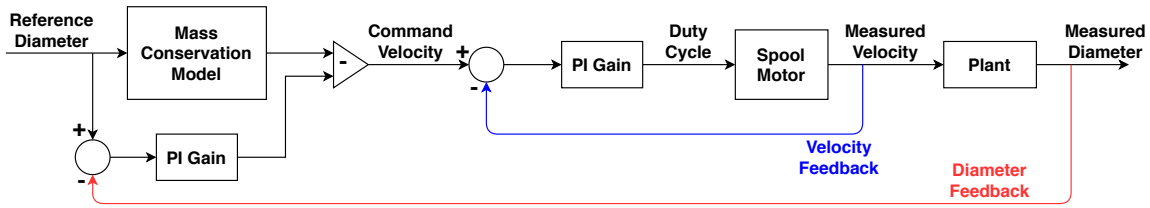


Figure 5-3: PI feedback control with diameter error

with 3.2 so ω_{spool} can be computed as:

$$\omega_{\text{spool}} = \frac{v_{\text{preform}}}{r_{\text{spool}}} \frac{A_{\text{preform}}}{A_{\text{fiber}}} = \frac{v_{\text{preform}}}{10 \text{ mm}} (\frac{6.86 \text{ mm}}{D_{\text{fiber}}})^2, \tag{5.2}$$

where D_{fiber} is the target fiber diameter.

5.4.2 PI Feedback Control

In the PI feedback controller, the error between measured and target diameter is fed back to the controller to compensate the error Figure 5-3. The P, I parameters were manually tuned at the set point diameter of 550 μm. The material feed rate was fixed to 0.37 mm/s and only the spool speed was controlled with P, I gain.

5.4.3 Quadratic Dynamic Matrix Control (QDMC)

QDMC [39, 40] is a type of model-based control that uses the step response model of the system. Under the assumptions that the system is linear and time-invariant, it predicts the future diameter and optimizes the present and future inputs by minimizing the cost function:

$$J = \sum_{i=1}^{p} (d_{t+i}^{\text{ref}} - \hat{d}_{t+i})^2 + r \sum_{i=0}^{c-1} \Delta \mathbf{u}_{t+i}{}^2, \tag{5.3}$$

where $\hat{d}$ is the predicted diameter in 100 μm. $\Delta \mathbf{u}_{t+i}$ is the input change. p and c are the prediction and control horizon. p is set to 50 because our DRL controller also looks 50 timesteps ahead. c is set 25, half of the prediction horizon. r is a weighting factor that defines ratio of importance between output error and input change. The

controller code was developed based on [41].

The model requires the response in diameter to a step change of each input. The square root of the extruder feed rate ($\sqrt{f}$) and reciprocal of the square root of command spool speed ($1/\sqrt{\omega}$) were used as the inputs since the diameter is proportional to $\sqrt{f/\omega}$ according to the mass conservation principle. Each input is normalized that the minimum and maximum are 0 and 1. The diameter response to the step change of $\sqrt{f}$ was measured at spool speeds of 0.6, 1, 1.4 revolution/second, then the average response was used for the step response model. For the diameter response to the step change of $1/\sqrt{\omega}$, the average response at extruder feed rates of 0.19, 0.37, 0.56 mm/s was used for the model.

The weighting factor r also needs to be tuned. If it is too large, the input changes too slow and results in a slow diameter response. If too small, the input responds too sensitive to disturbances or model error and results in fluctuation in diameter. Weighting factors of 5, 10, 20, 40, 80, 160, 320 were tested on the same reference diameter trajectory that was used for the training of the DRL controller. The mean error increased significantly at a weighting factor of 5 and 320. Between 10 and 160, the mean error difference was less than 10%. Therefore, r was set to 40.

Chapter 6

Evaluation

The trained controller was tested on various target diameter trajectories: steady, random step, and continuous. The controller was compared with three other baseline methods. Average error and response delay time were compared. Effect of several implemented modifications were also evaluated: action-speed linear mapping, window length, when-label.

6.1 Test on Various Target Diameter Trajectories

The model was trained for approximately 50,000 time steps (3.5 hours) and tested on several target trajectories: steady, random step, chirp and random spline. Each controller was tested 5 times for each of trajectories and the average responses are shown in the plots. Moving average of 40 timesteps is applied and moving standard deviation ($\times 1.96$) is shown as the shaded areas in Fig. 6-1,6-2,6-3,6-5.

6.1.1 Steady Target

In the steady target trajectory case, the DRL controller was compared with the mass conservation model, PI control, and QDMC (Fig. 6-1).

In the mass conservation model, there was a decreasing trend of diameter with respect to time. Since the model did not consider the increase in the spool radius

