Interactive Reinforcement Learning for Adaptive Thermal Comfort

Ata Korkusuz

Delft University of Technology

TU Delft

Interactive Reinforcement Learning for Adaptive Thermal Comfort

by

Ata Korkusuz

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday September 25, 2024 at 1:00 PM.

Student number:	5821959	
Project duration:	December 13, 2023 – September 25, 2024	
Thesis committee:	Prof. dr. ir. L. C. Siebert,	TU Delft, supervisor
	Prof. dr. ir. B. Dudzik,	TU Delft
	P. Rutgers,	Next Sense, supervisor

This thesis is confidential and cannot be made public until September 24, 2024.

Cover: Air Monitor Abstract Concept Illustration (Image by vectorjuice on Freepik) (Modified)

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Designing and implementing effective systems for thermal comfort management in buildings is a complex task due to the need to account for subjective preference parameters influenced by human physiology, bias and tendencies. This research introduces a novel approach to simulating human interactions for managing thermal comfort. Stochastic simulated humans provide feedback in the form of thermostat interactions, from which their thermal comfort is inferred converting these interactions into rewards, called human rewards. Control policies are obtained from training with Human reward or PMV reward by utilizing the Proximal Policy Optimization (PPO) algorithm. It is shown that the learning process can be guided solely by human rewards. Experiment results assess the impact of this simulated human reward system on the adaptability of the reinforcement learning model for single human scenarios, also comparing back to the PMV reward case as ground truth. The policy trained with PMV reward achieves thermal control that keeps the PMV values inside the [-0.2,0.2] range, while the policy trained with the human reward achieves a range of [-0.6,0.6]. Simulating human feedback as an interaction with the thermostat, the proposed model is shown to capture a rough estimate of human thermal preference. This research paves the way for using simulated humans for interactive reinforcement learning (RL) based thermal comfort control.

Preface

Preferences define the person we are. Whether it is the simple passions and desires that drive us or morals and rational that guide us, each person is only separated by a thin and inconsistent line called preference. For that reason, I believe that the loss of preference is no different than the death of self. In this modern world where it is common for the louder to suppress the silent and easy for the silent to follow the loud, preserving one's preferences comes with a heavy toll. Therefore, it falls on the aware to make the silent heard. Someone can show their preferences through even the smallest gestures. Interactive reinforcement learning makes it possible to capture preferences to an extent that surpasses the estimates made with general assumptions.

By living and working in indoor spaces, we are constantly faced with the question of whether are we happy with the thermal conditions even though the question is not explicit. While it is simple to adjust the thermal conditions, it is not so simple to know what the ideal condition we should aim for is. It is even harder to adjust for a group of people that you may not even be a part of in the case of thermal comfort in the workspace. With this thesis, I hope to show that it is possible to capture the preferences of people through their natural interactions with their environments.

I would like to express my heartfelt gratitude to my academic supervisor, Dr. Luciano Cavalcante Siebert, for his invaluable support in making this thesis possible and his patience throughout the process. I also wish to extend my thanks to my company supervisors Pim Rutgers and Adalberto Guerra Cabrera for sharing their insights and expertise, which greatly contributed to the development of this thesis.

Ata Korkusuz
Delft, September 2024

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
API	Application Programming Interface
HVAC	Heating, Ventilation, and Air Conditioning
MDP	Markov Decision Process
NaN	Not a Number
PMV	Predicted Mean Vote
PPO	Proximal Policy Optimisation
PID	Proportional-Integral-Derivative
PI	Proportional-Integral
RL	Reinforcement Learning

1

Introduction

Thermal comfort refers to the state of mind expressing satisfaction with the surrounding thermal environment, influenced by factors such as air temperature, humidity, and air movement [1]. In the context of office environments, maintaining optimal thermal conditions is essential for ensuring the well-being and productivity of employees. Uncomfortable thermal conditions can lead to decreased concentration, lower productivity, and increased absenteeism, significantly impacting organizational efficiency and morale [2].

Achieving thermal comfort in office environments is further complicated by the diverse preferences of occupants. Individual factors such as age, gender, clothing, and metabolic rate can lead to varying thermal comfort requirements, making maintaining a uniformly comfortable environment challenging. Additionally, personal preferences and perceptions of comfort can differ widely among employees, with some preferring cooler conditions while others favour warmer temperatures. This variability necessitates a dynamic and adaptive approach to thermal management that can adjust to the unique needs of each occupant. Addressing these preference concerns is crucial for achieving true thermal comfort.

Heating, Ventilation, and Air Conditioning (HVAC) systems are essential for maintaining thermal comfort in office environments. Traditional HVAC control methods often rely on thermostat-based systems, which regulate temperature to maintain a set point. More advanced methods, such as Proportional-Integral-Derivative (PID) controllers, provide finer control by continuously adjusting the HVAC output based on the difference between the desired and actual temperatures. Rule-based control systems utilize predefined logic to manage HVAC operations based on time schedules or occupancy sensors [3]. While these methods can be effective in maintaining general thermal conditions, they often fall short in addressing the diverse and dynamic thermal comfort preferences of individual occupants. As a result, there is an increasing interest in developing more adaptive and intelligent control strategies, such as RL, to better cater to the unique thermal comfort needs of office workers [4]–[8].

In light of the significant advancements achieved by reinforcement learning (RL), there is a growing interest in applying these methodologies to address intricate real-world challenges. Thermal comfort is one such challenge that has a significant potential to benefit from these advancements in RL. However, the effective deployment of RL methods in a socio-technical context, particularly where human interactions with artificial agents are prevalent, necessitates thoughtful consideration of individual preferences and limitations.

1.1. Problem Statement

Despite the widespread use of traditional HVAC control systems, they exhibit significant limitations in addressing the dynamic and individualized nature of thermal comfort preferences in office environments. While effective at maintaining a general temperature set point, thermostat-based systems and PID controllers lack the adaptability to respond to the varying comfort needs of different occupants. Although somewhat more flexible, rule-based systems still operate on predefined logic that does not account for real-time changes in occupancy and individual preferences [3]. Current RL-based thermal control

methods, despite working aware of occupancy, lack any adaptation to personal preference as the thermal comfort metric used is limited to PMV (further discussed in section 2.3)[4]. These limitations often result in a one-size-fits-all approach, leading to discomfort for many employees who do not fit the average thermal profile. Additionally, the inability to adjust to personal comfort preferences dynamically can decrease overall satisfaction and productivity over time as the people utilizing the same environment change. Consequently, there is a pressing need for more human preference-aware adaptive control mechanisms that can dynamically and accurately maintain thermal comfort tailored to individual needs.

Maintaining thermal comfort in office environments presents several specific challenges. Firstly, the diverse thermal preferences among occupants make establishing a universally comfortable temperature difficult. Factors such as individual metabolic rates, clothing choices, and personal sensitivities to temperature variations contribute to these differences [1]. Secondly, occupancy patterns in offices are often unpredictable, with fluctuations throughout the day as people move in and out of different spaces or throughout the year as people prefer to come to the office or not. This variability requires a responsive system that can adjust in real-time to changing conditions [2], [9]. Furthermore, the physical layout of office spaces, with varying levels of insulation, exposure to sunlight, and airflow patterns, adds another layer of complexity to achieving uniform thermal comfort. Addressing these challenges requires a sophisticated approach that can integrate multiple data types and dynamically adjust HVAC thermostat operations to meet the diverse and evolving needs of office occupants[9].

RL offers a fitting solution to the variability and unpredictability of preference problems. The ability of RL to handle problems where the solution is not known in advance and must be learned through exploration is promising as the exploration of how to satisfy an individual's thermal preference is what lies at the core of the problem. RL is also suitable for problems with complex, high-dimensional state spaces where traditional methods may struggle. As the thermal dynamics of a space is quite complex, such an approach is mandatory. RL also offers the flexibility to adapt to dynamic environments where the conditions or rules may change. Although it might be easy enough to keep a constant accepted temperature setpoint, such a setpoint might not always be optimal for thermal comfort. The optimal may change depending on the surroundings (season, air circulation, unexpected temperature drops, etc.) as well as on the individual (change of personal, sickness, activity, etc.). By being able to adapt to such variations, RL has the potential to create smarter and more resilient control agents.

1.2. Objectives and Scope

This thesis project focuses on developing an RL model designed to regulate office air temperature setpoints by incorporating user feedback. Recognising comfort as a subjective and personal experience, the primary objective is to create a learning system that engages with users, accounting for both the user's preference and the office environment's detailed state. Utilizing an RL model is expected to introduce adaptability to the system which is lacking in the single setpoint or the predefined logic approaches. It is also expected to remove the need for trying to estimate the human parameters that affect thermal comfort metrics correctly.

The scope of this project has been narrowed down specifically in terms of the actuator and the human feedback to better focus the research. The RL model only controls the air temperature setpoint while optimizing for thermal comfort. This constraint is in place as the air temperature setpoint is the most straightforward actuator parameter to both set and explain. This choice simplifies the design, installation and maintenance when deploying the model in real life while making the research applicable for virtually all HVAC systems as it is almost always readily available. Meanwhile, human feedback regarding their thermal comfort is collected through thermostat interaction. Considering that occupants' expectations of satisfaction with the indoor environment drive their interactions with devices, equipment, and energy systems in buildings, one or a set of multiple possible interactions could be chosen for interpreting their thermal comfort. Specifically, the interaction of adjusting the thermostat settings has been chosen as it is the most direct and natural response when uncomfortable with the thermal conditions. This choice also allows for the implementation of the proposed model with no extra setup or time taken from the occupants' routine.

Research Question

This thesis aims to develop an RL model that learns to maximize thermal comfort in a way that considers individual human preferences. To do this, the following main research question is formulated:

How can reinforcement learning be effectively applied for HVAC control to enhance thermal comfort, considering individual preference?

The following subquestions are formulated to assist with the clarity and scope of the thesis:

- How can interactive reinforcement learning be used to incorporate human interaction?
- How can thermal preferences be extracted from human interaction?

1.3. Thesis Structure

This thesis consists of 6 chapters. The introduction has explained the motivation and the research question that drives this research. Chapter 2, background, will briefly go over the necessary background information that the thesis builds on top of, as well as the relevant previous works in the literature that coincide with the thesis topic. Chapter 3, methodology, first explains how the RL model architecture is set to learn individual preferences for thermal comfort and change the thermostat setpoint accordingly through Energyplus. Then the chapter describes the experimental setup used to obtain comparable results fit for ablation analysis. Chapter 4, results, reveals the model's output in terms of its training and control performance. Finally, chapter 5 and chapter 6 consist of the discussion and conclusion respectively, summarizing the findings and listing possible future work.

2

Background and Related Works

2.1. Thermal Comfort

Thermal comfort is defined as the condition of mind that expresses satisfaction with the thermal environment. It is a subjective measure, influenced by environmental and personal factors. Achieving thermal comfort means creating an indoor climate that occupants find neither too hot nor too cold, but just right, allowing them to maintain their core body temperature without exerting unnecessary thermoregulatory effort.

Factors Affecting Thermal Comfort

Several factors affect thermal comfort, which can be broadly categorized into environmental and personal factors[1], [4]:

- Environmental Factors:
  - Air Temperature: The temperature of the air surrounding the occupant. It is the most direct and influential factor in determining thermal comfort.
  - Radiant Temperature: The surrounding surface temperature (walls, ceilings, floors). These surfaces can emit or absorb heat, affecting how warm or cool a person feels.
  - Humidity: The amount of moisture in the air. High humidity can hinder the body's ability to cool through sweating, while low humidity can cause dryness and discomfort.
  - Air Velocity: The speed of air movement around the occupant. Air movement can enhance heat loss through convection and evaporation, making a space feel cooler.
- Personal Factors:
  - Clothing Insulation: The amount of thermal insulation provided by a person's clothing. More clothing increases warmth, while less clothing allows more heat to dissipate.
  - Metabolic Rate: The rate at which the body generates heat. It varies with activity level, with higher activity increasing heat production and lower activity reducing it.
  - Age and Gender: These can influence thermal comfort preferences, as metabolic rates and thermal sensations can differ among different age groups and between genders.
  - Acclimatization: People who are acclimatized to a particular climate may have different thermal comfort preferences compared to those from different climates.

