A Novel Approach to Feedback Control with Deep Reinforcement Learning*

Yuan Wang∗ Kirubakaran Velswamy∗ Biao Huang**

* Chemical and Materials Engineering Department, University of Alberta, Edmonton, AB, Canada
** Correspondence: biao.huang@ualberta.ca

Abstract: A novel deep reinforcement learning (RL) algorithm is applied for feedback control application. We propose Proximal Actor-Critic, a model-free reinforcement learning algorithm that can learn robust feedback control laws from direct interaction data from the plant. We show efficacy of the algorithm on a benchmark problem in Heating Ventilation and Air Conditioning (HVAC) heating system, with the RL controller achieving lower Integral Absolute Error (IAE) and Integral Square Error (ISE) as compared to baseline Proportional-Integral (PI) and Linear Quadratic Regulator (LQR) controllers. We also provide details on establishing feedback control problems within the deep reinforcement learning framework, including policy parameterization, neural network architecture and training procedures.

Keywords: Reinforcement Learning, Feedback Control, Deep Learning, Artificial Intelligence, Neural Networks

1. INTRODUCTION

1.1 Motivation

Advanced process control methods are being utilized at an increasing rate across an ever-increasing range of practical, high-impact application areas. Classical control design methods typically require expertise in both controller design and process modelling, requiring development of accurate plant dynamic models for controller development. In addition, changes in operation conditions or variable drifts require additional controller tuning by experienced professionals, increasing down-time and cost of maintenance.

One enticing alternative is to develop general-purpose controllers that can (1) learn closed loop control laws directly from data, (2) minimal tuning needed to generalize to different application areas, ranging from Linear/Non-linear, deterministic/stochastic systems, Single Input Single Output (SISO)/Multiple Input Multiple Output (MIMO) systems to set-point tracking/regulatory control problems.

Reinforcement learning (RL) methods are one class of methods that fit the criteria for the above general-purpose controllers. Traditional controllers have been parameterized to be tuned (Rizvi and Lin (2017)) and also in control optimization (Lewis et al. (2012)) using RL methods. Specifically, model-free RL methods can learn closed-loop control laws directly from plant interaction data without an explicit model identification step. RL methods such as Actor-Critic (Konda and Tsitsiklis (2000)) are model-free algorithms that can utilize neural networks to learn non-linear control laws in non-Markovian plant dynamics. From a controls perspective, these algorithms can be seen as direct adaptive optimal control methods (Sutton et al. (1992)), which adaptively learn from direct plant interaction data, while minimizing some functional of the controlled system’s behaviour. This general-purpose definition means that RL methods can be applied to traditional control applications in both regulation/tracking and optimal control. Recent developments in combining deep learning methods (Mnih et al. (2015)) (Silver et al. (2016)) with reinforcement learning algorithms has led to exciting developments in algorithms that can learn complex control laws from scratch for a variety of applications, ranging from human-level performance on Atari games (Mnih et al. (2013))(Lillicrap et al. (2015)), robotics (Gu et al. (2017))(Peng et al. (2016)), building control (Sahu et al. (2017)) (Wang et al. (2017)) to autonomous vehicles (Zhang et al. (2016)). Data driven robust control using RL scheme has also been proposed (Jiang et al. (2018)).

1.2 Paper Outline

The paper is organized as follows: In section 2 we present a general overview of the RL based control paradigm. In section 3, we provide in-depth description of the reinforcement learning problem set-up as well as an overview of the proposed proximal actor-critic algorithm. In section 4, we provide an overview on class of deep recurrent neural networks termed Long Short Term Memory (LSTM) networks and how it ties into applications within our RL algorithm. Section 5 provides a brief overview of our HVAC air heating simulation set-up. Section 6 provides the Simulation Results and discussion.

2. CONTROL PARADIGM OVERVIEW

Figure 1 shows a general diagram set-up of the reinforcement learning method set-up for control of an arbitrary plant.
Fig. 1. An overview of the reinforcement learning problem.

The control paradigm is similar to that of a typical feedback control scheme. In this set-up, we have an environment (equivalent to the plant), and in addition there is an agent (similar to the controller) that computes control actions at for a given observation of the environment, st. A reward (similar to cost) signal is provided at every time step to indicate the performance of the agent with regards to some metrics. In regulation/ tracking problems, this per-time-step reward signal is typically some form of the per-time-step set-point deviations.

3. PROXIMAL ACTOR CRITIC ALGORITHM

3.1 Markov Decision Processes

The major blocks of a Reinforcement Learning environment are listed below. A mathematical formulation of this environment is based on the Markov Decision Process (MDP).