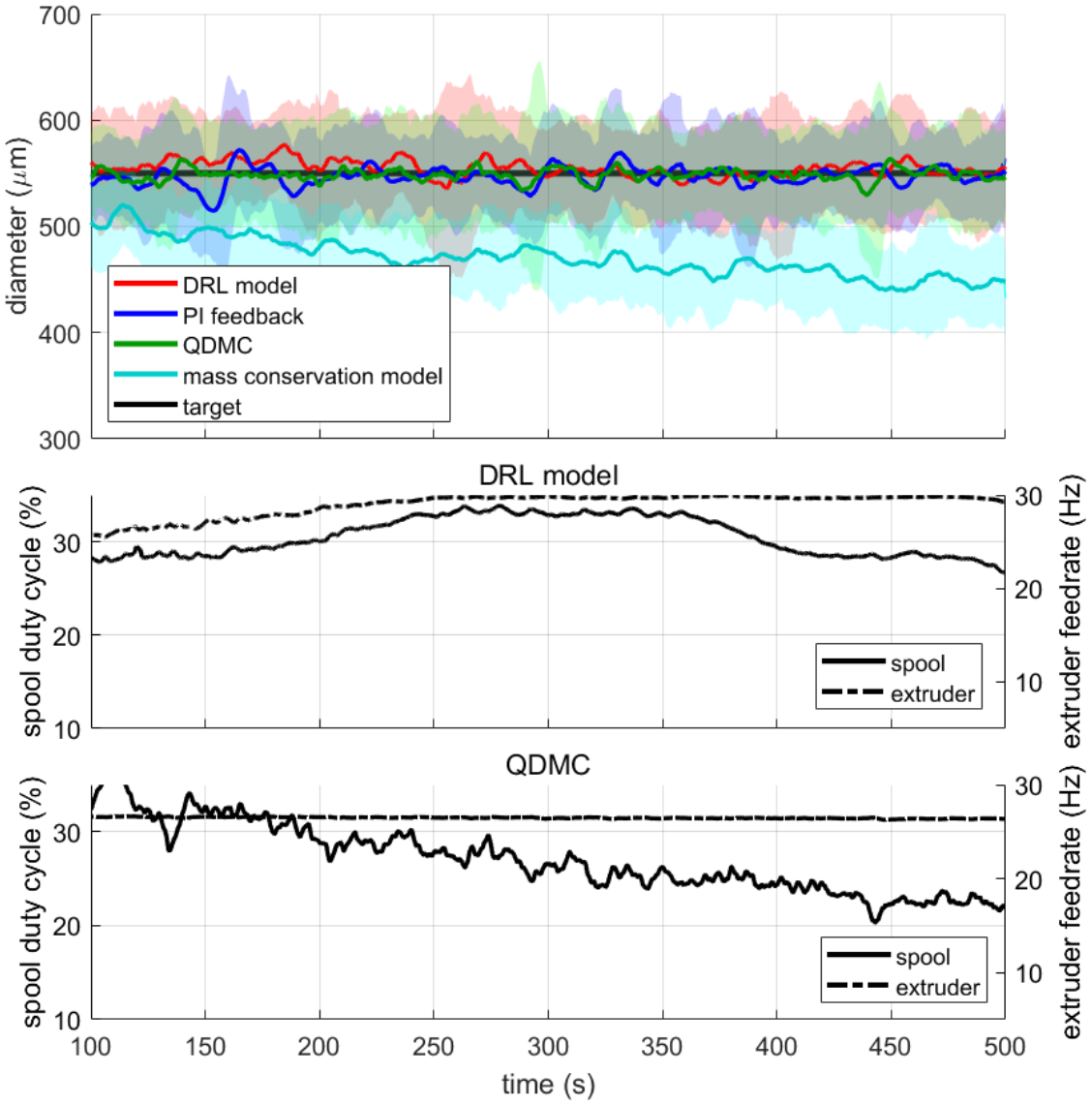


Figure 6-1: (top) Steady reference target diameter at 550 μm and measured diameter trajectory of DRL model, PI feedback control, QDMC, and mass conservation model open-loop control. (middle) The spool's duty cycle and the extruder's feed rate of the DRL model controller. (bottom) The spool's duty cycle and the extruder's feed rate of the QDMC.

and maintained the constant angular speed, the linear speed of the fiber increased and the diameter decreased with respect to time. In comparison, the DRL control, PI control, and QDMC maintained diameter close to the target. In DRL control, it can be seen from the figure that ratio of the extruder input (material feed rate) to the spool input increased with respect to time. It means that it compensated the effect of the stacking spool by feeding more material and rotating the spool slower. Similarly, in PI control, it maintained the constant diameter by decreasing the spool's angular speed with respect to time. As a result, the DRL controller showed an average diameter (551.3 μm) and a standard deviation (29.3 μm) similar to that of the PI control (545.5 / 25.9 μm) and QDMC (548.6 / 28.4 μm)

6.1.2 Random Step Target

The random step target used for testing had an interval of 50 seconds (Fig 6-2). When the PI controller was used for this trajectory, measured diameter response showed 5.7 seconds of average time lag estimated by the cross correlation analysis. It sometimes was not able to settle to the reference diameter within a single interval and sometimes it showed offshoot. Contrastly, the DRL controller and QDMC only showed -0.5 seconds and 0.5 seconds of time lag, respectively. They manipulated input actions in advance to the step changes. For the DRL controller, the spool input changed 4.5 seconds ahead of the steps and the extruder input changed 8.0 seconds in advance to the steps, both estimated by the cross correlation analysis. This is consistent with the intuition that pulling the fiber from the spool induces faster response in diameter change than feeding material from the extruder. This predictive control was possible since we fed into the DRL controller the information about the future trajectory as the observation. The DRL networks perceive the future reference trajectory as far as 50 timesteps (12.5 seconds) away so it can handle the dynamic change that happens within less than 12.5 seconds.