Guidelines and Standards for Thermal Comfort

Several guidelines and standards have been developed to ensure that indoor environments provide adequate thermal comfort. These provide objective criteria and recommendations for creating comfortable thermal conditions:

- ASHRAE Standard 55: The American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) Standard 55 specifies the conditions for acceptable thermal environments and defines the methods for measuring and evaluating thermal comfort. It considers factors such as temperature, humidity, air speed, and personal factors, providing a comprehensive approach to achieving thermal comfort.
- ISO 7730: The International Organization for Standardization (ISO) 7730 standard provides guidelines for determining and interpreting thermal comfort using calculation methods. It introduces the Predicted Mean Vote (PMV) and Predicted Percentage of Dissatisfied (PPD) indices to quantify thermal comfort levels.
- EN 15251: The European Standard EN 15251 specifies the indoor environmental parameters required to achieve thermal comfort in buildings, including temperature ranges, humidity levels, and ventilation rates. It emphasizes the importance of designing buildings and HVAC systems to meet these parameters to ensure occupant comfort.

These guidelines and standards are crucial for designing and maintaining indoor environments that promote occupant comfort and well-being. By adhering to these standards, buildings can provide a more consistent and satisfying thermal environment, reducing discomfort and enhancing productivity and health.

In this research, explicitly the ASHRAE 55 definition of thermal comfort and the PMV calculation method suggested by ISO 7730 has been used. Therefore, PMV calculations are done such that they only depend on the mean air temperature, mean radiant temperature, air relative humidity, relative air speed felt, metabolic rate, and clothing insulation. This implies that when taking PMV as a measure of thermal comfort, the factors affecting thermal comfort are assumed to be limited to those used to calculate PMV. It is worth noting that research surrounding thermal comfort criticises the limitations of the PMV scale suggesting that it fails to capture psychological and physiological differences between people [4]. Different adaptive thermal comfort models have been proposed that take into consideration the variances in tolerance to heat or cold as well as the lack of linear correlation between the PMV scale and the voicing of dissatisfaction [10]. The specific usage of PMV as a metric of thermal comfort in this research is explained in more detail under the section 3

2.2. Reinforcement Learning

Reinforcement learning is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment to achieve a specific goal. The fundamental concepts in RL include agents, states, actions, and rewards.

Agent is the decision-maker or learner in the RL framework. It takes actions based on the current state of the environment to maximize some notion of cumulative reward over time. The agent continually learns from the consequences of its actions, refining its strategy to improve performance.

State is a representation of the current situation or configuration of the environment. It encapsulates all the relevant information that the agent needs to make a decision. States can be fully observable, where the agent has complete information about the environment, or partially observable, where some aspects of the environment are hidden from the agent.

Action is a set of all possible moves the agent can take in a given state. The agent's objective is to select actions that will lead to desirable outcomes. Actions can be discrete, such as moving in a specific direction, or continuous, such as adjusting the speed of a vehicle.

Reward is a scalar feedback signal received by the agent after taking an action in a specific state. The reward signals how good or bad the action was in terms of achieving the agent's goal. The agent's objective is to maximize the cumulative reward, known as the return, over time. Rewards can be immediate or delayed, making the learning process more complex when long-term rewards are considered.

The interaction between these components is typically modelled as a Markov Decision Process (MDP), where the agent transitions from one state to another by performing actions, receiving rewards, and

updating its knowledge to improve future performance. This framework provides a robust foundation for developing algorithms that enable agents to learn optimal policies, which are mappings from states to actions that maximize cumulative rewards.

Interactive Reinforcement Learning

Interactive Reinforcement Learning (Interactive RL) extends the traditional RL framework by incorporating interactive elements that can significantly improve the learning process [11]. In standard RL, the agent learns solely through its interactions with the environment, typically in a trial-and-error fashion. Interactive RL, however, introduces additional sources of information and feedback to expedite learning and improve performance. Key components and concepts of Interactive RL include[12]:

Human-in-the-Loop: In many Interactive RL systems, human experts or users can provide, through some input, guidance to the agent. This can take the form of demonstrations, where the human shows the agent how to perform certain tasks, or through direct feedback, where the human provides positive or negative evaluations of the agent's actions. This can significantly accelerate learning, especially in complex environments where pure trial-and-error would be inefficient or impractical.

Exploration-Exploitation Trade-off: Interactive RL often employs strategies to balance exploration (trying new actions to discover their effects) and exploitation (choosing actions that are known to yield high rewards). Human feedback can help guide exploration, making it more targeted and effective. For example, a human can suggest promising actions or warn against risky ones, thus enhancing the agent's ability to discover optimal strategies.

Reward Shaping: In Interactive RL, reward shaping is a technique where additional rewards are provided by a human or an automated system to guide the learning process. These shaped rewards can be designed to reflect intermediate goals or milestones, helping the agent to learn complex tasks more effectively by breaking them down into simpler subtasks.

Adaptive Learning: Interactive RL systems can adapt their learning processes based on the feedback and guidance they receive. This can involve adjusting the learning rate, altering exploration strategies, or modifying the reward structure to better align with human input. This adaptability allows for more efficient and effective learning, especially in dynamic or uncertain environments.

Interactive Queries: The agent can actively query humans for specific information or clarification when faced with uncertainty. This interactive querying can help the agent resolve ambiguities or make better-informed decisions, further enhancing the learning process.

Interactive RL leverages the synergy between human expertise and machine learning capabilities, leading to faster and more robust learning. This approach is particularly beneficial in complex, real-world applications where traditional RL methods may struggle to achieve optimal performance within a reasonable timeframe. By incorporating human knowledge and interactive feedback mechanisms, Interactive RL provides a powerful tool for developing intelligent agents capable of handling intricate tasks in dynamic environments.

Proximal Policy Optimisation

Proximal Policy Optimisation (PPO) [13] is an RL algorithm that has gained popularity due to its robustness, efficiency, and ease of implementation. Developed by OpenAI, PPO strikes a balance between the need for complex optimisation techniques and the simplicity required for practical use. It is particularly well-suited for environments with continuous action spaces and has been effectively applied to various complex control tasks[14].

PPO is an actor-critic method, utilizing two primary components: the actor, which determines the actions to be taken, and the critic, which evaluates the actions by computing value functions. The algorithm's key innovation lies in its approach to policy updates, ensuring stable and reliable learning[13].

PPO has a clipped surrogate objective. The primary purpose of PPO is to improve the policy in a way that prevents drastic updates, which can destabilise the learning process. PPO uses a clipped surrogate objective function to achieve this. The function incorporates a clipping mechanism that limits the change in policy between updates, ensuring that the new policy does not deviate excessively from the old one. Mathematically, the objective is expressed as follows:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{ clip} \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_k}(s, a) \right) \tag{2.1}$$

While the PPO-clip updates policies via:

$$\theta_{k+1} = \arg\max_\theta \mathbb{E}_{s,a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \tag{2.2}$$

PPO does advantage estimation through the advantage function, which measures how much better or worse a particular action is compared to the average action taken from a given state. This helps the algorithm focus on actions that are better than expected, reinforcing good behaviour, and discouraging poor choices.

PPO often includes an entropy bonus in the objective function to encourage exploration and prevent premature convergence to suboptimal policies. This promotes a certain level of randomness in action selection, ensuring that the agent continues to explore the state space.

The algorithm steps are as follows:

- Initialize Policy and Value Networks: Start with initial parameters for the policy (actor) and value (critic) networks.
- Collect Trajectories: Run the policy in the environment to collect trajectories (sequences of states, actions, rewards, and next states).
- Compute Advantages: Use the value network to compute the advantage estimates for each action taken during the trajectories.
- Optimize Surrogate Objective: Use the collected data to perform multiple epochs of optimization on the clipped surrogate objective, updating the policy network parameters.
- Update Value Network: Simultaneously, update the value network to better estimate the value function using the collected rewards.
- Repeat: Repeat collecting trajectories and optimizing the policy and value networks until convergence.

PPO has been successfully applied in various domains, including robotics, playing games, and automated control systems [15], [16]. It is preferred due to its stability, reliability, sample efficiency, and ease of implementation [16]. PPO is used in this thesis for training the agent controlling the thermostat setpoint for these reasons.

2.3. Related Works

The previous literature on temperature and thermal control revolves around HVAC systems. Model predictive control for HVAC systems was initially classified into three categories: classical control, soft control, and hard control[3]. PID controllers and similar subsystems that do modulation of controlled variables according to error dynamics fall under classical control. Particle swarm optimization [17], adaptive self-tuning PI controllers [18], [19] and dynamic building simulation [20] expand on classical control while still only controlling the room air temperature. While it is not that common to see advanced extensions of PI or PID controllers in literature deployed in the field except for specialized temperature critical systems, most industry-grade HVAC systems do utilize a basic PID controller option through simple hardware implementations. Hard controllers, meanwhile, cover methods based on gain scheduling, nonlinear control, robust control, optimal control and model predictive control. Gain scheduling on its own for temperature control is limited to mostly instrumentation and machinery [21], [22]. Similarly, nonlinear control research focuses on its applications for chemical processes, reactors and engines [23]–[27].

While the research surrounding hard and classical control lacks any focus on thermal comfort controlling the temperature, soft control that is based on either fuzzy logic [28], [29] or neural networks [30]–[34] does delve into the human aspect. Methods used under the soft control category do training under previous data approaching the problem as a supervised classification or regression task [4]. Hybrid

controllers that combine soft and hard control methods also exist, but the scope of the research for such methods does not extend to the thermal comfort case.

When focusing on the research done for thermal comfort, RL control is a prominent control method which does not fit under the three main categories mentioned in the standard sense. Compared to other machine learning methods applied for thermal comfort, RL does learning in context [4].

Earlier RL research for thermal comfort used methods such as SARSA and Q-Learning [35]–[38]. These two methods, despite being outdated and outperformed, are still used commonly as a baseline for comparing newer algorithms. More recently, due to the complexity of thermal building systems due to various factors that influence the environment's state, deep learning has been a popular choice to deal with the complexity. Deep RL approaches for thermal comfort utilize algorithms containing a deep neural network at some stage (most commonly for the environment observation encoding) of the reinforcement learning loop. Such algorithms including deep Q-learning and deep DPG [5], [6] have been shown to perform well in optimizing for thermal comfort, with thorough investigation of their performance for variations in the applied environment, controlled variables that affect thermal conditions, and reward function [7], [39]–[45]. The main focus of most deep RL research for thermal comfort aims to optimize thermal comfort and energy consumption simultaneously, striking a balance according to some defined objective.

Another way to approach classifying RL research for thermal comfort is in terms of their handling of the thermal comfort index. The distinction between group-based comfort models and personal comfort models constitutes the base of this research. A review of the current literature reveals that a majority of methodologies are group-based comfort models, estimating the group's thermal preference, even though personal comfort models dominantly outperform them [4]. When narrowing down this review to just reinforcement learning research rather than all machine learning research, the dominance of using group-based comfort metrics is even more significant. To do human-in-the-loop learning, Cicirelli et al. produce separately and then combine environmental and human rewards when training a deep q-learning agent for controlling the HVAC and windows [46]. Liang et al., on the other hand, approach capturing individual preferences by simulating thermal comfort as a multi-agent problem [8], [47].

Overall, a personal RL comfort model has been chosen to be focused on due to its high potential with a lack of extensive analysis and research surrounding it. Unlike other research, the model proposed in this thesis shall capture and operate solely based on individual preference.

3

Methodology

This chapter explains the system architecture proposed for interactive RL for adaptive thermal comfort. This architecture offers two types of approaches: utilizing human feedback or utilizing PMV for thermal comfort. Each section explores the proposed model by describing the function of and rationale behind a block in the system architecture.