- s ∈ S: State space, entire set of states feasible for the given environment.
- a ∈ A: Action space, collection of possible actions translating to control effort at every sample time.
- r ∈ R: Reward, for a said action in each sample time, a scalar reward reflected from the environment.
- π(a | s): Policy, deduces an action a based on the state observed s from the environment. Usually, of stochastic nature, with scope for being deterministic also.
- P(s′ | r | s, a): Transition probability distribution, a probability that a reward r is emitted by the environment dynamics from previous step action a while transitioning from state s to subsequent state s′.

3.2 Policy Gradient

The RL agent’s goal is an optimal policy that maximizes the expected sum of discounted rewards, also called returns, Rt, shown in Eq.1. We use γ ∈ [0, 1.0] as the discount factor to keep Rt bounded.

\[ R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]  

The policy, πθ, is a parameterized function that maps from state observations, st, to actions at. One specific parameterization is to utilize a neural network with parameters θ ∈ R^d. Given this neural network-based parameterization, one can then directly optimize the policy to maximize the objective function J(θ), the cumulative total of discounted returns, under the policy πθ.

\[ \max_{\theta} J(\theta) = E\{R_t | \pi_{\theta} \} \]  

Various methods to estimate policy gradient from likelihood ratio method (Glynn (1990)) to finite difference methods (Ng and Jordan (2000)) have been reported. We focus on the former method, specifically REINFORCE policy gradient (Sutton et al. (2000)). The likelihood ratio method’s advantage is that it mitigates the need for small random perturbations to the policy parameters.

The REINFORCE (Sutton et al. (2000)) policy gradient, using the objective function J(θ) given in Eq.(1), calculates the unbiased gradient estimate of the policy.

\[ \nabla_{\theta} J(\theta) = E_{\pi} \left[ \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t, \theta) \sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right] \]  

The naive REINFORCE policy gradient update has high variance, as it is using the actual returns r_{t'} as a component of the gradient estimate. It can be shown that subtracting the returns r_{t'} by an arbitrary baseline term, b(s_t) \in \mathbb{R} can significantly minimize the variance in gradient estimates (Schulman et al. (2015b)). It is evident that baseline does not bias the gradient estimate (Schulman et al. (2015b)), i.e., E_{\pi}[\nabla_{\theta} \log \pi(a_t | s_t, \theta)b(s_t)] = 0.

Thus a careful selection of baseline can greatly minimize the variance alongside sample complexity in the gradient estimates.

With the variance-reducing baseline in hand, we can derive a general policy gradient estimate form that incorporates the baseline with actual returns as:

\[ \nabla_{\theta} J(\theta) = E_{\pi} \left[ \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi(a_t | s_t, \theta) \left( \sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right] \]  

where the term on the right is often called the advantage, A(s_t):

\[ A(s_t) = \left( \sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \]  

3.3 Actor-Critic Architecture

Actor-critic methods are a class of model-free RL algorithms that employ both a parameterized policy (actor) and a parameterized baseline function (critic). During optimization, both the policy and baseline functions are updated asynchronously to improve their estimates. Figure 2 shows the diagram of the actor-critic architecture set-up. It shows the flow of rewards and states from the system to the critic, as well as to the actor.

In our algorithm, we employ a state-value function, v^\pi(s) as our variance reducing baseline. The state-value function is optimized to improve its estimate of the discounted sum of returns, R_t.
Fig. 2. An overview of the architecture setup.

\[ v^\pi(s) = \mathbb{E}_\pi [R_t | s] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s \right] \]  

(6)

With the introduction of the proposed critic, the state-value function baseline \( v^\pi(s) \), we can write the overall policy gradient update as:

\[ \nabla \theta J(\theta) = \mathbb{E}_\pi \left[ \sum_{t=0}^{T-1} \nabla \theta \log \pi(a_t|s_t, \theta) \left( \sum_{t'=t}^{T-1} r_{t'} - v^\pi(s_t) \right) \right] \]  

(7)

Intuitively, the state-value function baseline \( v^\pi(s) \) is estimating an expected cumulative sum of rewards across many trajectories. With this acting as the baseline, we can get an good estimate of how well any particular trajectory was by subtracting rewards received in that trajectory by the baseline state-value function \( v^\pi \).

3.4 Actor and Critic Design

To allow continuous control outputs, we utilize a diagonal-Gaussian policy (parametrized by a neural network) map state observations \( s_t \) into two vectors, \( \mu \) and \( \ln(\sigma^2) \). These vectors are used to parameterize Gaussian distributions from which we sample actions from, \( a_t \sim N(\mu, \sigma) \). Updates to our policy parameters thus affect these two vectors and subsequent parameterized distributions our actions are sampled from.