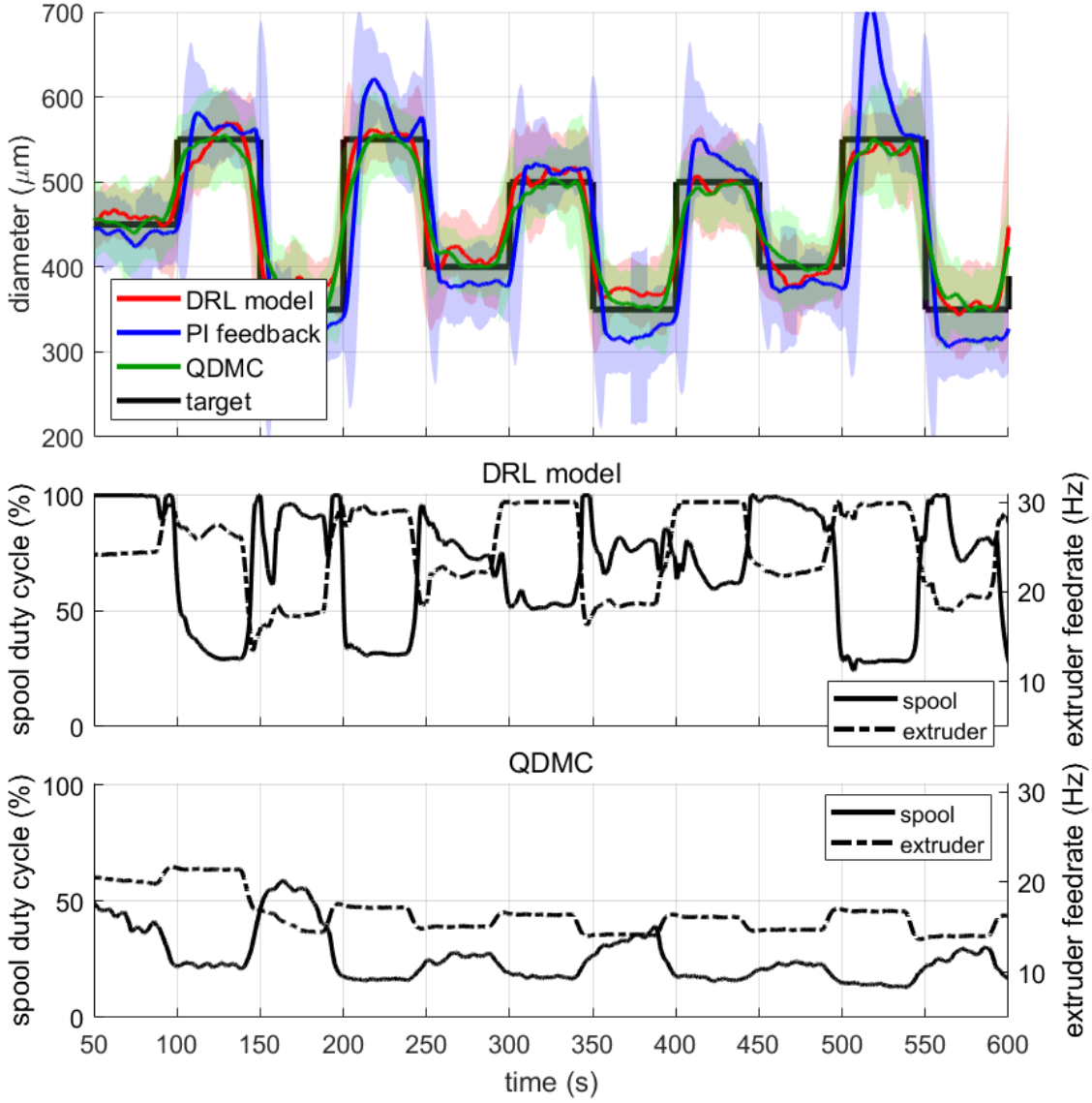


Figure 6-2: (top) Random step reference target diameter and measured diameter trajectory of DRL model, PI feedback control, and QDMC. (middle) The spool's duty cycle and the extruder's feed rate of the DRL model controller. (bottom) The spool's duty cycle and the extruder's feed rate of the QDMC.

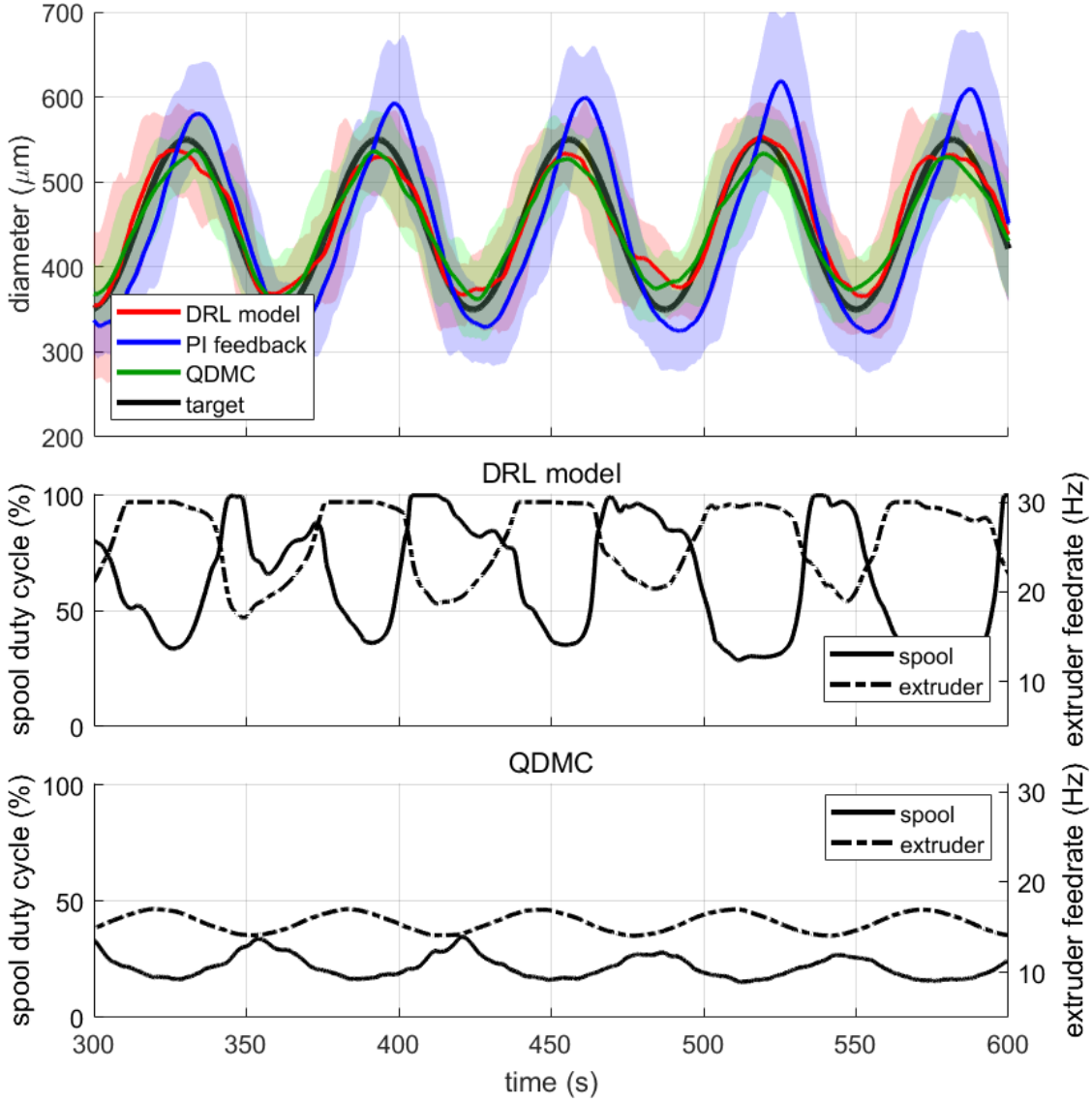


Figure 6-3: (top) Sinusoidal reference target diameter and measured diameter trajectory of DRL model, PI feedback control, and QDMC. (middle) The spool's duty cycle and the extruder's feed rate of the DRL model controller. (bottom) The spool's duty cycle and the extruder's feed rate of the QDMC.