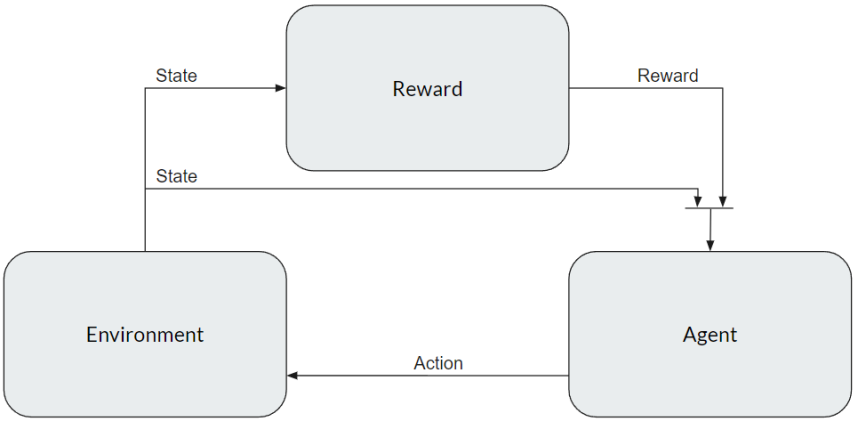

Figure 3.1: Classic interactive RL system overview block diagram

As seen in figure 3.1 the system consists of the environment, agent, and reward blocks. As a general overview, the environment block simulates the room environment in terms of the HVAC system and the building's thermal flow. The agent block determines and sets the temperature setpoint according to the RL model to optimize thermal comfort based on the observed state. Finally, the reward block provides an appropriate reward for the received state. The provided reward can be the *PMV reward* or the *human reward*. This block is designed in a way that allows it to be replaced by actual human interaction when deployed in real-life scenarios when using the *human reward*.

3.1. Environment Block

The environment block is responsible for simulating the thermal conditions of a space according to a provided action (the temperature setpoint) and the previous state. The environment then provides the thermal state of the space to the other blocks after simulating for the timestep, a set amount of time, with the given action.

The thermal simulation of the space is handled by EnergyPlus. EnergyPlus [48] is a whole building

energy simulation program that is used to model energy consumption for heating, cooling, ventilation, lighting and plug, and process loads. It provides a detailed simulation of the thermal dynamics within buildings, allowing for precise analysis and optimization of HVAC systems and energy usage.

EnergyPlus offers a comprehensive suite of tools for modelling building energy performance. It integrates advanced heat balance algorithms and considers various factors such as weather data, building geometry, materials, and occupancy schedules. The program can simulate complex interactions between different building components and systems, providing detailed energy consumption and thermal distribution outputs. Integration of EnergyPlus is valuable as it allows for a far more sophisticated simulation of the thermal conditions of a space, matching the industry standard.

Key features of EnergyPlus include:

- Thermal Modeling: EnergyPlus uses a heat balance method to simulate the thermal behaviour of buildings, accounting for conduction, convection, and radiation heat transfer.
- HVAC Simulation: The software models various HVAC systems, including conventional and advanced configurations, allowing for detailed system performance analysis.
- Flexible Customization: Users can customize their simulations with detailed input data and control strategies, making them suitable for a variety of research and design applications.
- Weather Data Integration: EnergyPlus can incorporate detailed weather data, enabling accurate simulations of building performance under different climatic conditions.

The EnergyPlus Python API facilitates interaction with EnergyPlus through Python scripts, allowing for more flexible and automated control over simulations. This API enables users to perform tasks such as modifying input parameters, running simulations, and extracting output data through Python programming.

EnergyPlus API provides 4 types of API to interact with the environment:

- State API: enables a client to create and manage state instances for using EnergyPlus API methods. Nearly all EnergyPlus API methods require a state object to be passed in, and when callbacks are made, the current state is passed as the only argument.
- Functional API: enables accessing structures and functionality inside EnergyPlus from an outside client. It is just an organizational class that provides access to nested functional classes through member functions.
- Runtime API: enables a client to hook into EnergyPlus at runtime and sense/actuate data in a running simulation through callback functions. Inside the callback function, the client can get sensor values and set actuator values using the DataTransfer API methods, and also look up values and perform calculations using EnergyPlus internal methods via the Functional API methods.
- DataTransfer API: enables data transfer between EnergyPlus and a client. Output variables and meters are treated as “sensor” data while some specific variables in EnergyPlus are controllable as actuators. There are also some static data members exposed as “internal” variables.

In this thesis, a custom gymnasium (gym) environment that uses the EnergyPlus Python API is used for the environmental simulation, due to the compatibility of the gymnasium as a single agent environment with multiple RL algorithm libraries including Stable-Baselines3. The repository by Galatoud [49] has been used as a starting point for the gym environment that utilizes EnergyPlus.

The EnergyPlus Runner which handles the data exchange is implemented heavily utilizing the Runtime API to run an instance of EnergyPlus for a given space with a provided weather file and the DataTransfer API to interact with this instance. Both APIs make use of the State and Functional APIs as they are built on top of them. The instance of EnergyPlus is run in a separate thread. Actions going into the simulation and observations coming out of the simulation are passed through the action and observation queues of size one respectively as only one timestep is processed at a time. EnergyPlus allows for getting the values of any number of defined variables or meters at each timestep. These variables are defined by their zone identifiers, node identifiers and variable names. Any number of actuators can be set at a timestep similarly. Both of these operations are done through the DataTransfer API by initializing an exchange channel and calling the set or get function for the exchange with the unique identifier tuple.

EnergyPlus allows for 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, or 60 timesteps per hour. At each time step, an EnergyPlus episode (not to be confused with RL episodes)is completed, for which the output, info and error files are written. Four timesteps of 15 minutes have been chosen for our purposes as too long timesteps reduce the accuracy of control and simulation results while too short timesteps unnecessarily increase simulation runtime.

The EnergyPlus Runner is then wrapped with a gymnasium class to ensure compatibility with the RL training setup. During initialization, the gym takes in input the environment configuration which defines the verbosity of the environment and the EnergyPlus episode output path. The path to the train or test weather file as well as the reward choice between PMV or human is also indicated here. Reset and Step functions are the two main functions that handle the interface with the RL algorithm. Inside the Reset function, the initial temperature setpoint for the new episode is selected through random sampling of the action space and set. This allows for needed variation during the training as consecutive days can have the same or too similar weather file data. Then the observation, PMV and reward history are cleared for a new episode and the new action and observation Queues are created. In the case that the simulation has stopped at the end of the last episode, the simulation is also restarted. In the step function, the timestep is incremented and the simulation is checked to make sure it is accessible. If the simulation is stopped, this is identified by the timeout mechanism or a None value in the observation queue. In those cases, the simulation episode is stopped early with the last observation passed for the current timestep once. If the simulation is running without any issues, then the episode is stopped at the determined episode length. The PMV is calculated here as well but only for keeping history. At the end of an uninterrupted episode (this is always the case if the total number of timesteps simulated is divisible by the episode length) the history containing the observations, rewards and PMVs at each timestep of the episode is stored, being appended to the pickle file containing histories until then.

Observation	Range	Unit
Site Outdoor Air Drybulb Temperature	[-40,45]	°C
Zone Mean Air Temperature	[0,45]	°C
Zone Air Relative Humidity	[0,100]	%
Zone Mean Radiant Temperature	[0,45]	°C

Table 3.1: Environment Observation Space

The observation space consists of 4 different observations with their ranges and units seen in table 3.1. 11 variables have been identified to be accessible and directly related to the thermal conditions of a zone in EnergyPlus. Of these variables, the heating and cooling setpoints fall under the action category while the People Occupant Count is only a static variable set at the beginning of the simulation with no measured impact dynamically during the simulation. Windows Total Transmitted Solar Radiation Rate, while informative and relevant, is not available for the chosen experiment space. These variables have therefore been excluded from the observation space provided by the custom gym. Operative Temperature, Electric Equipment Total Heating Rate, and Zone Infiltration Air Change Rate have been considered to be included in the observation space. However, the increase in dimensionality has been shown to hinder the precision of the trained model rather than improve it. The unnecessary expansion of the observation space has also shown repeated catastrophic forgetting during training in preliminary experiments. In light of these experiments, a reduction in the observation space to the sole parameters of PMV has been pursued, lowering the number of variables from 11 to 4. EnergyPlus offers the tracking of many other variables, but as this research focuses on the thermal comfort aspect, all variables related just to energy consumption have been intentionally not considered.

3.2. Agent Block

The agent block is responsible for determining an action, the temperature setpoint, for a provided state (thermal observations). For a single setpoint HVAC setup this action controls the one setpoint while for a dual setpoint setup like the one being used for this study, both the heating and the cooling setpoint need to be controlled. To significantly reduce the dimensionality of the problem, a design choice has been made to have the two setpoints offset apart from one another. This design choice, on top of the dimensionality benefits, mimics the one setpoint setup while still reducing the number of switches

between heating and cooling. The offset in our case has been chosen as 0.5 as keeping inside a temperature range of 0.5 is enough to satisfy the acceptable PMV ranges. Some preliminary testing with larger offsets has shown that the heating setpoint stays as the active temperature target the majority of the time. This reinforces the validity of using a single action to determine both setpoints.

Variable	Range
Normalized Action	[-1,1]
Zone Thermostat Heating Setpoint Temperature	[18,36]
Zone Thermostat Cooling Setpoint Temperature	[18.5,36.5]

Table 3.2: Action Space

The temperature values that range from 18 to 36 are normalized to a range of -1 to 1. As the PPO implementation used relies on a Gaussian distribution (initially centred at 0 with std 1) for continuous actions, making this normalization avoids harming the learning process.

3.3. Reward Block

This reward block is responsible for providing a reward for the current state of the environment at each timestep. We implemented two possible types of reward, PMV reward and Human reward. These two rewards allow for comparing the more informative PMV ground truth case to the less informative but applicable in real-life interactive Human reward. The PMV reward is taken as the true metric of thermal comfort. Assuming that the clothing insulation and metabolic rate for the PMV calculation are accurately provided, the PMV represents the ground value for thermal comfort. The logic behind this decision follows from the fact that the Human reward uses an underlying PMV value which is hidden from the agent block. As such, using the PMV reward models the case where the agent has access to the true thermal preference of the human while the human reward models the case where the agent has to interpret the thermal preference from just the human feedback. The state received for the reward calculation is a subset of the full state for both reward cases.

The PMV reward is defined as:

$$\text{Reward} = \begin{cases} -|\text{PMV}(\text{state})|, & \text{if not NaN} \\ -4, & \text{otherwise} \end{cases} \tag{3.1}$$

As PMV values range from -3.5 to 3.5, a linear relationship exists between the reward values and the optimal temperature as long as the thermal conditions are not extreme. In the extreme thermal discomfort case, PMV calculation produces NaN values which are handled by a -4 reward. Since 0 PMV represents the ideal state for thermal comfort and being further away from 0 represents thermal discomfort, taking the absolute value of PMV conveys the distance to 0 and hence the distance to the state of ideal thermal comfort. Finally, this value is multiplied by -1 to make it an appropriate reward as being further away from the ideal state needs to be punished more. Returning to the handling of the NaN case, a -4 reward is a worse reward than any not NaN PMV scenario as the minimum reward received with a not NaN PMV is -3.5. Even though a sense of direction can be lost with a fixed reward for the NaN region, the choice of a narrower action space forces the RL algorithm to experience some states in the not NaN PMV region, negating the effect.

The human reward option, as the name suggests can simulate a human with varying thermal preferences. A human model has an underlying probability distribution based on PMV which indicates the individual's probability of interacting with the thermostat. While the PMV formula and the environment variables such as humidity, air temperature and radiant temperature are the same for all individuals in the room, the clothing insulation and metabolic rate vary.

