2021

Reinforcement Learning Control for Vapor Compression Refrigeration Cycle

Tech Logg Ding
The University of Auckland, New Zealand, tdin993@aucklanduni.ac.nz

Alison Subiantoro

Stuart Norris

Reinforcement Learning Control for Vapor Compression Refrigeration Cycle

Tech Logg DING*, Alison SUBIANTORO, Stuart NORRIS

The University of Auckland,
Department of Mechanical Engineering,
Faculty of Engineering,
Auckland, New Zealand
tdin993@aucklanduni.ac.nz

* Corresponding Author

ABSTRACT

Vapor compression refrigeration cycles (VCRC) control optimization is an effective method to increase its' reliability and energy efficiency. In modern VCRC systems, the introduction of inverter compressors, electronic expansion valves, fan speed, and pump speed control has significantly increased their controllability. These improvements led to the development of many multivariable and predictive control strategies that improved the system's temperature tracking performance and increased the coefficient of performance (COP) by up to 30% compared to conventional on/off and PID controls. However, a VCRC also has high nonlinearities and parameter couplings, making it difficult to apply these modern control laws. This problem incentivizes the application of reinforcement learning (RL) to optimize VCRC control as RL demonstrated an unprecedented ability to optimize complex control problems. This study explores this idea by using RL to train a direct optimal controller for a VCRC. To test the concept, a VCRC simulation model was developed in the MATLAB Simulink environment to train an RL VCRC controller using the MATLAB reinforcement learning toolbox. The controller's goal is to track the desired internal air temperature and a 10˚C superheat setting. The controller used 17 observations containing the VCRC states, tracking errors, operating conditions, and previous actions to determine the optimal compressor speed and expansion valve opening percentage. The VCRC operating conditions were limited to ambient and internal air temperature ranges of 28-32˚C and 16-20˚C, respectively. This study used the twin delayed deep deterministic policy gradient (TD3) RL algorithm to train the controller. The TD3 training hyperparameters such as the noise model and deep neural network parameters were tuned to balance the exploration and exploitation of the solution space. The training converged to a suboptimal solution after completing 6500 episodes in 5 days using an Intel Core i7-8700 CPU 3.2GHz with 32 GB RAM. The developed RL controller was tested using custom ambient and internal air temperature profiles. The controller tracked both the internal air and superheat temperature settings well with low error and fast response time. However, when the ambient temperature fell below 29˚C, the actuators began to fluctuate, indicating that it did not learn a good policy for this region. This study showed that RL could optimize VCRC control, but more research is necessary to improve it.

1. INTRODUCTION

The vapor compression refrigeration cycle (VCRC) is widely used for various cooling and heating applications in many industries. It consumes approximately 15% of global electricity, contributes around 10% of global greenhouse gas emissions, and these numbers are projected to grow tenfold by 2050 (She et al., 2018). Therefore, there are significant incentives to improve the VCRC's temperature tracking performance and reduce its' energy consumption. This goal can be achieved by improving the VCRC controller to achieve better temperature tracking performance and higher energy efficiency. However, VCRC optimal control is challenging because the cycle has high nonlinearities and complexities (Goyal et al., 2019). A review on VCRC control showed that multivariable and predictive optimal controllers could improve VCRC control performance and energy efficiency by up to 30% compared to conventional controllers (Goyal et al., 2019). In the last decade, machine learning (ML) technology has shown the ability to improve VCRC controllers by enhancing the control design process with highly accurate data-driven models and better optimal

solution search algorithms (Wan et al., 2020). Besides, reinforcement learning (RL) algorithms also showed a high potential to improve VCRC optimal control. For instance, Google's Deepmind showed that a practical and safe supervisory RL controller could be deployed onto their data centers' cooling systems and achieved up to 40% in energy savings (Gasparik et al., 2018). Besides that, research on the application of Q learning RL controls for a CO_2 refrigeration system also showed that RL could find a suitable control strategy for demand-side energy management (Beghi et al., 2017). An RL controller was also developed for an Organic Rankine Cycle and showed great superheat tracking performance for a thermofluid system (Wang et al., 2020). These results demonstrated the potential for RL to optimize VCRC control effectively.

While RL is used to optimize heating, ventilation, and air conditioning systems (Mehmood et al., 2019), research into its application on VCRC control is scarce despite its potential. Hence, this paper intends to demonstrate RL's application for VCRC optimal control and encourage more research in this area. This paper focuses on applying the twin-delayed deep deterministic policy gradient (TD3) RL algorithm to develop a direct optimal RLVCRC controller. The RL controller was trained for a single-stage air-cooled VCRC to track the desired internal air temperature and superheat temperature. The controller was trained using a VCRC model with changing ambient temperature and internal air temperature settings. The RL controller manipulated the compressor speed and electronic expansion valve opening percentage to achieve the control goals. This paper starts by discussing some background knowledge for VCRC, VCRC control, and RL. The paper then discusses the RL controller's application on VCRC, including the training setup and RL controller development. Next, the novel RL controller's temperature tracking performance and results are analyzed and discussed. Lastly, potential future work and conclusions are presented.