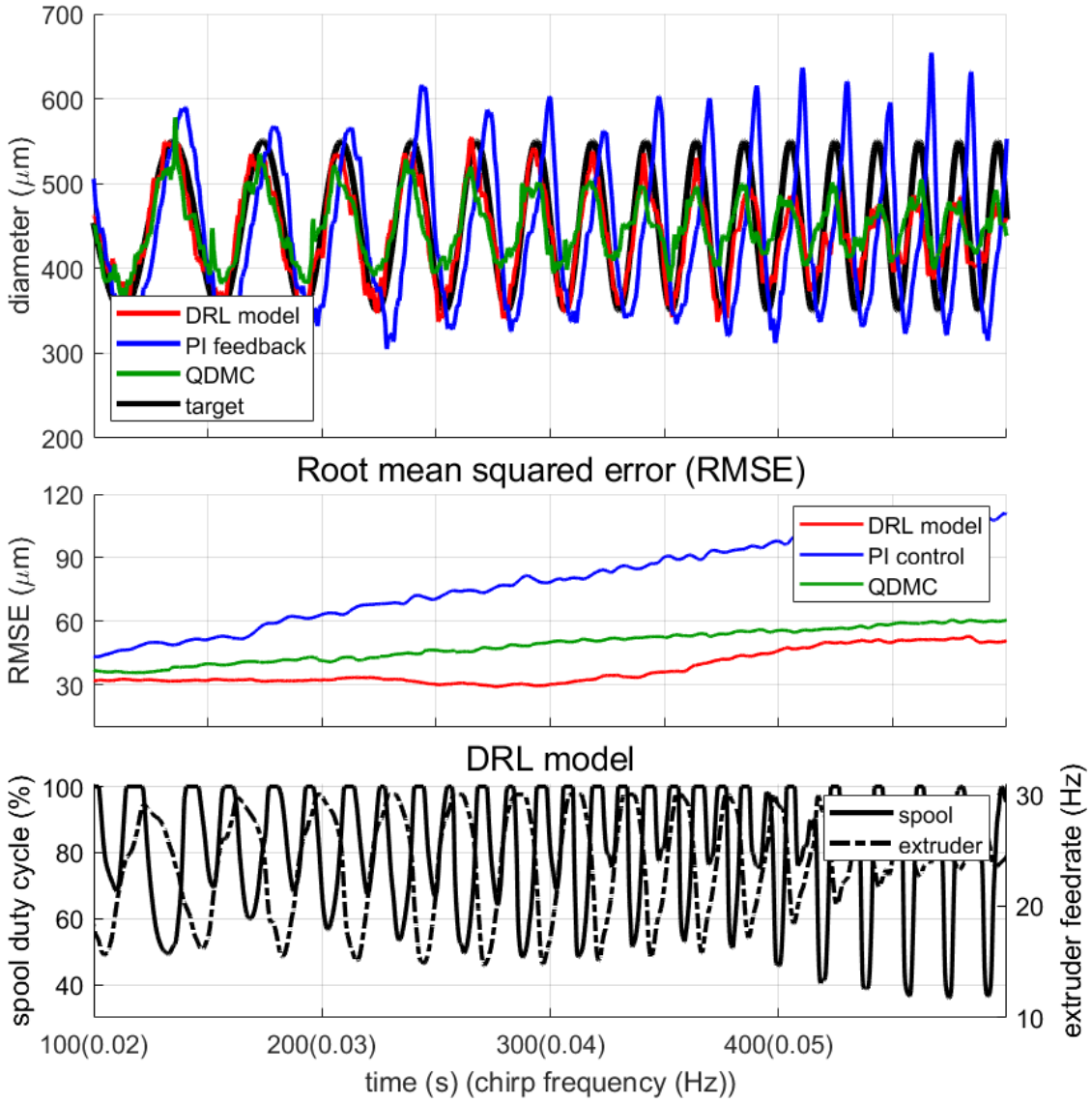


Figure 6-4: (top) Chirp reference target diameter and measured diameter trajectory of DRL model, PI feedback control, and QDMC. (middle) Mean absolute error for the DRL model control, PI feedback control, QDMC. Moving average of window size 500 (125 seconds) is applied. (bottom) The spool's duty cycle and the extruder's feed rate of the DRL model control.

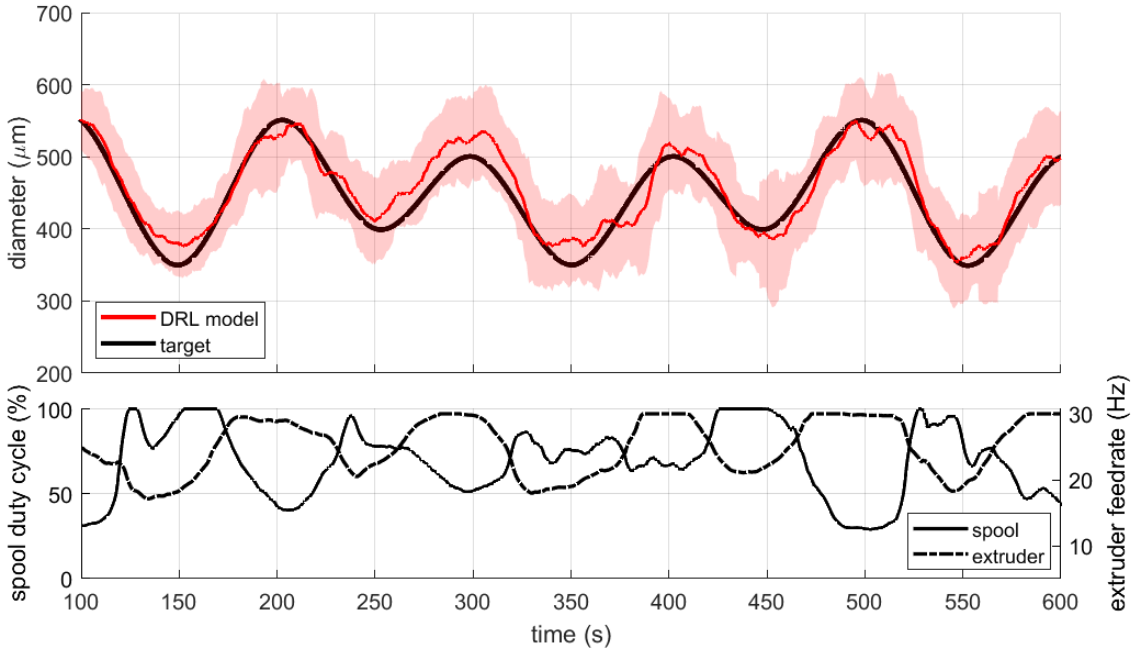


Figure 6-5: (top) Random spline reference target diameter and measured diameter trajectory of DRL model control. (bottom) The spool's duty cycle and the extruder's feed rate of the DRL model controller.

6.1.3 Continuous Target

Although the model was trained using a discontinuous step-changing target, it was tested on continuous target trajectories: sine, chirp (sine sweep) and random spline target (Fig. 6-3, 6-4, 6-5). In the sine reference, the mean and the amplitude were set to 450 μm and 100 μm and the frequency was set to 0.016 Hz (0.1 rad/s). The chirp trajectory swept from 0.01 Hz to 0.06 Hz with a chirpyness 10^{-4} Hz/s. The mean and the amplitude were fixed to 450μm and 100μm. The random spline target was generated by connecting several points with a B-spline curve. The diameter of each points were set between 350μm and 550μm and the time step difference between adjacent points were set between 20 time steps (5 seconds) and 80 time steps (20 seconds) so that it includes various frequency components with various amplitude.

In the sine reference, the DRL model and QDMC showed better performance in terms of diameter error and phase delay. The PI feedback control showed significant delay and overshoot.