The critic, \( v_\pi(s) \), is parameterized by another neural network, which estimates the expected sum of rewards at any state, \( R_t \).

For both Actor and Critic, the state observation consists of the six state variables for the HVAC system as well as the set-point deviation.

3.5 Natural Policy Gradient

The goal of policy gradient is to make small updates in parameters to improve the overall objective function, \( J(\theta) \). However, it is not clear whether optimizing policy parameters directly for \( J(\theta) \) provides the optimal gradient ascent direction. In particular, (Kakade (2002)) shows that vanilla policy gradient (direct optimization of \( J(\theta) \)) is sensitive to policy parameterization, even when policy parameter changes are small. Given this, the natural policy gradient (Amari (1998)) was proposed, as it is independent of policy parameterization, and can be utilized for monotonic improvement guarantee of a general stochastic policy (Schulman et al. (2015a)). Specifically, the natural policy gradient is related to the vanilla policy gradient by:

\[ \nabla \theta^{nat} \pi_\theta(s, a) \propto F_\theta^{-1} \nabla \theta \pi_\theta(s, a) \]  

(8)

where \( F_\theta \) is the Fisher information matrix:

\[ F_\theta = \mathbb{E}_{\pi_\theta} [\nabla \theta \log \pi_\theta(s, a) \nabla \theta \log \pi_\theta(s, a)^T] \]  

(9)

An alternative way to represent the Fisher information matrix above is the Hessian of the Kullback-Leibler (KL) divergence between two differential stochastic policies, \( \pi_\theta \) (updated policy) and \( \pi_{\theta, old} \) (old policy before update):

\[ F_\theta = \nabla^2_{\theta, old} D_{KL}(\pi_{\theta, old} \| \pi) \]  

(10)

This derivation provides an intuitive way for us to understand how natural policy gradient update the policy parameters. \( \theta \) is updated to maximize the objective function while being constrained to stay close to its old parameters via the KL constraint.

3.6 Trust Region and KL Constraint

In (Schulman et al. (2015a)), an algorithm based on Natural Policy gradient was developed and is termed Trust-Region Policy Optimization (TRPO). It is a constrained policy optimization problem where at every parameter update, the surrogate objective function was maximized subject to a small, constant KL constraint on the policy changes. TRPO also utilized conjugate gradient method to speed up the computation and inversion of the Fisher information matrix, as well as line-search to ensure the surrogate objective and KL constraints were met.

In our implementation, we opt for an easier to implement method, following an algorithm laid out in (Schulman et al. (2017)). We apply a fixed KL constraint as an additional loss term to our regular policy gradient objective. This method mitigates the need for complexity; scales well with increasing number of policy parameters, and can be easily implemented in any neural network optimization package.

A pseudocode of the algorithm is provided for the same.

4. RECURRENT NEURAL NETWORKS POLICY

Artificial neural networks (ANN) are a class of nonlinear, differentiable, nested function approximators. The specific branch of ANN, often termed Deep Learning, has received unprecedented interests and attention in the past few years from academia and industry alike. For our policy parameterization, we chose to utilize a specific class of recurrent artificial neural networks (RNN), termed Gated Recurrent Unit (GRU)(Chung et al. (2014)).

4.1 Vanilla Recurrent Neural Networks

In vanilla RNNs, the sequential input at every step \( x_t \) is first affine transformed (Eq.8), and then a non-linear activation function \( g(x) \) is applied to the affine transform \( a_t \) as seen in Eq.9. RNNs have persistent hidden unit \( h_t \) that is persistently carried forward at every sequential step.

\[ v^\pi(s) = \mathbb{E}_\pi [R_t | s] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s \right] \]  

(6)


Algorithm 1 Proximal Actor-Critic

Require: Initialize Policy $\pi$ having parameters $\theta_\pi$ and Value Critic $v_\pi$ having parameters $\theta_v$

1: for each episode do
2: Get initial state $s$
3: Initialize Storage buffer $S, A, R, S'$
4: for $t = 1, 2, \ldots, N$ steps do
5: Sample action with policy: $a \sim \pi_\theta(s)$
6: Run action through Environment, obtain reward and post state: $r, s' \leftarrow ENV(s, a)$
7: Collect and store: $S, A, R, S' \leftarrow s, a, r, s'$
8: end for
9: Compute Discount Returns: $\hat{V} = \sum_{t=0}^{N} \gamma^t r_{t+1}$
10: Update $\theta_v$ to Minimize $\sum_{n=1}^{N} \|v_{n}(s_n) - \hat{V}_n\|^2$
11: Compute Advantage: $A_{\text{Advantage}} = \hat{V} + v_{\pi}(S) - v_{\pi}(S')$
12: With learning rate $\alpha$, update policy $\nabla \theta_\pi$ to maximize objective function $L(\theta_{\pi}) = \log \pi(A|S) \cdot A_{\text{Advantage}} - \beta \cdot KL(\theta_\pi, \theta_{\pi}^{\text{old}})$
13: end for
14: 

15: $a_t = Wx_t + Uh_{t-1} + b$
16: $h_t = g(a_t)$

When unfolded through time, the vanilla RNN can be seen as a deep feed-forward neural network, with the depth of the network being the length of the sequential data.