2. VAPOR COMPRESSION REFRIGERATION CYCLE (VCRC)

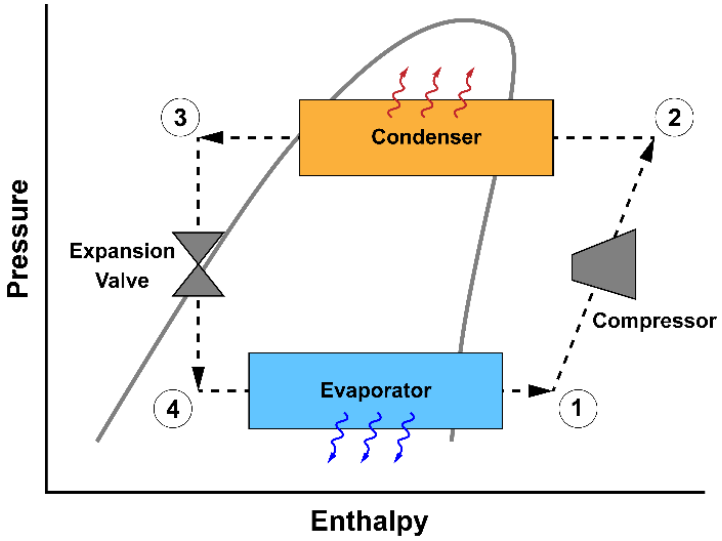


Figure 1: Vapor compression refrigeration cycle

Figure 1 shows the Vapor Compression Refrigeration Cycle (VCRC), where the refrigerant is circulated to exchange heat with external fluids. It has four main components, which are the compressor, expansion valve, evaporator, and condenser. The evaporator and the condenser are heat exchangers that enable heat transfer between the refrigerant and the external fluids. The compressor and expansion valves are actuators that dictate the refrigerant mass flow rate and the pressure difference between the evaporator and the condenser. From points 1 to 2, the compressor does work on the refrigerant and increases its pressure. From points 2 to 3, the refrigerant passes through the condenser that expels heat from the refrigerant to the hot reservoir by condensing the refrigerant at high pressure and temperature. From points 3 to 4, the refrigerant is expanded through an expansion device to reduce its pressure and saturation temperature. From points 4 to 1, the refrigerant passes through the evaporator that absorbs heat from the cooled fluid by evaporating the refrigerant at low pressure and temperature. The refrigerant then flows back into the compressor and completes the cycle. Both heat exchangers also have an actuator that controls the external fluid flow rate that alters the amount of heat transferred. Throughout the cycle, the thermophysical and heat transfer properties of the refrigerant change constantly. Hence, the VCRC has high nonlinearities, complexities, and parameter couplings that make it challenging to model and control optimally.

2.1. VCRC Modelling

An air-cooled VCRC simulation model was developed for this research. It consisted of the compressor, expansion valve, evaporator, condenser, receiver, accumulator, connecting pipes, and the external environment. In this project, only the compressor speed and expansion valve opening were manipulated. Both fan speeds were kept constant. In a VCRC, the two heat exchangers' dynamics are much slower than the actuators' dynamics. Hence, the two actuators were modeled as quasi-steady components using empirical maps and steady-state conservation equations (Rasmussen & Alleyne, 2006). The heat exchangers were modeled using the Moving Boundary (MB) method that treated every refrigerant fluid phase as a single control volume with variable length. The average fluid properties in every control volume were used to determine its state and heat transfer properties. The refrigerant fluid thermal-physical properties were modeled using a lookup table populated using the CoolProp library (Bell et al., 2014). Lastly, the condenser's ambient side was modeled as an infinite air reservoir, and the cooled space on the evaporator side was modeled as a constant volume air system.