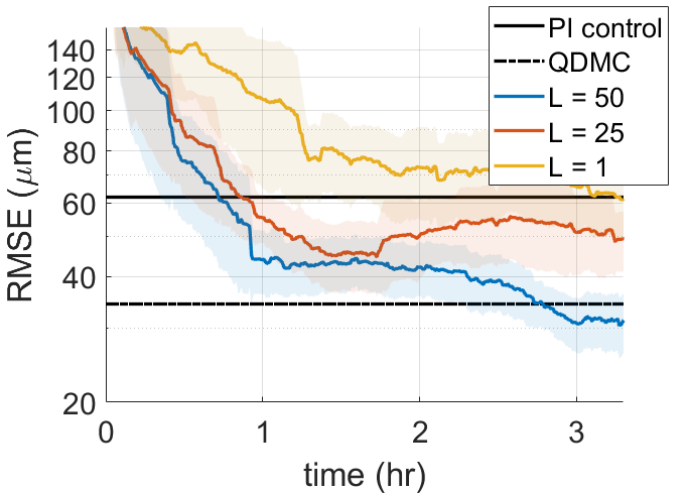


Figure 6-6: Learning curve comparison with regard to the window length. Moving average of 10,000 steps (41.8 minutes) is applied for the average reward.

In the chirp trajectory, all the controllers showed the increasing trend in root mean squared error (RMSE) as the sine frequency increases because there is a physical limit on how fast the system can respond to the input changes. The PI controller showed significantly larger root mean squared error (RMSE) than other methods at all frequency range. The DRL controller and QDMC showed similar RMSE at below 20 mHz and at over 50 mHz. Between 20 mHz and 50 mHz, the DRL controller showed significantly less RMSE than QDMC. It was able to regulate the RMSE under 40 μm until the sine frequency reached 45 mHz, while QDMC was able to regulate only until 25 mHz. It implies that the RL controller has advantages over QDMC especially when the reference trajectory involves fast and continuous change.

The DRL model controller was also able to track the random spline trajectory with various frequency and amplitude by gradually varying the input actions. It shows that the learned controller can be used for not only specific types of trajectory but also other trajectory types that it has never observed during the training process.

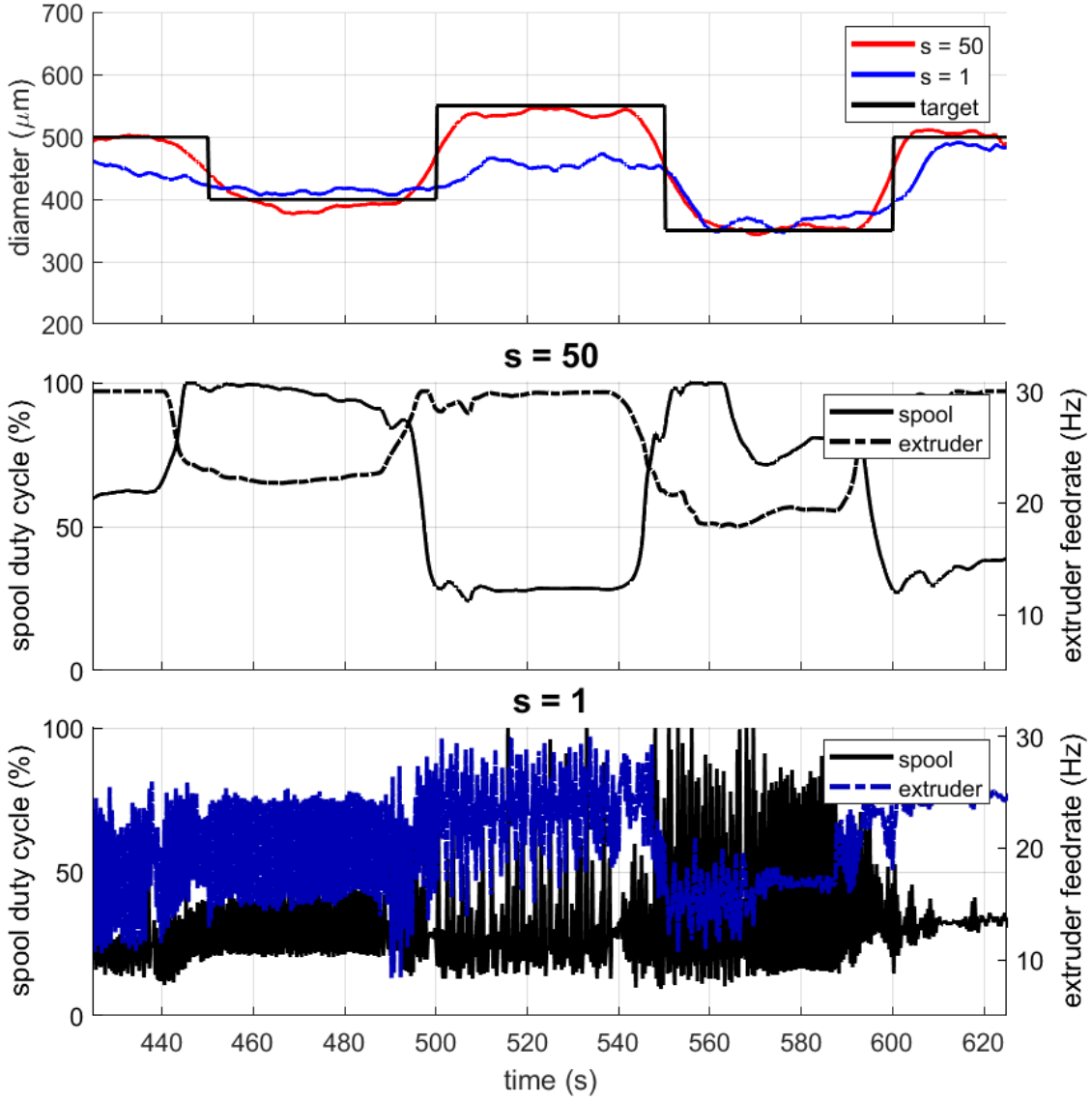


Figure 6-7: (top) Random step reference target diameter and measured diameter trajectory of DRL models with window length 50 and 1. Moving average of window size 40 (10 seconds) is applied. (middle) The spool's duty cycle and the extruder's feed rate of the DRL model controller with window length 50. (bottom) The spool's duty cycle and the extruder's feed rate of the DRL model controller with window length 1.

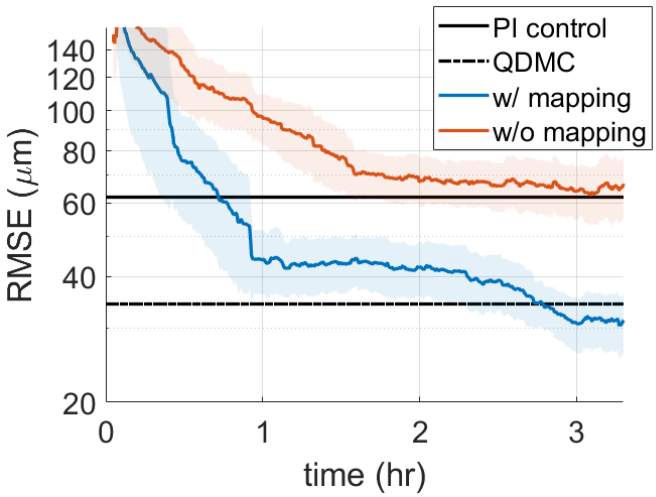


Figure 6-8: Learning curve comparison between the DRL models of with and without the linear mapping. Moving average of 10,000 steps (41.8 minutes) is applied for the average reward.