Picking a probability distribution for the probability of complaint is less than straightforward as supported by research surrounding PMV and adaptive thermal comfort covered under chapter 2. We propose a five-parameter approach that attempts to better align with modern adaptive thermal comfort models by allowing for more flexibility and making the study more future-proof. The probability distribution itself is a sum of two exponentials with moving parameters as represented in the following formula:

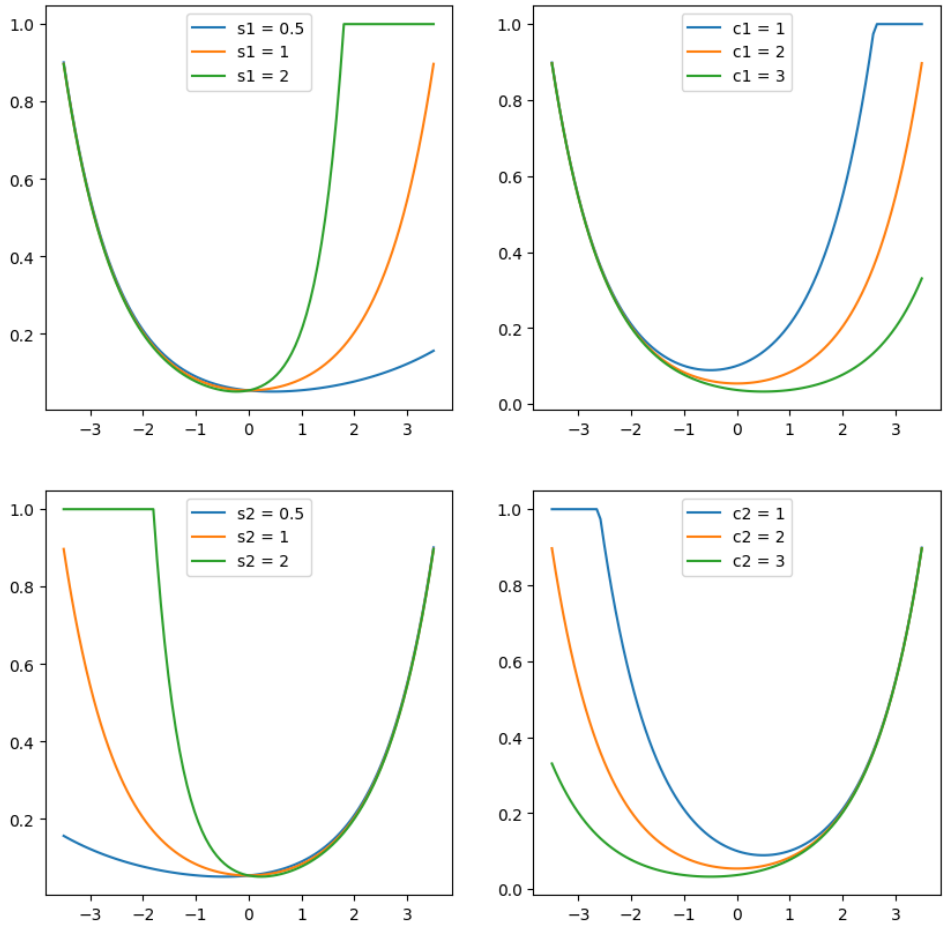

Figure 3.2: Probability Distribution over PMV with Varying s and c Parameters (default values: k, s_1, c_1, s_2, c_2 = 0.2, 1, 2, 1, 2)

$$P(x_{pmv}) = min(k(\exp(s_1 \times x_{pmv} - c_1) + \exp(-s_2 \times x_{pmv} - c_2)), 1) \tag{3.2}$$

The calculated PMV value from the environment and individual variables are then input into this probability function in place of x_{pmv}. The s_1 and s_2 values control the slopes on the two sides of the function, which corresponds to an individual's tolerance to too-hot or too-cold sensation. The c_1 and c_2 values on the other hand control the shift of the centres of the two exponentials, corresponding to how much a person likes or complies with hotter or colder sensations. The four parameters in unison allow for modelling symmetric and non-symmetric preferences towards hot and cold sensations, squeezing, expanding, shifting and skewing the shape of the probability distribution if needed.

Any scaling in the probability axis can be explained with a scalar multiplication of the same function, where the scalar multiplier represents the general willingness of the individual to complain. Hence, an additional variable, k, has been introduced to multiply with the rest of the function as suggested to adjust the willingness of the individual to complain. The human block returns a -1 reward if a complaint happens for that timestep and a 0 reward otherwise.

In this research, only symmetric probability distributions centred on the 0 PMV value (i.e. s_1 is equal to s_2 and c_1 is equal to c_2) have been considered. Choices of skewed or uncentered probability distributions introduce a change in the optimal PMV value away from 0. While this has been intentionally made possible to allow for modelling more unique and nuanced thermal preferences in the probability distribution defined in equation 3.2, in this use case, non-zero centred probability distributions make a sound comparison to the PMV case impossible.

A zeroed-out version of the probability distribution has also been used defined by the equation:

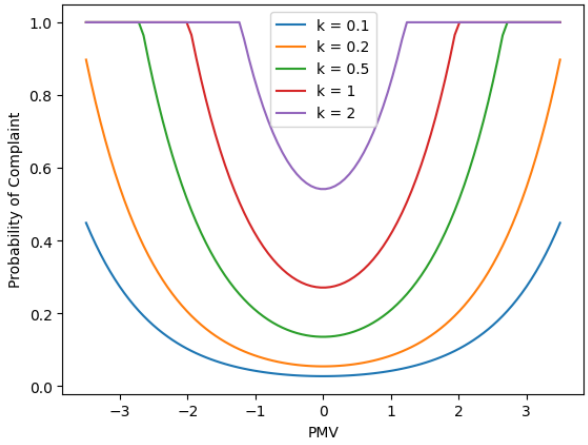

Figure 3.3: Probability Distribution over PMV with Varying k Parameters (default values: s_1, c_1, s_2, c_2 = 1, 2, 1, 2)

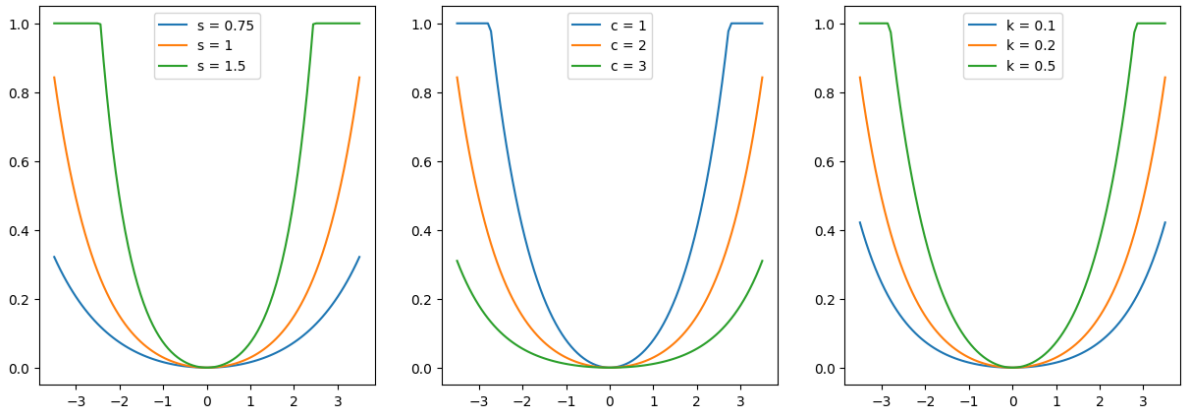

Figure 3.4: Zeroed Probability Distribution over PMV with Varying s and c Parameters (default values: k, s, c = 0.2, 1, 2)

$$P(x_{pmv}) = min(k(\exp(s \times x_{pmv} - c) + \exp(-s \times x_{pmv} - c) - 2 \times \exp(-c)), 1) \tag{3.3}$$

This equation is only applicable in the symmetric case where the c values are equal and s values are equal. This Probability distribution assumes that no complaint is made in the ideal 0 PMV case. This distribution helps with experimenting in a scenario with fewer unnecessary complaints.

As seen in figure 3.4, by zeroing out the distribution such that the probability at PMV = 0 is 0, variations in the s, c and k parameters result in almost identical distributions. Therefore only variations in the k parameter have been experimented with to avoid redundancy.

The internal calculations including the underlying PMV model are not directly reachable by the agent or environment to stay consistent with the assumption that the RL model does not have any information about the human except from what is inferred from the human's interaction.

4

Results

This chapter covers the experimental setup used and the results of the training and testing. The PMV reward case results and the human reward case results are given separately under the training and test result sections. All training uses the seed value 42 and all test runs use deterministic policy True which chooses the most probable action at each timestep for reproducibility.

4.1. Experimental Setup

The model training is run on the Energyplus modelling of an existing space called BRIGHT lab in Amsterdam [50]. This single-room space has an HVAC capable of both heating and cooling and the modelling can be edited to allow for both ideal and non-ideal thermal element setups. In our experiments, a close-to-ideal setup has been used which allows for a delta of 10 degrees in 2-3 timesteps.

The Stable-Baselines3 implementation for PPO is used to train the RL agent that controls the thermostat setpoint [51]. While this library supplies a variety of other RL algorithms, PPO was specifically chosen as it works well with continuous action spaces and handles complex observation spaces as well.

The PMV reward case and the human reward case are trained separately and are then tested. The training data consists of a weather file of Amsterdam, which is the original location of the BRIGHT lab. As for test data, a weather file of Groningen is used. Both weather files are a year long.

The episode length has been chosen as 96 timesteps (1 whole day). A day represents the shortest period that is repeating a similar scenario. The whole day is used rather than just the working hours as, even though a working space might only be occupied inside the working hours, the rest of the day is still valid and valuable training data. This choice is expected to not jeopardize the thermal conditions of the space during working hours but would also introduce the model to a wider range of scenarios potentially improving its generalizability.

Both the training and testing are run for a full year (365 days) and therefore the full weather file each. This means a total of 35040 timesteps per training and per test run.

In the Human reward case, the human has been initialized to have a metabolic rate of 1.4 and a clothing insulation of 1.1. As a metabolic rate of 1.0 represents the resting metabolic rate, a metabolic rate of 1.4 is chosen to model a human doing a very minor activity such as typing with an occasional walk around the office space. This of course carries the assumption that this is an average human with an average basal metabolism as defined by ASHRAE. Similarly, the clothing insulation value chosen also carries an assumption that the human we are modelling wears thicker-than-average clothing with added insulation from the chair being sat on year-round. This results in the optimal temperature for the modelled human to linger around 20-22°C.

4.2. Training Results

4.2.1. PMV Reward

The PPO training configuration for the PMV reward case uses the hyperparameter settings seen in table 4.1.

Hyperparameter	Value
Gamma	0.95
Learning Rate	1e-5
Entropy Coefficient	0.1
Training Batch Size	96
SGD Minibatch Size	96
Number of Epochs	10

Table 4.1: PMV Reward PPO Configuration Hyperparameters

These values have been chosen after hyperparameter tuning to find a stable learning curve. Higher learning rates caused unstable reward and loss curves while the PMV reward was informative enough that 10 epochs proved to be sufficient. The entropy coefficient helps with avoiding catastrophic forgetting with the added randomness stopping extreme convergence.

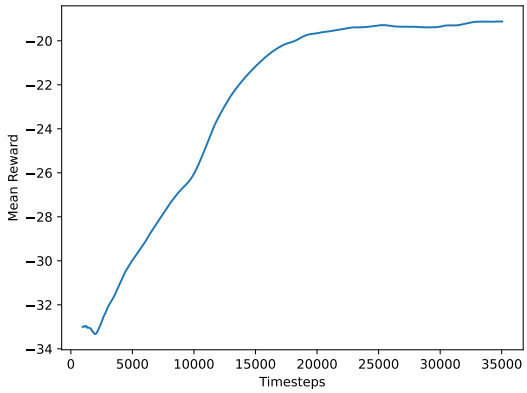

Figure 4.1: PMV Reward Case Training Mean Rewards over Timesteps

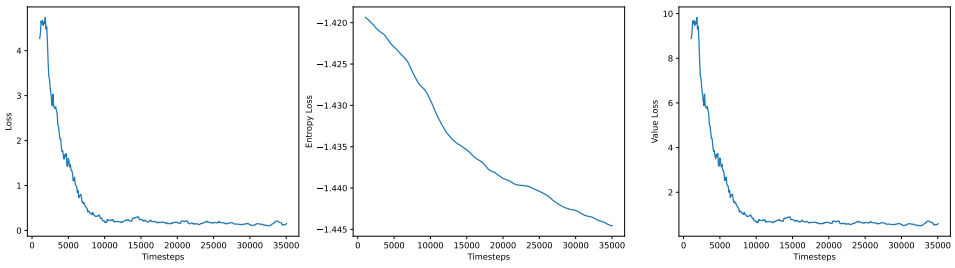

Figure 4.2: PMV Reward Case Training Losses over Timesteps

Figures 4.2 and 4.1 show the mean episodic rewards and losses over the training duration using the PMV reward. Both the losses and the rewards show a smooth convergence over time. The mean reward is converging to above -20 which means a mean absolute PMV below 0.2 at each timestep.