4.2 Gated Recurrent Unit

One issue with the vanilla RNN is the problem of vanishing/exploding gradients when the RNN is used to model long sequences (Pascanu et al. (2013)). One method of alleviating these issues is the use of learnable gate structures within the network. Several popular variants exist, such as LSTM(Gers et al. (1999)) and Gated Recurrent Unit (GRU)(Chung et al. (2014)). We chose to utilize GRU here due to their simpler structure (compared to LSTM) and comparable performance(Chung et al. (2014)).

The GRU has a total of two gates, the update gate $z_t$ and the reset gate $r_t$. The $\sigma$ indicates the use of a Sigmoid non-linear activation function, which squashes its inputs to the range between $[0, 1]$. This acts as a gating mechanism in Eq.13, on the two gates modulate the amount of information flow into the computation.

\[ z_t = \sigma(W_f x_t + U_f s_{t-1} + b_z) \]
\[ r_t = \sigma(W_r x_t + U_r s_{t-1} + b_r) \]
\[ h_t = \tanh(W_c x_t + U_c (s_{t-1} \odot r_t) + b_h) \]
\[ s_t = (1 - z_t) \odot h_t + z_t \odot s_{t-1} \]

We can observe from Eq.12 that the reset gate $r_t$ modulates how to combine the current input with previous memory $s_{t-1}$. And also observe that in Eq.13, the update gate $z_t$ modulates how much information from the previous memory is kept for current computation.

4.3 RNNs for POMDP

Given the nature of most real-life problems, where the Markov properties are violated due to long-term correlations and noise in state observations. These effects change our problem definition from that of a Markov Decision Process (MDP) to that of a Partially-Observable Markov Decision Process (POMDP).

We can alleviate this issue with the introduction of recurrent neural network (RNN) to parameterize both actor and critic. Since RNNs have hidden states, they’re able to learn and infer the true state observations $s_t$ from noisy, correlated observations.

5. HVAC SIMULATION OVERVIEW

For benchmark comparison, we chose a HVAC air heating simulation(Anderson et al. (1997)) to demonstrate a highly relevant, potential application area for our RL based methods. The HVAC process itself contains high nonlinearity and varies drastically at different operating points. In addition, the influence of external variables on the HVAC dynamics makes it a difficult problem to predict and design appropriate control laws. Factors such as changing weather, heating schedule, etc., can have large effects on the HVAC dynamics.

5.1 Control Challenge

The HVAC heating coil is a non-linear, MIMO system with various external disturbances as seen in Figure 3.

Fig. 3. An overview of the HVAC system setup

The system has 3 external process variables not influenced by the controller, $T_{ai}$, the inlet air temperature, $f_n$, inlet cold air flow rate, and $T_{wi}$, the inlet hot water temperature. The hot water flows through a tube and heats up the incoming air. The control action, which is adjusting the hot water flow rate valve, will have an effect on the 3 internal process variables, output air temperature $T_{ao}$, output hot water flow rate $F_w$, and the hot water outlet temperature $T_{wo}$. The goal is to adjust the hot water flow rate valve to control the output air temperature $T_{ao}$ to some set point.

5.2 HVAC Heating Coil Dynamics

The nonlinear process model utilized here is based on one developed by Underwood (Underwood and Crawford (1991)). The model is given below:

\[ f_w(t) = 0.008 + 0.00703(-41.29 + 0.30932 \cdot u_t - 0.3681 \cdot 10^{-4} \cdot u^2 - 9.56 \cdot 10^{-8} \cdot u^3) \]
The HVAC system’s external disturbance variables $f_a$, $T_{wi}$ and $T_{ai}$ are modified by random walk during each sample time to simulate various disturbances from the environment.

The ranges for the random walk are $0.6 \leq f_a \leq 0.9$ [kg/s], $73 \leq T_{wi} \leq 81$ [$^\circ$C], and $4 \leq T_{ai} \leq 10$ [$^\circ$C].