6.2 Ablative Analysis

6.2.1 Effect of Window Length

Models with several different window lengths are compared. The learning curve comparison shows that the window length must be long enough to achieve optimal performance (Fig. 6-6). When the window length is 1, it computes the input action based on only one time step of observation. Therefore, it cannot consider the previous history of the process. Also, it cannot capture the stochastic nature of the system. As a result, the computed input action fluctuates violently as shown in Fig. 6-7. The model with window length 25 was also not as good as that with window length 50. This is because 25 time steps (6.25 seconds) are not enough to capture the delayed dynamics when the step change occurs. As mentioned earlier, change in the extruder input should occur 8.0 seconds earlier than the diameter step change. Therefore, the window length should be at least 32 time steps (8.0 seconds) to capture these delayed dynamics.

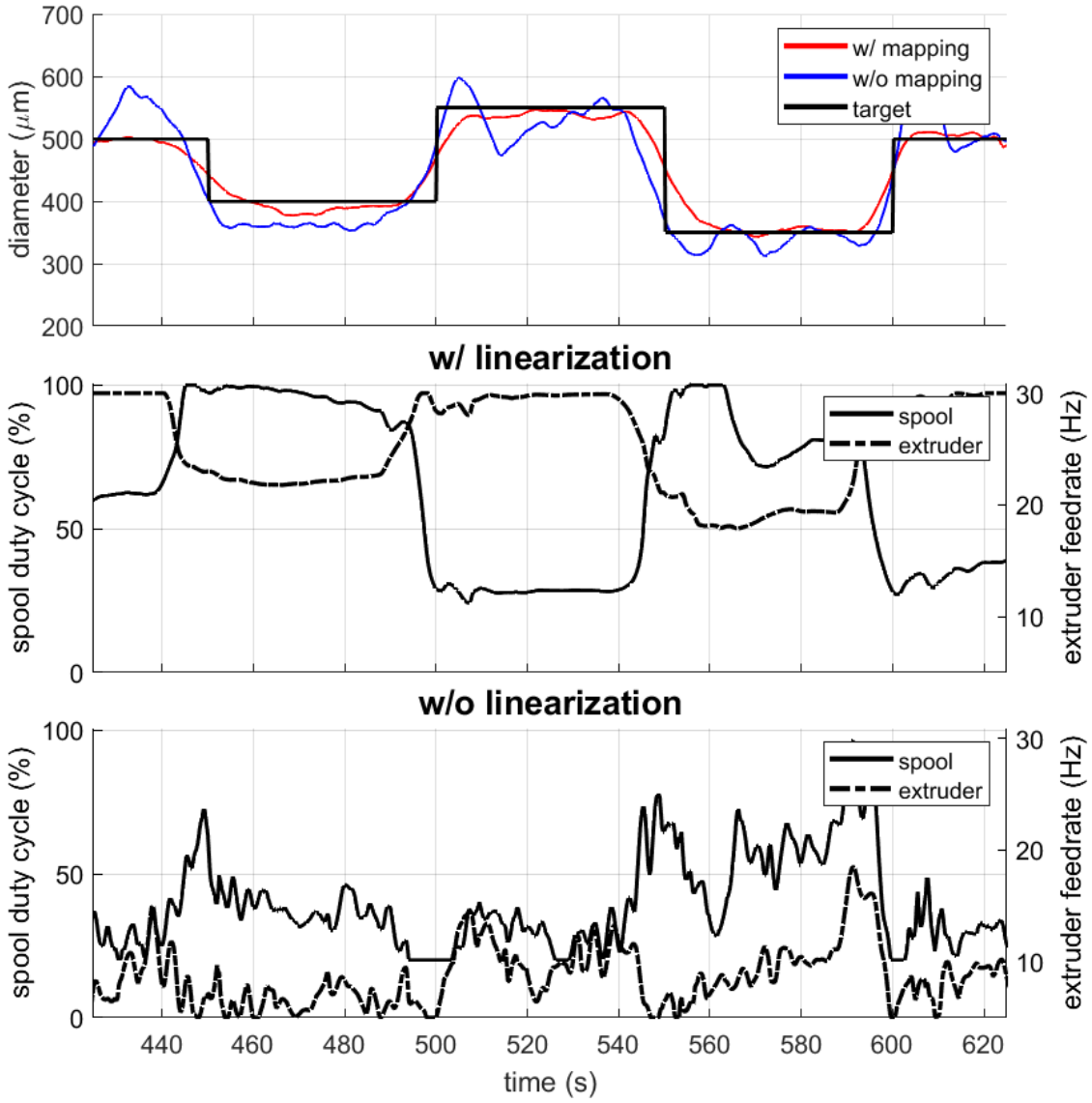


Figure 6-9: (top) Random step reference target diameter and measured diameter trajectory of DRL model with and without the linear mapping. Moving average of window size 40 (10 seconds) is applied. (middle) The spool's duty cycle and the extruder's feed rate of the DRL model controller with the linear mapping. (bottom) The spool's duty cycle and the extruder's feed rate of the DRL model controller without the linear mapping.

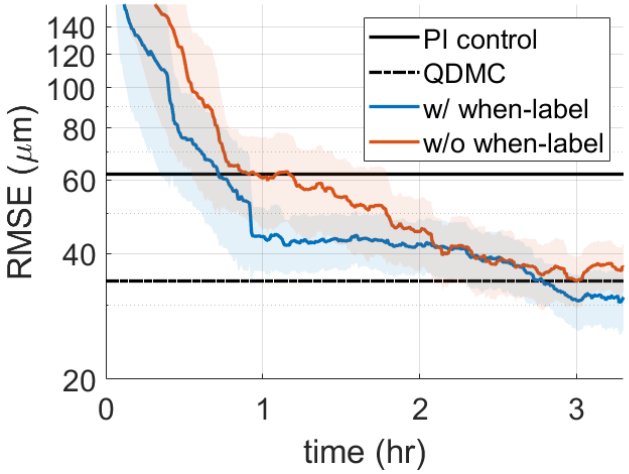


Figure 6-10: Learning curve comparison between the DRL models of with and without the when-label. Moving average of 10,000 steps (41.8 minutes) is applied for the average reward.