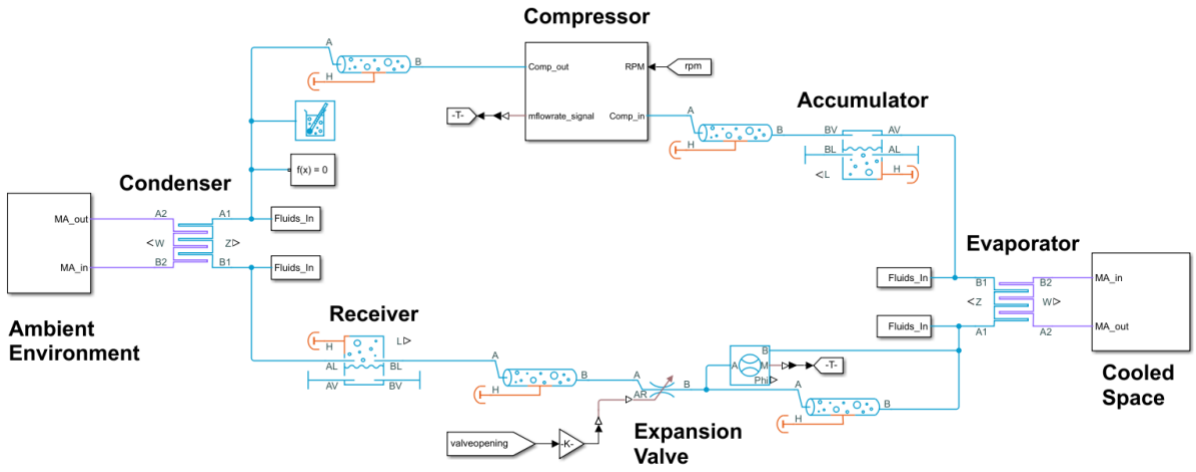


Figure 2: MATLAB Simulink VCRC simulation model

In this project, the MATLAB Simulink environment was used to develop the simulation model shown in Figure 2. The two-phase fluid Simscape Toolbox was used to model the VCRC components, and the model was tuned using data presented in a thesis (Rasmussen & Alleyne, 2006). The compressor speed and valve opening percentage were between 500-3500 RPM and 17-25%. Besides that, the internal air and ambient temperatures were between 16-20˚C and 28-32˚C, respectively. A constant 800W heating source was also added to represent the product load. The Simscape 2P-MA heat exchanger component model was used to develop the MB heat exchanger models. The heat transfer between the refrigerant and air was modeled using the effectiveness-NTU method. Next, the variable-speed compressor and electronic expansion valve were modeled using empirical relationships provided in the thesis. The refrigerant states, its' pressure and specific internal energy, and other refrigerant properties such as its' density and temperature were treated as perfectly measurable values. Although the model could not be adequately validated due to the lack of experimental data, the model trends, dynamics, and state values seemed realistic. Therefore, this model is sufficient to demonstrate the application of a reinforcement learning (RL) controller for VCRC control. The final model was compiled into a system of nonlinear differential equations that were solved using Simulink's ode15s solver.

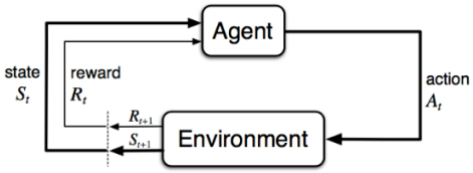
2.2. VCRC Control

The objective of VCRC control is to track the room temperature and the desired superheat while consuming minimal energy. The common VCRC actuators are the compressor, expansion valve, and the pump or fan for the external fluids. These actuators control the VCRC mass flow rate, evaporating pressure, and condensing pressure to meet the cooling load requirement and maintain the desired internal air temperature. Conventionally, the superheat and internal air temperature are tracked individually using the expansion valve and compressor, respectively. The conventional VCRC controllers are the on-off, PID, and LQR controllers. On-off controls are easy to apply, but it is known to be inefficient, and it can also reduce the compressors' shelf-life due to the constant fluctuations. PID and LQR feedback control laws perform better than on-off control, but they only work on the linear time-invariant (LTI) systems. A VCRC is not an LTI system; hence linearizing the model about a single operating point affects its' accuracy

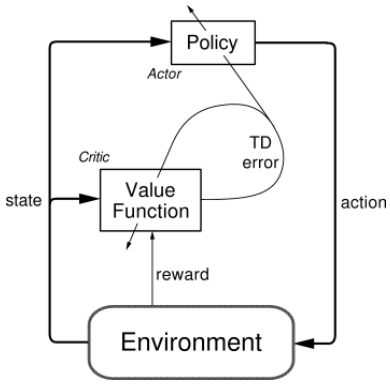
significantly because of the wide range of operating conditions. Therefore, these linear control laws are insufficient for accurate optimal control of VCRC. In literature, cascaded control loops with higher-level supervisory controllers and lower-level reference tracking controllers are used to mimic nonlinear VCRC control (Goyal et al., 2019). The standard supervisory controllers are the gain scheduling algorithms, adaptive algorithms, model predictive control (MPC) optimizers, and real-time optimizers (RTO). These supervisory controllers either change the lower-level controller gains to approximate a nonlinear controller or outputs precomputed optimal references. These control strategies work better with the nonlinearities and parameter couplings in VCRC and produce better temperature tracking performance that results in energy saving (Goyal et al., 2019). These facts suggest that RL has the potential to improve VCRC control because it can work with nonlinear, complex, and multi-objective optimization problems.

3. REINFORCEMENT LEARNING (RL) CONTROLLER DESIGN

3.1. RL control



(a) Markov Decision Process



(b) Actor-Critic structure

Figure 3: Reinforcement learning structures (Sutton, 2018)

This section aims to provide an intuitive and brief overview of Reinforcement Learning (RL) based control. The RL controller design goal is to find an optimal control policy by training an agent. The "policy" is the optimal control law, and the "agent" is the optimal controller. RL is a class of machine learning algorithms that learn optimal actions that maximize the reward using observations from the environment. RL controllers showed great potential for direct optimal control when performing complex tasks such as controlling walking robots (Sutton, 2018). In RL control design, the controller is named the agent, and everything else is called the environment. The environment is formulated as a Markov Decision Process (MDP), which is a discrete-time stochastic control process (Lillicrap et al., 2015). Figure 3(b) shows the MDP process, where it views the environment as a set of states (S). Using the states as inputs, the agent then takes an action (A), that causes a state to transition into a new state and returns a reward (R). RL algorithms use this MDP structure to learn the best actions that produce the best rewards. Prior to the training process, three components are required. The first component is the controlled environment, including the simulation model, disturbances, sensors, and more. Next, a policy representation is required to relate the observations to the optimal state in the agent. A typical policy representation is a deep neural network (DNN) that approximates the relationship between the observations and the optimal actions. The third component is the reward function that is a scalar function that aims to reward "good" behavior and "penalize" bad behavior. For the training process, an RL algorithm needs to be selected based on the application and complexity. The algorithm's training hyperparameters must then be tuned to balance the exploration and exploitation of the solution space. Lastly, the training episodes need to be designed to ensure that the agent is exposed to all operating conditions. This step is crucial because RL control policies are black-boxed; hence, they are only guaranteed to work with conditions that they were trained in.