4.2.2. Human Reward

The PPO training configuration for the PMV reward case uses the hyperparameter settings seen in table 4.2.

Hyperparameter	Value
Gamma	0.95
Learning Rate	1e-5
Entropy Coefficient	0.1
Training Batch Size	96
SGD Minibatch Size	96
Number of Epochs	100

Table 4.2: PMV Reward PPO Configuration Hyperparameters

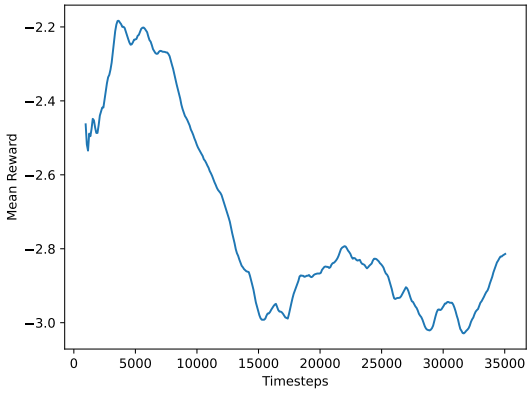

Figure 4.3: Human Reward Case Training Mean Rewards over Timesteps

Compared to the PMV reward, the human reward exhibited less tendency to exhibit catastrophic forgetting due to the reward's inherent stochasticity. Combined with how the human reward is less informative, the use of higher epochs proved to have slightly better results. It is worth noting that the improvement in the learned policy due to different epochs is observed to be marginal and the final performance relies heavily on the stopping point of the learning. The training curves with different epochs can be seen in the Appendix in section B.1.3.

Figures 4.4 and 4.3 show the mean episodic rewards and losses over the training duration using the Human reward. The mean reward using the human reward fluctuates quite a lot, showing only a slight convergence around the [-3.0, -2.8] range. This suggests that a complaint happens 2 to 3 times per day, or a mean of 0.012 times every hour. The value loss also fluctuates which follows from how the reward itself is stochastic. The entropy loss shows a stable decreasing trend which makes the total loss carry a similar downward trend over the trend despite the fluctuations.

A value of 0.1 has been used for the k value in the Human reward case. To investigate if the lack of complaints due to the low probability of complaints is causing an increase in the fluctuations during the learning, training has been done with higher k values. The training plots for these experiments can be

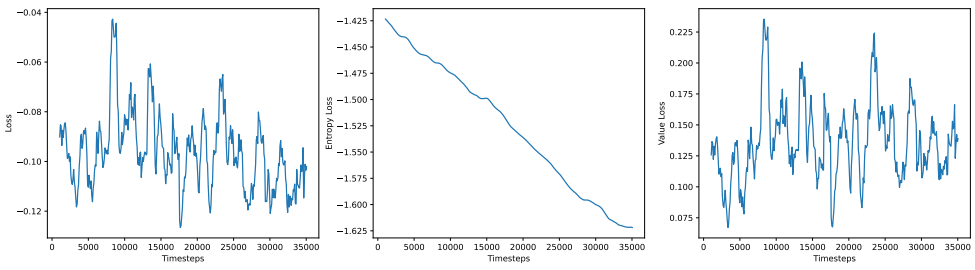

Figure 4.4: Human Reward Case Training Losses over Timesteps

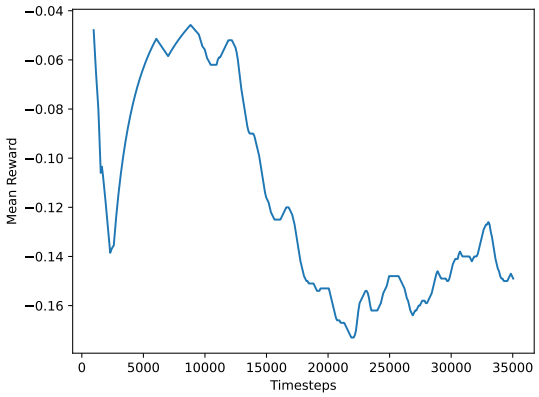

Figure 4.5: Human Reward Case with Zeroed Probability Training Mean Rewards over Timesteps

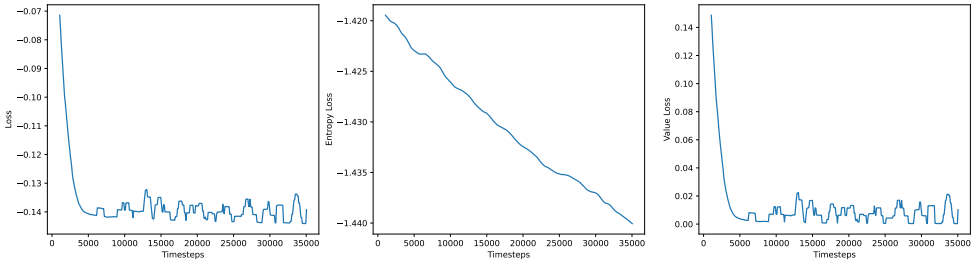

Figure 4.6: Human Reward Case with Zeroed Probability Training Losses over Timesteps

found in the Appendix under section B.1.1. Seeing as higher k values resulted in similar training trends with just a lower mean reward proportional to the increase in the probability of complaint, the k value has been kept as 0.1.

A separate experiment has been done to investigate if the suboptimal mean reward convergence is caused by false complaints that happen even in close to optimal thermal conditions. In these experiments, the zeroed probability distributions have been used, making the human never complain in the optimal 0 PMV case and less likely the complain around that region.

Figures 4.6 and 4.5 show the mean episodic rewards and losses over the training duration using the Human reward with a zeroed probability distribution for complaint. The mean reward in this case still fluctuates quite a lot, showing a slight convergence around the [-0.15, -0.13] range. The value loss still also fluctuates a bit but a convergence trend is quite more visible. The entropy loss shows a stable decreasing trend similar to the non-zeroed-out probability case.

A value of 0.1 has been used for the k value in the zeroed-out probability distribution Human reward case as well. To investigate if the lack of complaints due to the low probability of complaints is causing an increase in the fluctuations during the learning, training has been done with higher k values parallel to the classic Human reward case. The training plots for these experiments can be found in the Appendix under section B.1.2. Seeing as higher k values resulted in similar training trends with just a lower mean reward proportional to the increase in the probability of complaint or even a loss of convergence in the value loss in too high k values, the k value has been kept as 0.1.

4.3. Test Results

4.3.1. PMV Reward

Figure 4.7 shows how the agent using the learned policy from the PMV reward sets the temperature setpoint over a day and how the mean air temperature inside the space follows it. The mean temper-

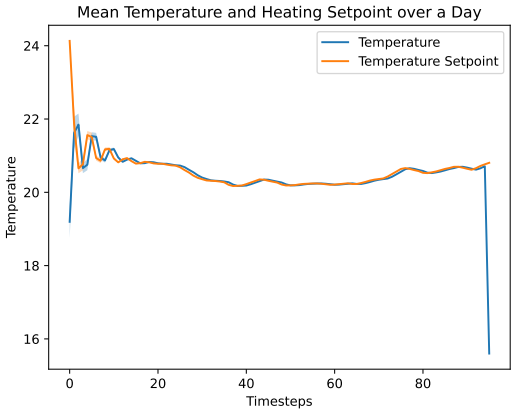

Figure 4.7: PMV Reward Case Test Mean Setpoint and Temperature over a Day

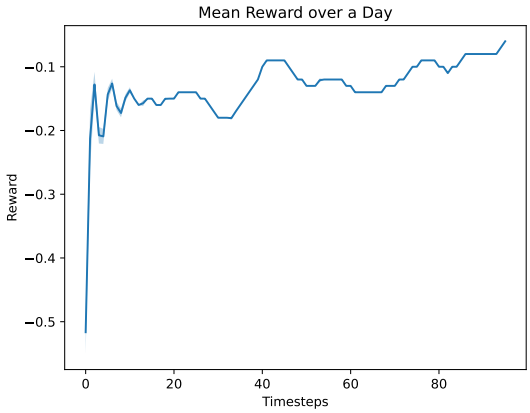

Figure 4.8: PMV Reward Case Test Mean Reward over a Day

ature setpoint is set higher during the initial timesteps as the mean starting temperature is lower than the targeted optimal temperature. After stabilizing the temperature around the targeted range, the temperature setpoint is kept quite stable making only small adjustments. There is only a small deviation in the temperature setpoints and the temperature itself over the whole year, which is caused by the randomized initial temperature of each day. The policy converges to the same temperatures quickly at the start and then follows a constant trend afterwards. The mean temperature is kept around 21°C.

Looking at Figure 4.8, the policy learned by the agent with the PMV reward has a mean reward higher than -0.2 after. As the reward is just the negative of the absolute PMV in the PMV reward case, this implies that the PMV values stay between the [-0.2, 0.2] range. The mean reward is quite stable and does not have much variation between episodes except for the one caused by the initial temperature.

4.3.2. Human Reward

Figure 4.9 shows how the agent using the learned policy from the Human reward sets the temperature setpoint over a day and how the mean air temperature inside the space follows it. The mean temperature setpoint is set higher during the initial timesteps as the mean starting temperature is lower than the targeted temperature by the policy. After stabilizing the temperature around the targeted range, the temperature setpoint is kept quite stable making only small adjustments. There is only a small deviation in the temperature setpoints and the temperature itself over the whole year, seemingly only caused by the randomized initial temperature of each day. The policy converges to the same temperatures quickly at the start and then follows a constant trend afterwards. The policy sets and keeps the temperature around 25°C.

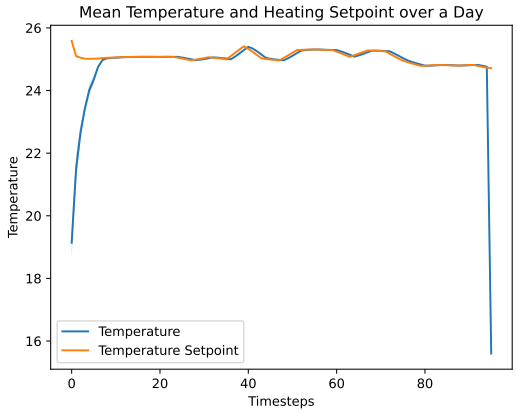

Figure 4.9: Human Reward Case Test Mean Setpoint and Temperature over a Day

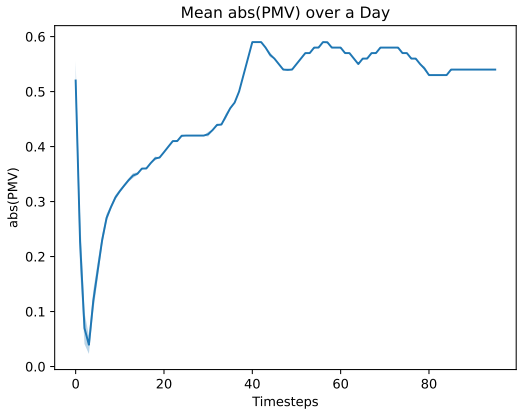

Figure 4.10: Human Reward Case Test Mean Absolute PMV over a Day

Looking at Figure 4.10, the policy learned by the agent with the Human reward has a mean absolute PMV lower than 0.6 so the PMV values stay between the [-0.6, 0.6] range. The PMV value converges to a suboptimal that does not quite accurately represent the Human's preference. Nevertheless, the policy stays near the [-0.5, 0.5] range which is the accepted range for satisfaction. The PMV does not show much variation between episodes except for the start of the day.