6.2.2 Effect of Action-Speed Linear Mapping

Fig. 6-8 shows that the action-speed linear mapping is critical to achieving a good performance. The model without the linear mapping converged to the average reward approximately 0.2 smaller than the model with the mapping. This means that the average diameter error was approximately 20 μm bigger. The model showed poor performance especially when the target diameter was large, where low spool speed is required (Fig. 6-9). This is because it is hard to control the speed precisely at the low speed range if action-speed is not linearly mapped. Linearly mapping the spool action to the speed enables the model to control the speed precisely throughout the entire speed range and result in better performance.

6.2.3 Effect of When-Label

Fig. 6-10 shows that the when-label speeds up the learning, especially at the early phase of the learning. The when-label helps the learning of the model by providing additional information about the time history of when the data was observed. Thereby, the model can learn the process faster than when the label is not provided. Also, the model with the when-label computed more consistent outputs. In compari-

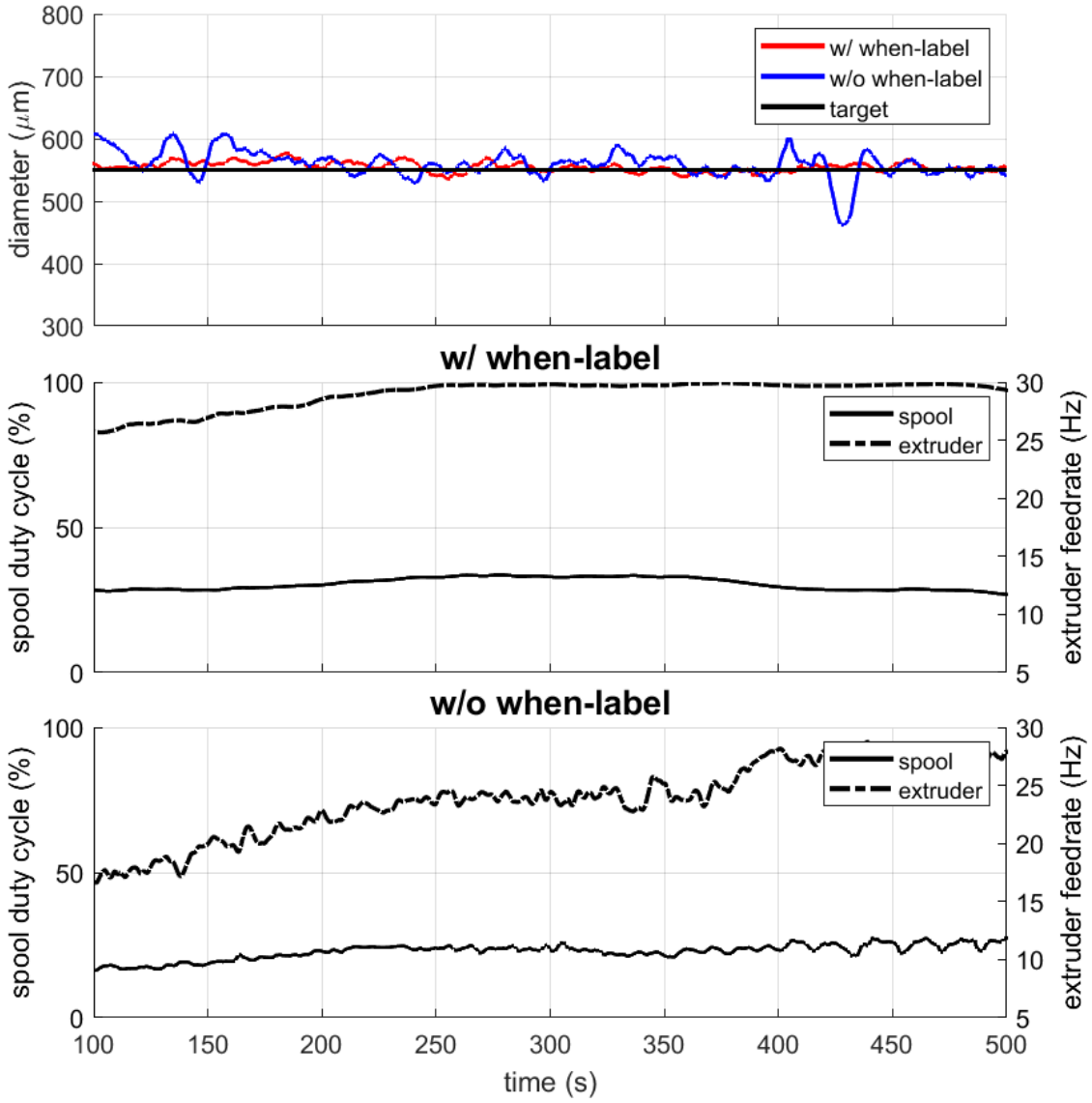


Figure 6-11: (top) Steady reference target diameter and measured diameter trajectory of DRL models with and without the when-label Moving average of window size 40 (10 seconds) is applied. (middle) The spool's duty cycle and the extruder's feed rate of the DRL model controller with the when-label. (bottom) The spool's duty cycle and the extruder's feed rate of the DRL model controller without the when-label.

son, the model without when-label showed some fluctuation in its outputs, as shown in Fig. 6-11. This high-frequency fluctuation is unrealizable since the system cannot physically respond to such high frequency.

Chapter 7

Conclusion and Future Work

We introduced the compact fiber drawing system and implemented the control strategy to it. The drawing system is significantly smaller and less expensive than industrial fiber draw towers, so it is suitable for prototyping fiber and thus can facilitate smart fiber research and novel controllers. We developed a DRL based control method that can be deployed to the desktop system. We focused on regulating the fiber diameter to track various target trajectories. By modifying and customizing DRL algorithms, we were able to improve the performance of the control in terms of tracking error. With neither analytical nor numerical models of the physical system, the controller learned to track various types of target trajectories under the stochasticity and the non-linear delayed dynamics of the system. It was also able to track the target that it had never experienced in the training process.

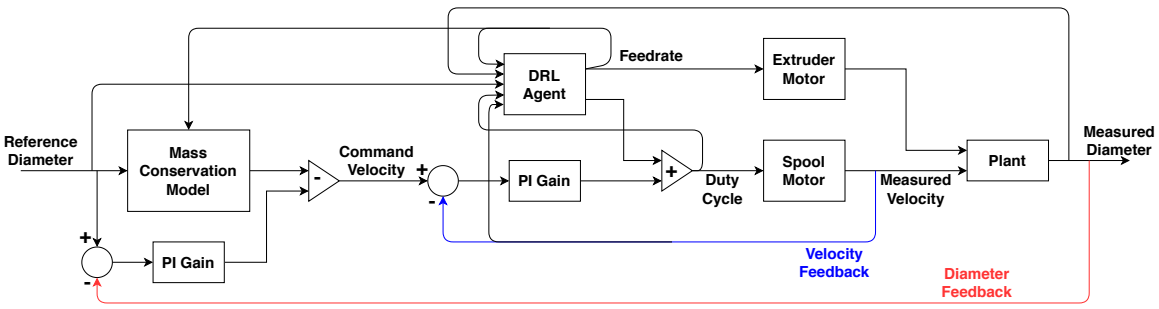


Figure 7-1: Block diagram of the control system that combines the DRL approach with the conventional PI feedback control.