The actor-critic structure in Figure 3(b) and the learning process will be discussed here (Sutton, 2018) to provide an intuitive view of the RL algorithm and training. Some common RL terms and their definitions are listed in Table 1 to supplement this discussion. In an actor-critic reinforcement learning agent structure, there are 2 Deep Neural Networks (DNN), which represents the actor (μ) and critic (Q). The DNN is used to model the relationship between inputs and outputs. The actor is the controller that takes in the state observations and outputs the optimal action. A critic is an evaluation tool that measures the value of state-action pairs by determining their Q values. A vital advantage of this structure is that it combines both the value-based and policy-based RL algorithms to improve the search for the

optimum solution. The actor and critic DNNs learn the optimal parameters by minimizing their respective loss functions. The goal of the critic is to predict the correct Q values for all state-action pairs. Therefore, the critic's loss function is the Q value prediction error. This error can be measured by calculating the error between the observed Q value and the "current" Q value. Therefore, the critic uses a value-based RL algorithm to update itself. For the actor, the objective is to correlate the observed states to the best action that maximizes the expected return. The expected return is defined as the discounted sum of the Q values for the states. Hence, the actor's loss function is determined using the derivative of its' objective function with respect to the network parameters (Yoon, 2019). The derivatives tell the RL algorithm how much to change and in which direction to achieve the goal. The actor updating process is also referred to as the gradient ascent, a policy-based RL algorithm.

Table 1: Common reinforcement learning terms

Term	Definition
Q value	The Q value defines the expected discounted cumulative reward given the observed state-action pair. In layman's terms, the Q value shows how good it is to take an action when a state is observed.
Neural Network parameters, θ	The neural network parameters are the weights and biases for all the connections between every layer and every node in the deep neural network. They are the terms adjusted during the learning process to output the optimal output from the inputs.
Value-based algorithm	Value-based algorithms determine the best action based on the learned Q values. Hence, the algorithm updates the Q value predictions for every state-action pairs.
Policy-based algorithm	Policy-based algorithm directly updates the policies' parameters to increase the likelihood of taking actions that return the most reward. This is typically done using gradient ascent.
Mini-batch training	Mini-batch training refers to training the DNN using small batches of randomly sampled experiences instead of collecting all the experiences then updating the network. This method increases training efficiency significantly.
Experience tuples	The experience tuple is based on the MDP and consists of the current states, actions, reward, and the next states (S, A, R, S').
Experience buffer	The experience buffer stores the collected experience tuples to be used when training the RL agent. The algorithm will sample mini-batches of experiences from this buffer.

Many RL algorithms are available, but this paper will only discuss the twin-delayed deep deterministic policy gradient (TD3) algorithm (Fujimoto et al., 2018). The TD3 algorithm was developed to improve the popular deep deterministic policy gradient (DDPG) algorithm developed by researchers at Deepmind (Lillicrap et al., 2015). While DDPG is the most used algorithm for continuous control problems, it tends to overestimate the value functions and cause undesired convergence towards suboptimal solutions. The value overestimations also cause training instabilities and high sensitivity towards the training hyperparameters. This issue limits the application of DDPG to complex control problems with large solution spaces. TD3 introduced three critical improvements to the DDPG algorithm to address these issues (Fujimoto et al., 2018). They are the introduction of a second critic network, delayed update of the target network, and target policy smoothing noise. These mechanisms reduce overestimations and high variance in Q value predictions and produce more regular training. The TD3 algorithm also has several key features that are explained here. The TD3 algorithm uses a target actor (μ') and critic (Q') to stabilize the RL training process. The target networks are a copy of the real actor-critic and are updated periodically to avoid sudden and abrupt parameter updates. Another key TD3 component is the actor noise model, N, which is usually an Ornstein Uhlenbeck noise model. The noise is added to the actions to observe more state-action pairs and their rewards. The degree of exploration can be tuned by changing the noise model mean, variance, and decay rate. A discount factor, γ, between 0 and 1, is also used in TD3 to indicate how important future rewards are for every time step. Finally, the TD3 algorithm is described below. Refer to the referenced paper for a more in-depth explanation (Fujimoto et al., 2018):

1. Initialize 2 critics DNN parameters, $\theta_{Q_{1,2}}$, and 1 actor DNN parameters, θ_{μ}, randomly and independently.
2. Copy the initialized parameters over to the target actor, $\theta_{\mu'}$ and target critics, $\theta_{Q_{1,2}}{}'$.
3. Initialize the experience replay buffer to the specified size.
4. For each training step:
    1) For the current state observations *S*, select an action, $A = \mu(S) + N$.