The policies learned from the Human reward case with zeroed probability distribution are also tested in an unseen year as they have shown promise in terms of training stability. Figure 4.11 shows how the agent using the learned policy from the zeroed probability distribution Human reward sets the temperature setpoint over a day and how the mean air temperature inside the space follows it. The mean temperature setpoint is set higher during the initial timesteps as the mean starting temperature is lower than what seems to be the targeted temperature by the policy. After stabilizing the temperature around the targeted range, the temperature setpoint is kept quite stable with no adjustments afterwards. There is even a smaller deviation in the temperature setpoints over the whole year, seemingly only caused by the randomized initial temperature of each day. The policy converges to the same temperature at the start slower than the previous two cases as the target temperature is quite high. The policy sets the temperature around 28°C while the temperature itself spends some time around the [24,28] while reaching that setpoint.

Looking at Figure 4.12, the policy learned by the agent with the zeroed probability distribution Human reward has a mean absolute PMV lower than 1 so the PMV values stay between the [-1, 1] range. The PMV value converges to a suboptimal that does not represent the human's preference but the initial one-third of the day spent on reaching the temperature setpoint does stay inside the accepted PMV

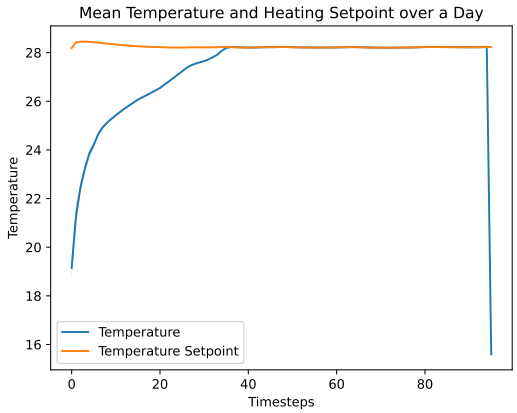

Figure 4.11: Human Reward Case with Zeroed Probability Test Mean Setpoint and Temperature over a Day

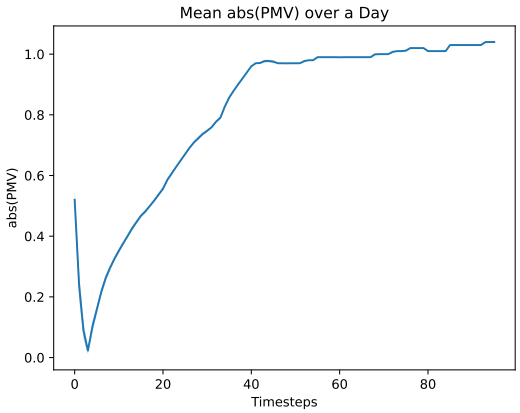

Figure 4.12: Human Reward Case with Zeroed Probability Test Mean Absolute PMV over a Day

margin. The PMV does not show much variation between episodes except for the start of the day.

5

Discussion

The optimization using the PMV reward for thermal comfort assuming that the PMV accurately reflects the indoor thermal environment's alignment with occupant comfort levels, successfully sets the thermostat setpoint in a way that minimizes the absolute PMV. By utilizing the PMV directly, the PMV reward function successfully ensures the system prioritizes maintaining optimal thermal conditions, which is crucial in HVAC control. This shows the model's capability to adapt and maintain environments that align well with established thermal comfort standards given that the reward is informative enough. The PMV reward model consistently maintains PMV values within a tighter range of -0.2 to 0.2, which surpasses the commonly accepted range of -0.5 to 0.5. This narrower band signifies a high level of control over the thermal environment, improving the occupants' overall comfort experience. Such precision reflects the model's effectiveness in fine-tuning the system to avoid PMV values that, while technically acceptable, could still be perceived as uncomfortable by users.

The human reward model's performance is not so sensitive to the selection of hyperparameters but is sensitive to the stopping point, which introduces instability in the acquired policy after training. The model's instability is mainly caused by the stochastic nature of the training process. Stochastic environments inherently introduce variability in outcomes, which complicates the training and optimization of the reward model. This sensitivity suggests that a series of misleading experiences during the training before the policy is acquired can lead to significant fluctuations in outcomes. The reliance on stable training episodes limits the model's robustness and generalizability if deployed in the real world, requiring further refinement to achieve consistent performance.

Another source of instability stems from the relatively low informativeness of the Human reward, which hampers sample efficiency. The -1 reward signal lacks any direction or scale of how bad the current state of the environment is at that timestep. The complaint can also be made much later for a thermal state that is far from optimal even though the probability of complaint is high for each timestep. When rewards are sparse or inadequately informative, the model struggles to learn effective control strategies, leading to unstable or slow convergence. Enhancing the richness of the feedback would improve the system's ability to stabilize and more quickly converge on optimal policies.

Despite the instability and suboptimal policy learned by the Human reward case, the PMV values remain close to the accepted comfort range of [-0.5, 0.5]. This indicates that while the model may not always achieve the optimal PMV target, it still ensures conditions that may be considered comfortable by a significant amount of people. The fact that the system maintains comfort despite instabilities highlights its overall effectiveness, though there is room for improvement in optimizing thermal control further.

Comparing the test results with higher and lower probability of complaint in the Human reward case with both zeroed and non-zeroed probability distributions, the increase in the probability of user complaints does not necessarily result in improved control. Having less probability of complaint even produces a better learning curve in the zeroed probability distribution Human reward case, although the resulting policies are not better with less complaint probability in the test cases.

Both Reward cases demonstrate a reasonable ability to handle delayed rewards, which is a crucial aspect of managing HVAC systems where feedback on actions may not be immediate. Policy learned in both cases adjusts the away from the unwanted temperatures immediately at the beginning of the day. This is a nice surprise as it shows that a history or frame stacking is not necessary and that the policy can be activated at any time to control the temperature setpoint immediately. The system's ability to adjust and learn from these delayed responses suggests that it can accommodate the inherent time lag in real-world thermal environments, enhancing its practical applicability.

The limitations of this study reveal possible directions for future work. A possible way to handle the issue of sample efficiency is to increase the information available in the reward. Keeping with the theme of producing rewards from thermostat interaction as the complaint, expanding the model to increase the number of simulated humans can make the reward more informative. An increase in the number of complaints per timestep introduces a sense of distance to the optimal conditions. Also, the reward becomes more stable as the number of people increases.

The use of EnergyPlus for simulating the thermal environment requires daily simulations, which include non-working hours. This forces the model to learn a wider range of cases. While this might help increase the generalizability of the model, it might not be helpful coming to the real-life case as the model must account for periods where thermal control is less critical. A more targeted approach, focusing on working hours, could streamline the training process and lead to more efficient and real-life applicable model development.

The weather data used in the simulations has been averaged, which reduces the variation in environmental conditions encountered during training. This can be observed from how the temperature does not vary much during the day in the test data in a day over the year. This lack of diversity in the training data could limit the model's ability to generalize to real-world scenarios, where weather conditions fluctuate significantly. Incorporating more dynamic and varied weather data would likely improve the robustness and adaptability of the model.

Testing the model with uncentered and skewed probability distributions could provide valuable insights into its behaviour under different thermal preference conditions. However, this approach requires a clearer definition of ground values to ensure accurate comparisons and evaluations which is why it was avoided in this study. The proposed model offers a good amount of flexibility in defining how human thermal comfort complaint is stochastically simulated. Working with more adaptive definitions of thermal comfort, the proposed model makes it possible to further research into areas of thermal control where other physiological and psychological factors not captured by PMV are not overlooked.

Future research could explore scenarios involving multiple humans with varying thermal preferences, which would more closely mirror real-world conditions. However, this introduces challenges related to fairness, as the system must balance competing preferences equitably. Investigating methods for fair allocation of thermal comfort could lead to more inclusive and adaptable HVAC control strategies.

6

Conclusion

This thesis aimed to answer how RL could be utilized to incorporate individual preferences while controlling the HVAC for thermal comfort. It has been shown that human feedback can be collected through human interactions with the HVAC, specifically through setting the thermostat setpoint. It has also been shown that this interaction can be exploited to gather individuals' thermal preferences. Models that depend on human interactions with the thermostat as a form of complaint have been shown to successfully infer the underlying thermal preference of the occupants to some extent. This task which was handled with PMV estimations before can now be estimated with human interaction collection. This has the benefit of dynamically adjusting to the preferences of the occupants without needing to wait on a new manual PMV estimation. Since the system architecture has been developed with real-life deployment in mind from the start, the proposed system can directly be used to replace the current fixed thermal control systems.

This thesis has also produced a system architecture which is well-integrated with EnergyPlus, modifiable in terms of the RL algorithm and flexible in the modelling of human simulated. This implies that without any changes to the proposed system architecture, simulated humans can be tested with different RL algorithms, simulated humans can be adjusted to be more in line with adaptive thermal comfort models that consider physiological and psychological differences, and the full spectrum of variables and actuators available in the Energyplus suite can be accessed according to any need. The proposed method of stochastically simulating human complaints with an underlying comfort metric is also usable in other architectures and contexts, making RL thermal control personalized without the need for real-life data. This thesis therefore paves the way for individual preference extraction for thermal comfort, taking the first step in showing that thermal preference inference is possible. The use of RL can substantially improve the current capabilities of HVAC systems beyond scheduled control, allowing for rapid and automatic adaptation to occupant and environmental changes and removing the dependency on planning analysis and estimations.

References

[1] *ANSI/ASHRAE Standard 55-2013: Thermal Environmental Conditions for Human Occupancy* (ASHRAE standard). ASHRAE, 2013. [Online]. Available: https://books.google.nl/books?id=AGxFjwEACAAJ.

[2] A. M. Bueno, A. A. de Paula Xavier, and E. E. Broday, "Evaluating the connection between thermal comfort and productivity in buildings: A systematic literature review," *Buildings*, vol. 11, no. 6, p. 244, 2021.

[3] A. Afram and F. Janabi-Sharifi, "Theory and applications of hvac control systems – a review of model predictive control (mpc)," *Building and Environment*, vol. 72, pp. 343–355, 2014, ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2013.11.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132313003363.

[4] Z. Q. Fard, Z. S. Zomorodian, and S. S. Korsavi, "Application of machine learning in thermal comfort studies: A review of methods, performance and challenges," *Energy and Buildings*, vol. 256, p. 111 771, 2022.

[5] Y. R. Yoon and H. J. Moon, "Performance based thermal comfort control (ptcc) using deep reinforcement learning for space cooling," *Energy and Buildings*, vol. 203, p. 109 420, 2019.

[6] G. Gao, J. Li, and Y. Wen, "Deepcomfort: Energy-efficient thermal comfort control in buildings via reinforcement learning," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8472–8484, 2020.

[7] F. Cicirelli, A. Guerrieri, C. Mastroianni, L. Scarcello, G. Spezzano, and A. Vinci, "Balancing energy consumption and thermal comfort with deep reinforcement learning," in *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*, IEEE, 2021, pp. 1–6.

[8] L. Yu, Z. Xu, T. Zhang, X. Guan, and D. Yue, "Energy-efficient personalized thermal comfort control in office buildings based on multi-agent deep reinforcement learning," *Building and Environment*, vol. 223, p. 109 458, 2022.

[9] H. S. Hens, "Thermal comfort in office buildings: Two case studies commented," *Building and Environment*, vol. 44, no. 7, pp. 1399–1408, 2009.

[10] R. J. de Dear, T. Akimoto, E. A. Arens, *et al.*, "Progress in thermal comfort research over the last twenty years," *Indoor air*, vol. 23, no. 6, pp. 442–461, 2013.

[11] C. Arzate Cruz and T. Igarashi, "A survey on interactive reinforcement learning: Design principles and open challenges," in *Proceedings of the 2020 ACM designing interactive systems conference*, 2020, pp. 1195–1209.

[12] J. Lin, Z. Ma, R. Gomez, K. Nakamura, B. He, and G. Li, "A review on interactive reinforcement learning from human social feedback," *IEEE Access*, vol. 8, pp. 120 757–120 765, 2020.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[15] C. Yu, A. Velu, E. Vinitsky, *et al.*, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[16] L. Engstrom, A. Ilyas, S. Santurkar, *et al.*, "Implementation matters in deep rl: A case study on ppo and trpo," in *International conference on learning representations*, 2019.