Our DRL algorithm can be easily deployed to other processes that require regulatory control. Especially, it is beneficial when the process is hard to model and requires accurate predictive tracking. For example, it can be used in the process that involves heat/mass transfer or chemical reaction such as coffee roasting and oil refining.

One drawback of the current method is that it requires a lot of data to train the DRL model. While it is not a big problem when the data is cheap, it can make the implementation unrealistic when the cost for data collection is high. On top of that, the exploration noise used for training may cause the safety problem to the system. Therefore, the maximum and minimum boundary of the actions must be carefully decided when designing the model.

As future work, one can develop a way to improve the data efficiency of the training so that the amount of data and time required for training is reduced. For example, the DRL approach and the conventional PID controller can be combined as shown in Fig 7-1. By augmenting the DRL agent to the PI feedback controller, the DRL agent can use the PI controller as the starting point and improve the performance by the training.

Bibliography

[1] Sangwoon Kim, David Kim, and Brian Anthony. Dynamic control of a fiber manufacturing process using deep reinforcement learning. *Unpublished*.

[2] Flurin Wieland, Alexia N. Gloess, Marco Keller, Andreas Wetzel, Stefan Schenker, and Chahan Yeretzian. Online monitoring of coffee roasting by proton transfer reaction time-of-flight mass spectrometry (ptr-tof-ms): towards a real-time process control for a consistent roast profile. *Analytical and Bioanalytical Chemistry*, 402(8):2531–2543, Mar 2012.

[3] S. R. Choudhury, Y. Jaluria, and S. H.-K. Lee. A computational method for generating the free-surface neck-down profile for glass flow in optical fiber drawing. *Numerical Heat Transfer Part A: Applications*, 35(1):1–24, 1999.

[4] U. C. Paek and R. B. Runk. Physical behavior of the neck-down region during furnace drawing of silica fibers. *Journal of Applied Physics*, 49(8):4417–4422, 1978.

[5] A. L. Yarin. Stationary configuration of fibers formed under nonisothermal conditions. *Journal of Applied Mechanics and Technical Physics*, 23(6):865–870, Nov 1982.

[6] S.H.K. Lee and Y. Jaluria. Simulation of the transport processes in the neck-down region of a furnace drawn optical fiber. *International Journal of Heat and Mass Transfer*, 40(4):843 – 856, 1997.

[7] S. Roy Choudhury and Y. Jaluria. Practical aspects in the drawing of an optical fiber. *Journal of Materials Research*, 13(2):483–493, 1998.

[8] A. Mawardi and R. Pitchumani. Optical fiber drawing process model using an analytical neck-down profile. *IEEE Photonics Journal*, 2010.

[9] Z. Yin and Y. Jaluria. Neck down and thermally induced defects in high-speed optical fiber drawing. *ASME Journal of Heat Transfer*, 2000.

[10] Zhilong Yin and Y. Jaluria. Thermal transport and flow in high-speed optical fiber drawing. *ASME Journal of Heat Transfer*, 120, 1998.

[11] Andryas Mawardi and Ranga Pitchumani. Numerical simulations of an optical fiber drawing process under uncertainty. *J. Lightwave Technol.*, 26(5):580–587, Mar 2008.

[12] Susan H. Law, Geoffrey W. Barton, and Thanh N. Phan. The causes and nature of diameter variations along optical fiber, 2005.

[13] A. Mulpur and C. Thompson. Modal diameter control of linear isothermal optical fibers. In *Proceedings of IEEE International Conference on Control and Applications*, pages 433–438 vol.1, Sep. 1993.

[14] A. Mulpur and C. Thompson. Nonlinear control of optical fiber diameter variations. *IEEE Transactions on Control Systems Technology*, 4(2):152–162, March 1996.

[15] S. Tchikanda and Kok-Meng Lee. State space modeling for optical fiber drawing process. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 6, pages 4954–4959 vol.6, May 2002.

[16] S. Tchikanda, Kok-Meng Lee, and Z. Zhou. A state space model for modern feedback control of optical fiber drawing process. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 2, pages 856–861 vol.2, July 2003.

[17] Zhiyong Wei, Kok-Meng Lee, Serge W. Tchikanda, Zhi Zhou, and Siu-Ping Hong. Free surface flow in high speed fiber drawing with large-diameter glass preforms. *ASME Journal of Heat Transfer*, 126, 2004.

[18] Zhiyong Wei, Kok-Meng Lee, and Zhi Zhou. A reduced order model for robust control of optical fiber drawing. volume 73, 01 2004.

[19] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484, January 2016.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[21] Timothy P. Lillicrap, Jonathan J. Hunt, Alexand er Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints*, page arXiv:1509.02971, Sep 2015.

[22] S. P. K. Spielberg, R. B. Gopaluni, and P. D. Loewen. Deep reinforcement learning approaches for process control. In *2017 6th International Symposium*

on Advanced Control of Industrial Processes (AdCONIP), pages 201–206, May 2017.

[23] R. Cui, C. Yang, Y. Li, and S. Sharma. Adaptive neural network control of auvs with control input nonlinearities using reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(6):1019–1029, June 2017.

[24] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, May 2017.

[25] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana. Deep reinforcement learning for high precision assembly tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 819–825, Sep. 2017.

[26] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, Oct 2017.

[27] David D. Kim and Brian Anthony. Design and fabrication of desktop fiber manufacturing kit for education. In *Dynamic Systems and Control Conference*, 2017.

[28] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

[29] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv e-prints*, page arXiv:1512.04455, Dec 2015.

[30] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv e-prints*, page arXiv:1802.09477, Feb 2018.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[32] D. R. Song, C. Yang, C. McGreavy, and Z. Li. Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 311–318, Nov 2018.

[33] F. Altché and A. de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 353–359, Oct 2017.

[34] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *INTERSPEECH-2012*, 2012.

[35] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2016.

[36] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[37] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930.

[38] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014.

[39] C. R. Cutler and B. L. Ramaker. Dynamic matrix control??a computer control algorithm. *Joint Automatic Control Conference*, 17:72, 1980.

[40] CARLOS E. GARCIA and A.M. MORSHEDI. Quadratic programming solution of dynamic matrix control (qdmc). *Chemical Engineering Communications*, 46(1-3):73–87, 1986.

[41] Matthias Freiherr von Adrian-Werberg. Mpc quadratic dynamic matrix controler with soft constraints.