2) Observe reward R and next state observation S'.
3) Save the experience tuple (S, A, R, S') from this time step in the experience replay buffer.
4) Sample a random mini-batch of experiences (S_i, A_i, R_i, S_i') from the experience replay buffer.
5) If the current state, S_i, is at the end of an episode, set the value function target to $y_i = R_i + \gamma \times min\left(Q_{1,2}'\left(S_i', clip\left(\mu'(S_i'|\theta_\mu) + \varepsilon\right)|\theta_{Q_{1,2}'}\right)\right)$. The value function target is the sum of the experience reward and the discounted future reward.
6) Update each critic network's parameters by minimizing the mean squared loss L between the updated Q value and the original Q value, as shown in the equation below.
$$L_{1,2} = \frac{1}{M}\left(y_i - Q_{1,2}(S_i, A_i|\theta_{Q_{1,2}})\right)^2$$
7) Update the actor policy parameters using the sampled policy gradient to maximize the expected future return, $\mathbb{E}$. To do so, the gradient of the objective, or the expected reward function $J(\theta)$, is taken with respect to the policy network parameters using the chain rule, as shown below. This is gradient ascent.
$$J(\theta) = \mathbb{E}[Q(s,a)]$$
$$\nabla_{\theta\mu} J \approx \frac{1}{M}\sum_{i=1}^{M} \nabla_Q \nabla_A$$
$$\nabla_Q = \nabla_A Q(S_i, A_i|\theta_Q) \; where \; A = \mu(S_i|\theta_\mu)$$
$$\nabla_A = \nabla_{\theta\mu}\mu(S_i|\theta_\mu)$$
8) Update the target actor and critic parameters every second training step.

3.2. RL controller for VCRC

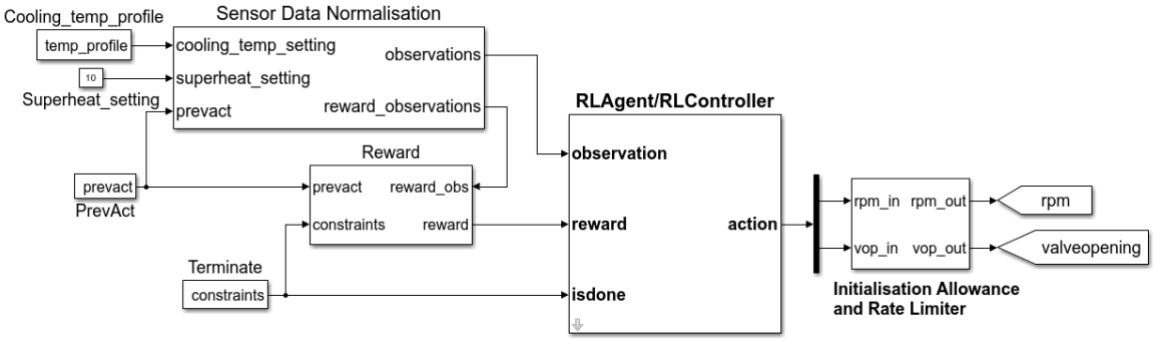


Figure 4: Reinforcement learning controller structure

Figure 4 shows the RL controller structure developed for this project. The controller manipulates the compressor speed and the valve opening percentage, ranging between 500-3500 rpm and 17-25%. The RL controller's goal was to track the desired room temperature and track a 10˚C superheat degree. The controller was fed 17 observations to approximate a fully observable VCRC. These observations included the internal temperature tracking error, the superheat tracking error, the ambient temperature, the desired internal air temperature, the current internal air temperature, the average refrigerant pressure and specific internal energy at the evaporator and condenser, the compressor and valve mass flow rates, the power consumed, and the previous controller action. All observations were normalized to prevent biases by subtracting their means. For the learning and simulation process's stability, all observations were also delayed by one sample time to avoid algebraic loops in the model.

Next, the RL training hyperparameters were tuned to ensure that the agent can find and model the optimal control policy. The critic and actor both had a conventional DNN structure with two hidden layers for every path. Each layer had 400 nodes with ReLU activation functions, which was sufficient for this problem. The TD3 training hyperparameters were set to a critic learning rate of 5×10^{-4}, an actor learning rate of 1×10^{-4}, a target smooth factor of 1×10^{-3}, an experience buffer length of 2×10^{6}, a mini-batch size of 512, and a discount factor of 0.99. The critics' learning rate was twice the actor's learning rate to ensure that the critic converged first. This setting allows the actor to learn from the correct Q network value predictions. The noise model was also tuned to encourage more exploration of the solution space. The initial action noise model variance was defined as 10% of the respective actuator ranges and decayed to a minimum of 1% with a variance decay rate of 1×10^{-6}.