[17] H. Asifa and S. Vaishnav, "Particle swarm optimisation algorithm based pid controller," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, 2010, pp. 628–631. DOI: 10.1109/ICETET.2010.145.

[18] J. Bai and X. Zhang, “A new adaptive pi controller and its application in hvac systems,” *Energy Conversion and Management*, vol. 48, no. 4, pp. 1043–1054, 2007, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2006.10.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890406003359.

[19] J. Bai, S. Wang, and X. Zhang, “Development of an adaptive smith predictor-based self-tuning pi controller for an hvac system in a test room,” *Energy and Buildings*, vol. 40, no. 12, pp. 2244–2252, 2008, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2008.07.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778808001527.

[20] M. R. Kulkarni and F. Hong, “Energy optimal control of a residential space-conditioning system based on sensible heat transfer modeling,” *Building and Environment*, vol. 39, no. 1, pp. 31–38, 2004, ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2003.07.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132303001628.

[21] H. Sarhan, “A software-based gain scheduling of pid controller,” *International Journal of Instrumentation and Control System*, vol. 4, pp. 1–10, 2014.

[22] W. L. Luyben, “Fed-batch reactor temperature control using lag compensation and gain scheduling,” *Industrial & engineering chemistry research*, vol. 43, no. 15, pp. 4243–4252, 2004.

[23] B. W. Bequette, “Nonlinear control of chemical processes: A review,” *Industrial & Engineering Chemistry Research*, vol. 30, no. 7, pp. 1391–1413, 1991.

[24] P. Setlur, J. R. Wagner, D. M. Dawson, and E. Marotta, “An advanced engine thermal management system: Nonlinear control and test,” *IEEE/ASME transactions on mechatronics*, vol. 10, no. 2, pp. 210–220, 2005.

[25] M. Soroush and C. Kravaris, “Nonlinear control of a batch polymerization reactor: An experimental study,” *AIChE journal*, vol. 38, no. 9, pp. 1429–1448, 1992.

[26] G. Özkan, Ş. Özen, S. Erdoğan, H. Hapoğlu, and M. Alpbaz, “Nonlinear control of polymerization reactor,” *Computers & Chemical Engineering*, vol. 25, no. 4-6, pp. 757–763, 2001.

[27] S. Salehi and M. Shahrokhi, “Two observer-based nonlinear control approaches for temperature control of a class of continuous stirred tank reactors,” *Chemical Engineering Science*, vol. 63, no. 2, pp. 395–403, 2008.

[28] M. M. Gouda, S. Danaher, and C. P. Underwood, “Thermal comfort based fuzzy logic controller,” *Building Services Engineering Research and Technology*, vol. 22, no. 4, pp. 237–253, 2001. DOI: 10.1177/014362440102200403. eprint: https://doi.org/10.1177/014362440102200403. [Online]. Available: https://doi.org/10.1177/014362440102200403.

[29] R. Z. Homod, K. S. M. Sahari, H. A. Almurib, and F. H. Nagi, “Gradient auto-tuned takagi–sugeno fuzzy forward control of a hvac system using predicted mean vote index,” *Energy and Buildings*, vol. 49, pp. 254–267, 2012, ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2012.02.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778812000904.

[30] Z. Deng and Q. Chen, “Artificial neural network models using thermal sensations and occupants’ behavior for predicting thermal comfort,” *Energy and Buildings*, vol. 174, pp. 587–602, 2018.

[31] J. Liang and R. Du, “Thermal comfort control based on neural network for hvac application,” in *Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005.*, IEEE, 2005, pp. 819–824.

[32] P. Ferreira, A. Ruano, S. Silva, and E. Conceicao, “Neural networks based predictive control for thermal comfort and energy savings in public buildings,” *Energy and buildings*, vol. 55, pp. 238–251, 2012.

[33] T. Chaudhuri, Y. C. Soh, H. Li, and L. Xie, “A feedforward neural network based indoor-climate control framework for thermal comfort and energy saving in buildings,” *Applied energy*, vol. 248, pp. 44–53, 2019.

[34] D. Palladino, I. Nardi, and C. Buratti, “Artificial neural network for the thermal comfort index prediction: Development of a new simplified algorithm,” *Energies*, vol. 13, no. 17, p. 4500, 2020.

[35] Q. Fu, L. Hu, H. Wu, F. Hu, W. Hu, and J. Chen, “A sarsa-based adaptive controller for building energy conservation,” *Journal of Computational Methods in Sciences and Engineering*, vol. 18, no. 2, pp. 329–338, 2018.

[36] M. Han, R. May, X. Zhang, *et al.*, “A novel reinforcement learning method for improving occupant comfort via window opening and closing,” *Sustainable Cities and Society*, vol. 61, p. 102 247, 2020.

[37] J. Brusey, D. Hintea, E. Gaura, and N. Beloe, “Reinforcement learning-based thermal comfort control for vehicle cabins,” *Mechatronics*, vol. 50, pp. 413–421, 2018.

[38] T. An, “Whole-building energy co-simulation for dynamic occupant-based heating and cooling control with rule-based and q-learning algorithms,” in *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, IEEE, 2021, pp. 121–127.

[39] S. Brandi, M. S. Piscitelli, M. Martellacci, and A. Capozzoli, “Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings,” *Energy and Buildings*, vol. 224, p. 110 225, 2020.

[40] Z.-K. Ding, Q.-M. Fu, J.-P. Chen, H.-J. Wu, Y. Lu, and F.-Y. Hu, “Energy-efficient control of thermal comfort in multi-zone residential hvac via reinforcement learning,” *Connection Science*, vol. 34, no. 1, pp. 2364–2394, 2022.

[41] K.-H. Yu, Y.-A. Chen, E. Jaimes, *et al.*, “Optimization of thermal comfort, indoor quality, and energy-saving in campus classroom through deep q learning,” *Case Studies in Thermal Engineering*, vol. 24, p. 100 842, 2021.

[42] Z. Wu, Y. Mu, S. Deng, *et al.*, “Towards comfortable and cost-effective indoor temperature management in smart homes: A deep reinforcement learning method combined with future information,” *Energy and Buildings*, vol. 275, p. 112 491, 2022.

[43] X. Liu and Z. Gou, “Occupant-centric hvac and window control: A reinforcement learning model for enhancing indoor thermal comfort and energy efficiency,” *Building and Environment*, vol. 250, p. 111 197, 2024.

[44] Y. An and C. Chen, “Energy-efficient control of indoor pm2. 5 and thermal comfort in a real room using deep reinforcement learning,” *Energy and Buildings*, vol. 295, p. 113 340, 2023.

[45] Y. Sakuma and H. Nishi, “Adaptive control method of hvac for uniformizing comfort at japanese residential living rooms using deep reinforcement learning,” *IEEJ Transactions on Electronics, Information and Systems*, vol. 141, no. 3, pp. 373–382, 2021.

[46] F. Cicirelli, A. Guerrieri, C. Mastroianni, G. Spezzano, and A. Vinci, “Thermal comfort management leveraging deep reinforcement learning and human-in-the-loop,” in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, IEEE, 2020, pp. 1–6.

[47] L. Yu, Y. Sun, Z. Xu, *et al.*, “Multi-agent deep reinforcement learning for hvac control in commercial buildings,” *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 407–419, 2020.

[48] U. D. of Energy, *Energyplus: A whole building energy simulation program*. [Online]. Available: `https://energyplus.net`.

[49] A. Galataud, *Rllib-energyplus: Simple energyplus environments for control optimization using reinforcement learning*. [Online]. Available: `https://github.com/airboxlab/rllib-energyplus`.

[50] *Bright lab b. amsterdam*, `https://b-events.amsterdam/en/site/event_rooms/bright-lab/`.

[51] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: `http://jmlr.org/papers/v22/20-1364.html`.

A

Source Code Snippets

A.1. Human Class

```
1
2  import numpy as np
3  from pythermalcomfort.models import pmv_ppd
4  from pythermalcomfort.utilities import v_relative, clo_dynamic
5  from pythermalcomfort.utilities import met_typical_tasks
6  from pythermalcomfort.utilities import clo_individual_garments
7
8
9  class Human:
10     def __init__(self, icl:float=1.1, met:float=1.4,
11                  exp_a:float=1.0, exp_b:float=2.0, exp_c:float=1.0, exp_d:float=2.0) -> None:
12         # pmv parameters
13         self.icl = icl # total clothing insulation, [clo]
14         self.met = met  # activity metabolic rate, [met]
15
16
17         # interaction parameters
18         # self.dist_skew = 0.0  # skewness of the probability distribution
19         # self.dist_loc = 0.0  # location of the probability distribution
20         # self.dist_scale = 1 # scale of the probability distribution
21
22         # interaction probability parameters
23         self.prob_func = "exp_zeroed" # Probability function to use. Options: "sigmoid", "exp
               ", "exp_zeroed"
24
25         # P(pmv) = exp(ax-b) + exp(-cx-d)
26         self.exp_a = exp_a
27         self.exp_b = exp_b
28         self.exp_c = exp_c
29         self.exp_d = exp_d
30         self.normalizer = 1.0
31
32
33     def calcpmv(self, tdb: float, tr: float, v: float, rh: float) -> float:
34         """
35         Calculate the Predicted Mean Vote (PMV) based on the input variables.
36
37         Parameters:
38         - tdb: Dry bulb air temperature, [°C]
39         - tr: Mean radiant temperature, [°C]
40         - v: Average air speed, [m/s]
41         - rh: Relative humidity, [%]
42
43         Returns:
44         - pmv: Predicted Mean Vote.
45         """
46         vr = v_relative(v=v, met=self.met)
```

```
        clo = clo_dynamic(clo=self.icl, met=self.met)
        results = pmv_ppd(tdb=tdb, tr=tr, vr=vr, rh=rh, met=self.met, clo=clo, standard="
            ASHRAE")
        return results['pmv']

    def temp2pmv(self, min_tdb = 10.0, max_tdb = 40.0, step_tdb = 0.5, tr = 25, v = 0.1, rh
        =50) -> dict:
        """ Uniformly samples the pmv values varying the temperature
        min_tdb: min dry bulb air temperature, [°C]
        max_tdb: max dry bulb air temperature, [°C]
        step_tdb: step of the dry bulb air temperature, [°C]
        tr: mean radiant temperature, [°C]
        v: average air speed, [m/s]
        rh: relative humidity, [%]
        """
        pmvs = {"pmv": [], "tdb": []}
        vr = v_relative(v=v, met=self.met)
        clo = clo_dynamic(clo=self.icl, met=self.met)
        for tdb in np.arange(min_tdb, max_tdb, step_tdb):
            results = pmv_ppd(tdb=tdb, tr=tr, vr=vr, rh=rh, met=self.met, clo=clo, standard="
                ASHRAE")
            pmvs["pmv"].append(results['pmv'])
            pmvs["tdb"].append(tdb)
        return pmvs

    def calcprobability(self, pmv: float, ) -> float:
        """
        Calculate the probability of complaint based on the current pmv.

        Parameters:
        - pmv: Current pmv.

        Returns:
        - probability: Probability of complaint.
        """
        if self.prob_func == "exp":
            probability = np.exp(self.exp_a * pmv - self.exp_b) + np.exp(-self.exp_c * pmv -
                self.exp_d)
        elif self.prob_func == "exp_zeroed":
            probability = np.exp(self.exp_a * pmv - self.exp_b) + np.exp(-self.exp_c * pmv -
                self.exp_d) - np.exp(-self.exp_d) - np.exp(-self.exp_b)
        else:
            probability = 0.0
        # limit  probabilities between 0 and 1
        probability = max(0.0, min(1.0, self.normalizer * probability))

        return probability
```