5.3 Baseline Controllers

Baseline controllers were implemented with Proportional-Integral (PI) and Linear Quadratic Regulator (LQR) structures. The Relay feedback method was implemented to obtain an overdamped response, from which a First Order Plus Dead Time (FOPDT) model was derived using the system identification Toolbox in Matlab. Tyreus-Luyben method was used to tune the PI controller.

6. SIMULATION AND RESULTS

6.1 Training and Testing Procedures

For training the RL controller, data from episodic rollouts of 150 sample steps were simulated and collected. 150 sample steps were chosen to allow the RL controller sufficient steps to learn to bring the system to the desired set-point and maintain it. Specifically, for each episode (rollout), we uniformly-randomly initialized all state variables within their permissible range, as well as the desired air output temperature set-point, $T_{ao}$. The RL controller is then allowed to interact with the environment to achieve the desired setpoint within 150 time steps.

For testing, we run our RL controller against baseline controllers for a total of 1000 time steps, where setpoint changes were made at $t = \{250, 500, 750\}$ corresponding to output air temperature setpoint values of $sp = \{45, 48, 45\}$ [$^\circ$C].

Table 1 shows the neural network architecture for the RL controller. We parameterized the Actor network with a two hidden layer recurrent GRU network, and used a simple multilayer perceptron (MLP) as the Critic network.

Table 2. Table showing the Integral Absolute Error (IAE), Integral Square Error (ISE) and controller variance for the three controllers.

7. CONCLUSION

A novel reinforcement learning approach has been developed for feedback control applications. This approach has the benefit of learning control laws directly from data obtained through interactions with plant. An overview of the
reinforcement learning framework, as well as the algorithm termed Proximal Actor-Critic has been provided. This algorithm updates via natural policy gradient, and using a value function baseline and a Kullback-Liebler divergence constraint on the policy update. The method has been demonstrated in a HVAC heating coil simulation, and its performance was compared against PI and LQR baselines. Based on the IAE and ISE metrics, the RL controller, compared to the two baselines, lowered the metrics while resulting in a higher controller variance.

Future research will investigate methods for reducing the controller variance. Potential avenues of research includes applying low pass filters to the raw controller output, to reward shaping to discourage controller variance that does not improve control performance.

REFERENCES
Amari, S.I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2), 251–276.
Anderson, C.W., Hittle, D.C., Katz, A.D., and Kretchmar, R.M. (1997). Synthesis of reinforcement learning, neural networks and pi control applied to a simulated heating coil. *Artificial Intelligence in Engineering*, 11(4), 421–429.
Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
Gers, F.A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.
Glynn, P.W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10), 75–84.
Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA)*, 2017 IEEE International Conference on, 3389–3396. IEEE.
Jiang, H., Zhang, H., Cui, Y., and Xiao, G. (2018). Robust control scheme for a class of uncertain nonlinear systems with completely unknown dynamics using data-driven reinforcement learning method. *Neurocomputing*, 273, 68.
Kakade, S.M. (2002). A natural policy gradient. In *Advances in neural information processing systems*, 1531–1538.
Konda, V.R. and Tsitsiklis, J.N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*.
Lewis, F.L., Vrabie, D., and Vanvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6), 76.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., and Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
Ng, A.Y. and Jordan, M. (2000). Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 406–415. Morgan Kaufmann Publishers Inc.
Pascaru, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 1310–1318.
Peng, X.B., Berseth, G., and Van de Panne, M. (2016). Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4), 81.
Rizvi, S.A.A. and Lin, Z. (2017). Output feedback reinforcement q-learning control for discrete-time linear quadratic regulator problem. In *Proceedings of IEEE 56th Annual Conference on Decision and Control*, 1311–1316.
Sahu, C., Kirubaskaran, V., Radhakrishnan, T.K., and Sivakumaran, N. (2017). Explicit model predictive control of split-type air conditioning system. *Transactions of the Institute of Measurement and Control*, 39(5), 754–762.
Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 1889–1897.
Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.
Sutton, R.S., Barto, A.G., and Williams, R.J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*.
Sutton, R.S., McAllester, D.A., Singh, S.P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.
Underwood, D.M. and Crawford, R.R. (1991). Dynamic nonlinear modeling of a hot-water-to-air heat exchanger for control applications. *American Society of Heating, Refrigerating and Air-Conditioning Engineers*.
Wang, Y., Velswamy, K., and Huang, B. (2017). A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. *Processes*, 5(3), 46.
Zhang, T., Kahn, G., Levine, S., and Abbeel, P. (2016). Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *Robotics and Automation (ICRA)*, 2016 IEEE International Conference on, 528–535. IEEE.