An RL training reward function (r) was developed to encourage the desired VCRC control behavior. The reward function was inspired by the cost and objectives functions from VCRC model predictive control (MPC), Linear Quadratic Gaussian (LQG), and supervisory setpoint optimization controllers. Firstly, the reward function rewarded low internal and superheat temperature tracking errors. The function also penalized high energy consumption, fluctuations, and safety constraints breaches. The final reward function is described in Equation 1 and its' terms are listed below. The error reward terms had a progressive reward structure to encourage the RL agent to aim for lower tracking errors. Next, both the power and fluctuation terms punished the undesired high-power consumption and actuator fluctuations. The fluctuations are punished to encourage stable and smooth actuator commands. Lastly, to promote faster learning, the training episode was terminated and punished if the cooled temperature limits are breached. The superheat was not treated as such to allow more opportunity for the RL agent to optimize the superheat behavior.

$$r = 0.1\left(r_{te} + r_{she} + r_{fluc} + r_{lc} + r_{power} + r_{lim}\right) \qquad \text{Equation (1)}$$

- Temperature tracking error reward: $r_{te} = -2(e_t > 3) + 2(2 < e_t \leq 3) + 4(1 < e_t \leq 2) + 6(0.5 < e_t \leq 1) + 7(0.1 < e_t \leq 0.5) + 10(e_t \leq 0.1)$, where e_t is the temperature tracking error.
- Superheat tracking error reward: $r_{she} = -2(e_{sh} > 5) + 2(3 < e_{sh} \leq 5) + 4(2 < e_{sh} \leq 3) + 6(1 < e_{sh} \leq 2) + 7(0.5 < e_{sh} \leq 1) + 10(e_{sh} \leq 1)$, where e_{sh} is the superheat temperature tracking error.
- Fluctuations penalty: $r_{fluc} = -15\left(\frac{da_t}{dt} \cdot \frac{da_{t+1}}{dt} < 0\right)$, where a is the controller's actions
- Large change penalty: $rl_c = -15\left[(a_{t+1} - a_t) > \left(0.25 \times a_{range}\right)\right]$, where a_{range} are the actuator ranges.
- Power reward: $r_{power} = -1\left(\frac{power}{350}\right)^2$
- Constraints breach penalty: $r_{lim} = -3000\left(T_c < T_{c,min} \parallel T_c > T_{c,max}\right)$

The training episodes were designed to be simple, efficient, and cover the entire operating conditions range. The ambient temperature, internal air temperature setting, and initial cooled space temperature were randomly selected within the operating range. Every episode was designed to run for 500 seconds, which results in a maximum of 500 experiences per episode. No predefined temperature profiles were used for this problem as the training would take a much longer time to complete. The episodes were terminated if the internal air temperature exceeds 22˚C and 14˚C. It was not terminated when the superheat degree hits 0˚C because it would significantly hinder the learning process. Instead, a punishment was introduced when the superheat degree was 0˚C.

4. RESULTS

4.1. VCRC RL Controller results

The RL training completed 6500 episodes in 5 days and converged to a suboptimal solution using an Intel Core i7-8700 CPU 3.2GHz with 32 GB RAM. The VCRC RL controller was tested using a set of ambient and internal air temperature profiles, as shown in Figure 5. The results showed that the RL controller tracked the internal air temperature and superheat temperature settings closely for all regions except when the ambient temperature fell below 29˚C. In this region, the internal air and superheat temperatures fluctuated significantly, indicating that the RL controller failed to learn the region's optimal controller actions. This behavior could because the valve opening percentage lower limit was too high. The valve opening percentage plot showed that the opening had a pulsing behavior at the 17% minimum value, which suggested that this inference could be correct. This problem can also be caused by other reasons, including insufficient training time, imperfect training setup, imperfect DNN structures, inaccuracy of the VCRC simulation model, and more. For the internal air temperature tracking, the steady-state error was within 0.1˚C, and the response time was fast. For the superheat tracking performance, the steady-state error was within 0.5˚C as the reward function in section 3.2 did not incentivize it to lower the error any further. Looking at the actuator plots, the agent showed that it learned how to control both actuators simultaneously to achieve the control objectives. The RL agent also showed that it was aware of changes in the model's disturbances and references and reacted accordingly. The power consumption plot showed that power consumption was generally kept low. However, as the superheat degree setting was fixed, the power consumption might not be optimized. Therefore, an alternative control strategy that tackles VCRC energy consumption needs to be developed to reduce energy usage. In summary, these results showed the ability of the RL controller to optimize VCRC controls.

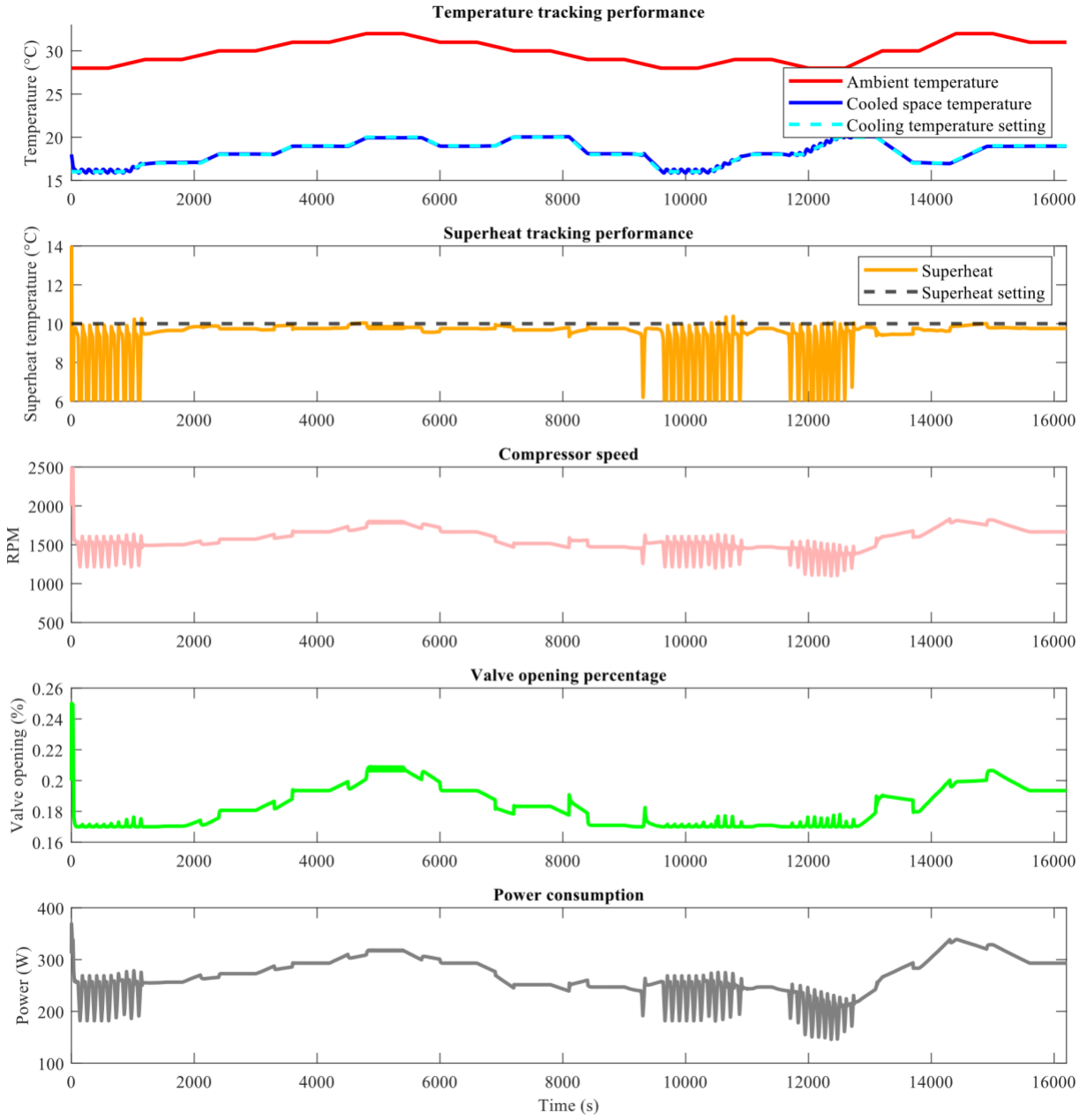


Figure 5: RL controller results

4.2. VCRC RL Controller Implementation Discussion

The VCRC RL controller development process also showed several significant findings that are discussed here. The first issue was the initialization of the VCRC simulation model for every training episode. A fixed set of initial values such as the initial pressure and initial compressor speed must be predefined and used at the start of every episode. This factor leads to a problem where the observations close to the initialization values will produce some false rewards, which affects the RL training process. However, over an extensive range of sampled steps and experiences, this problem could become insignificant. It is also important to note that RL agents cannot be configured to always take a fixed initial action; therefore, an external component is necessary. In this project, a switch was used to pass the initial actuator commands at the start of the simulation. Another important issue is the noisy actuator fluctuations that were produced when the ambient temperature was below 29˚C. This problem leads to instabilities when training and simulating the VCRC. Hence, a rate limiter was used to limit the actuators' dynamics to prevent excessive fluctuations.

In this project, the model used a rate limiter block to impose a 10 rpm/s maximum slew rate when the ambient temperature was below 29˚C. Next, the DDPG and TD3 algorithms were also compared in this project, and it was found that TD3 was significantly better. The shortcomings of DDPG in section 3.1 were present, and the learning process was not stable. The agent frequently converged to a poor suboptimal solution when using DDPG. However, it is also important to note that both DDPG and TD3 RL algorithms suffer from sensitivity towards hyper-parameters tuning as there is no exact method to tune them.

5. FUTURE WORK

The results showed that more research is required to develop a VCRC RL controller fully. Firstly, the training process needs to be further tuned and improved to produce a more comprehensive learning process for the VCRC control problem. This process would include testing various hyperparameters, reward functions, episode designs, and RL algorithms to find the best RL structure for VCRC control optimization. Besides that, to make it a viable solution, a principal component analysis must be conducted to identify the most critical inputs to reduce the number of sensors required. Next, state estimation models need to be developed to reduce the need for expensive sensors if necessary. Ultimately, a standardized economical method to develop optimal RL controllers for VCRC control problems should be developed. Another critical future work for this project is to improve the VCRC simulation model using an in-house test rig. The test rig is currently in development, and it will be used to tune the model. Besides that, the RL controller should also be developed for a wider range of operating conditions. Ideally, the RL controller should be able to work under all weather conditions all year long. The air humidity should also be accounted for by the controller. Next, the controller's robustness should be addressed. As an RL controller is black-boxed, it is crucial to provide confidence that it is safe and reliable. Therefore, additional safety features such as fault protection systems and sensor measurement filters need to be developed for VCRC RL controllers. Alternatively, the RL agent could also be trained in a noisy environment to learn how to be more robust. It is also essential to test and compare the RL controller to current state-of-the-art VCRC controllers. In literature, complete verification of the RL controller on a real system is rare, but it is essential to prove that it genuinely works. Finally, a time-domain analysis of the controller's response should be performed and compared to existing controllers.

6. CONCLUSION

A reinforcement learning controller was developed for vapor compression refrigeration cycle control. The control objectives were to track an internal air temperature reference and a 10˚C superheat temperature. A VCRC simulation model was developed using MATLAB Simulink to train the RL controller. The RL agent was trained using the twin-delayed deep deterministic policy gradient (TD3) algorithm and converged to a suboptimal control policy, or control law, after 6500 training episodes. The training took five days using an Intel Core i7-8700 3.2GHz CPU with 32 GB RAM. The VCRC ambient and internal air temperature ranges were between 28-32˚C and 16-20˚C, respectively. The RL controller learned to use 17 observations to output two optimized controller actions, the compressor speed and the expansion valve opening percentage. The observations included information on the temperature tracking error, superheat temperature tracking error, operating conditions, energy consumption, and previous controller actions. A reward function that rewarded low temperature tracking errors, robustness, and stability was used for training. The training hyperparameters, including the learning rates, noise model, and discount factor, were also tuned to balance exploration and exploitation of the solution space during training. The RL controller's actor and critic used deep neural networks with two hidden layers and 400 nodes in each layer to approximate the control policy and value function. Next, the completed RL controller was tested using custom ambient and internal air temperature profiles. Generally, the RL controller showed great temperature tracking performance with minor steady-state errors and high response time for internal air and superheat temperature tracking. However, the controller actions showed fluctuating behavior when the ambient temperature fell below 29˚C, indicating that the algorithm failed to generalize this region's controller action. In conclusion, this research showed that the RL could optimize VCRC control, and it should be explored further.

References

Beghi, A., Rampazzo, M., & Zorzi, S. (2017). Reinforcement Learning Control of Transcritical Carbon Dioxide Supermarket Refrigeration Systems. *IFAC-PapersOnLine, 50*(1), 13754-13759. doi:https://doi.org/10.1016/j.ifacol.2017.08.2565

Bell, I. H., Wronski, J., Quoilin, S., & Lemort, V. (2014). Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library CoolProp. *Industrial & Engineering Chemistry Research, 53*(6), 2498-2508.

Fujimoto, S., Van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *arXiv Preprint arXiv:1802.09477,*

Gasparik, A., Gamble, C., & Gao, J. (2018). Safety-first ai for autonomous data centre cooling and industrial control. *DeepMind Blog,*

Goyal, A., Staedter, M. A., & Garimella, S. (2019). A review of control methodologies for vapor compression and absorption heat pumps. *International Journal of Refrigeration, 97*, 1-20. doi:https://doi.org/10.1016/j.ijrefrig.2018.08.026

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., . . . Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv Preprint arXiv:1509.02971,*

Mehmood, M. U., Chun, D., Zeeshan, Han, H., Jeon, G., & Chen, K. (2019). A review of the applications of artificial intelligence and big data to buildings for energy-efficiency and a comfortable indoor living environment. *Energy and Buildings, 202*, 109383. doi:https://doi-org.ezproxy.auckland.ac.nz/10.1016/j.enbuild.2019.109383

Rasmussen, B. P., & Alleyne, A. G. (2006). *Dynamic modeling and advanced control of air conditioning and refrigeration systems.* Air Conditioning and Refrigeration Center. College of Engineering …).

She, X., Cong, L., Nie, B., Leng, G., Peng, H., Chen, Y., . . . Luo, Y. (2018). Energy-efficient and -economic technologies for air conditioning with vapor compression refrigeration: A comprehensive review. *Applied Energy, 232*, 157-186. doi:https://doi.org/10.1016/j.apenergy.2018.09.067

Sutton, R. S. (2018). In Barto A. G. (Ed.), *Reinforcement learning : an introduction* (Second edition.. ed.) Cambridge, Massachusetts : The MIT Press. 2018.

Wan, H., Cao, T., Hwang, Y., & Oh, S. (2020). A review of recent advancements of variable refrigerant flow air-conditioning systems. *Applied Thermal Engineering, 169*, 114893. doi:https://doi.org/10.1016/j.applthermaleng.2019.114893

Wang, X., Wang, R., Jin, M., Shu, G., Tian, H., & Pan, J. (2020). Control of superheat of organic Rankine cycle under transient heat source based on deep reinforcement learning. *Applied Energy, 278*, 115637. doi:https://doi-org.ezproxy.auckland.ac.nz/10.1016/j.apenergy.2020.115637

Yoon, C. (2019). Deep Deterministic Policy Gradients Explained. Retrieved Dec 8, 2020, from https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b