A.2. EnergyPlus gym Abstract Class

```
    class EnergyPlusEnv(gym.Env, metaclass=abc.ABCMeta):
    """Base, abstract EnergyPlus gym environment.

    This class implements the OpenAI gym (now gymnasium) API. It must be subclassed to
    implement the actual environment.
    """

    def __init__(self, env_config: Dict[str, Any], reward_type: str = "pmv", w_file: str = '
        Train'):
        self.spec = gym.envs.registration.EnvSpec(f"{self.__class__.__name__}")

        self.env_config = env_config
        self.episode = -1
        self.timestep = 0

        self.observation_space = self.get_observation_space()
        self.last_obs = {}

        self.action_space = self.get_action_space()
        self.default_action = self.post_process_action(self.action_space.sample())
```

```
20
21          self.energyplus_runner: Optional[EnergyPlusRunner] = None
22          self.obs_queue: Optional[Queue] = None
23          self.act_queue: Optional[Queue] = None
24
25          self.reward_history = []
26          self.obs_history = []
27          self.pmv_history = []
28
29          if reward_type in ["pmv", "human", "zero"]:
30              self.reward_type = reward_type
31          else:
32              raise ValueError(f"Invalid reward type: {reward_type}")
33
34          # each day is 96 timesteps (15 minutes)
35          self.episode_length = 96
36
37          self.w_file = w_file
38
39          self.runner_config = RunnerConfig(
40              epw=self.get_weather_file(),
41              idf=self.get_idf_file(),
42              output=self.env_config["output"],
43              variables=self.get_variables(),
44              meters=self.get_meters(),
45              actuators=self.get_actuators(),
46              csv=self.env_config.get("csv", False),
47              verbose=self.env_config.get("verbose", False),
48              eplus_timestep_duration=self.env_config.get("eplus_timestep_duration", 0.25),
49          )
50
51      @abc.abstractmethod
52      def get_weather_file(self) -> Union[Path, str]:
53          """Returns the path to a valid weather file (.epw).
54
55          This method can be used to randomize training data by providing different weather
56          files. It's called on each reset()
57          """
58
59      @abc.abstractmethod
60      def get_idf_file(self) -> Union[Path, str]:
61          """Returns the path to a valid IDF file."""
62
63      @abc.abstractmethod
64      def get_observation_space(self) -> gym.Space:
65          """Returns the observation space of the environment."""
66
67      @abc.abstractmethod
68      def get_action_space(self) -> gym.Space:
69          """Returns the action space of the environment."""
70
71      @abc.abstractmethod
72      def compute_step_reward(self, obs: Dict[str, float]) -> float:
73          """Computes the reward for the given observation."""
74
75      @abc.abstractmethod
76      def get_variables(self) -> Dict[str, Tuple[str, str]]:
77          """Returns the variables to track during simulation."""
78
79      @abc.abstractmethod
80      def get_meters(self) -> Dict[str, str]:
81          """Returns the meters to track during simulation."""
82
83      @abc.abstractmethod
84      def get_actuators(self) -> Dict[str, Tuple[str, str, str]]:
85          """Returns the actuators to control during simulation."""
86
87      def post_process_action(self, action: Union[float, List[float]]) -> Union[float, List[
            float]]:
88          """Post-processes the action(s) before sending it to EnergyPlus.
89
```

```
        This method can be used to implement constraints on the actions, like rescaling.
        Default implementation returns the action unchanged.
        """
        return action

    def reset(self, *, seed: Optional[int] = None, options: Optional[Dict[str, Any]] = None):
        print("Episode:", self.episode, "␣finised␣at␣Timestep:", self.timestep)

        self.episode += 1
        # self.last_obs = self.observation_space.sample()

        # reset history
        self.reward_history = []
        self.obs_history = []
        self.pmv_history = []

        if self.energyplus_runner is not None:
            self.energyplus_runner.stop()

        # observations and actions queues for flow control
        # queues have a default max size of 1
        # as only 1 E+ timestep is processed at a time
        self.obs_queue = Queue(maxsize=1)
        self.act_queue = Queue(maxsize=1)

        self.energyplus_runner = EnergyPlusRunner(
            episode=self.episode,
            obs_queue=self.obs_queue,
            act_queue=self.act_queue,
            runner_config=self.runner_config,
        )
        self.energyplus_runner.start()

        # wait until E+ is ready.
        # self.last_obs = obs = self.energyplus_runner.init_exchange(default_action=self.
            default_action)
        _random_action = self.post_process_action(self.action_space.sample()[0])
        # print("Reseting with Random Action:", _random_action)
        self.last_obs = obs = self.energyplus_runner.init_exchange(default_action=
            _random_action)
        return np.array(list(obs.values())), {}

    def step(self, action):
        self.timestep += 1
        done = False

        # check for simulation errors
        if self.energyplus_runner.failed():
            raise RuntimeError(f"EnergyPlus␣failed␣with␣{self.energyplus_runner.sim_results['
                exit_code']}")

        # simulation_complete is likely to happen after last env step()
        # is called, hence leading to waiting on queue for a timeout
        if self.energyplus_runner.simulation_complete:
            done = True
            obs = self.last_obs
        else:
            # post-process action
            action_to_apply = self.post_process_action(action)
            # do not post-process action
            # action_to_apply = action
            if not isinstance(action_to_apply, np.float32):
                raise ValueError(f"Invalid␣action␣type:␣{type(action_to_apply)}")

            # Enqueue action (sent to EnergyPlus through dedicated callback)
            # then wait to get next observation.
            # Timeout is set to 2s to handle end of simulation cases, which happens async
            # and materializes by worker thread waiting on this queue (EnergyPlus callback
            # not consuming anymore).
            # Timeout value can be increased if E+ timestep takes longer
            timeout = 2
```

```
            try:
                self.act_queue.put(action_to_apply, timeout=timeout)
                obs = self.obs_queue.get(timeout=timeout)
            except (Full, Empty):
                obs = None
                print("Timeout waiting for observation")
                pass

            # obs can be None if E+ simulation is complete
            # this materializes by either an empty queue or a None value received from queue
            if obs is None:
                done = True
                obs = self.last_obs
            else:
                self.last_obs = obs

            # finish episode if episode_length is reached
            if self.timestep % self.episode_length == 0 and self.timestep > 0:
                done = True
                # print("Last action:", action_to_apply)

        # compute reward
        reward = self.compute_step_reward(obs)

        # compute pmv
        _pmv = pmv(tdb=obs["air_tmp"], tr=obs["rad_tmp"], vr=0.1, rh=obs["air_hum"], met=1.4,
             clo=1.1)

        # store history
        self.reward_history.append(reward)
        self.obs_history.append(obs)
        self.pmv_history.append(_pmv)

        if done:
            self.save_history("./tmp/history_"+self.w_file+".pkl")

        obs_vec = np.array(list(obs.values()))
        return obs_vec, reward, done, False, {}

    def close(self):
        if self.energyplus_runner is not None:
            self.energyplus_runner.stop()

    def save_history(self, filepath):
        # combine the dicts in obs_history to single dict
        comb_history = {key: [obs[key] for obs in self.obs_history] for key in self.
            obs_history[0]}
        comb_history['reward'] = self.reward_history
        comb_history['pmv'] = self.pmv_history

        # append the combined history to a pickle file
        with open(filepath, 'ab') as f:
            pkl.dump(comb_history, f)
```

A.3. Step Reward Calculation

```
        def compute_step_reward(self, obs: Dict[str, float]) -> float:
        """A reward function that penalizes on human complaints and rewards no complaints."""

        if self.reward_type == "zero":
            return 0.0
        elif self.reward_type == "pmv":
            # calculate the pmv value
            _pmv = pmv(tdb=obs["air_tmp"], tr=obs["rad_tmp"], vr=self.pmv_dict["vr"], rh=obs
                ["air_hum"], met=self.pmv_dict["met"], clo=self.pmv_dict["clo"])
            # return negative distance of pmv from 0
            reward = -1*abs(_pmv)
            # if reward is nan, return -4
            if np.isnan(reward):
                return -4
```

```
            else:
                return reward
        elif self.reward_type == "human":
            # no complaint counter and threshold
            no_complaint = 0
            no_complaint_threshold = 4

            # cumulative reward for timestep
            step_cum_reward = 0

            # iterate over humans
            for human in self.humans:
                # calculate pmv value for the current human
                temp_pmv = human.calcpmv(obs["air_tmp"], obs["rad_tmp"], self.pmv_dict["vr"],
                     obs["air_hum"])

                # get probability of complaint
                prob = human.calcprobability(temp_pmv)

                # generate random number between 0 and 1
                rand = np.random.rand()

                # check if the human complains
                complaint = rand < prob

                if complaint:
                    step_cum_reward += -1
                    no_complaint = 0
                else:
                    no_complaint += 1

                if no_complaint >= no_complaint_threshold:
                    step_cum_reward += 0.2
            return step_cum_reward
```

B

Figures

B.1. Training

B.1.1. Change in Probability (k)

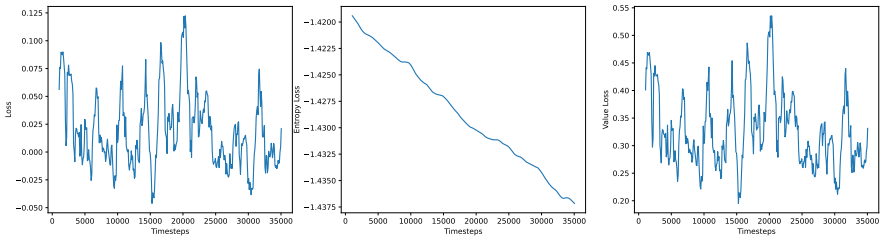

Figure B.1: human k 0.2 losses

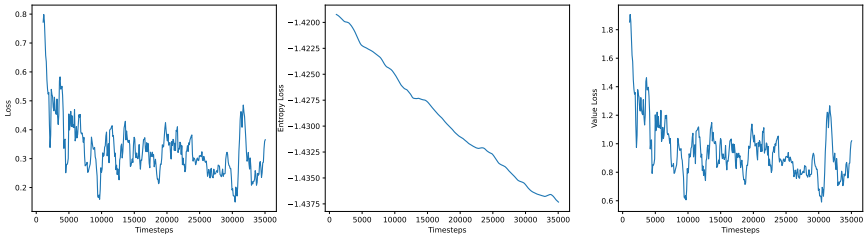

Figure B.2: human k 0.5 losses

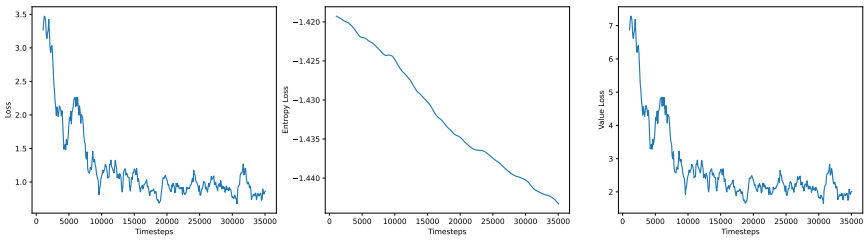

Figure B.3: human k 1.0 losses

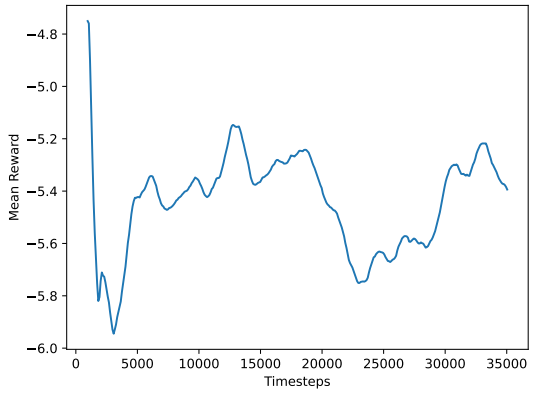

Figure B.4: human k 0.2 mean reward

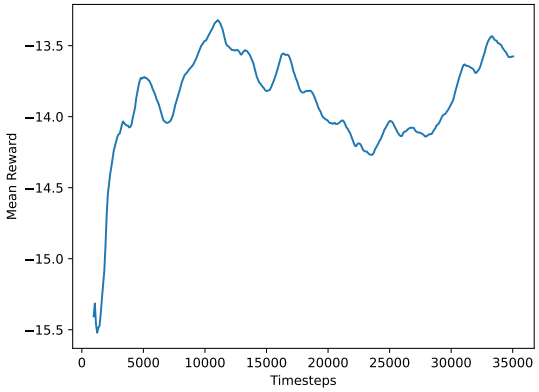

Figure B.5: human k 0.5 mean reward

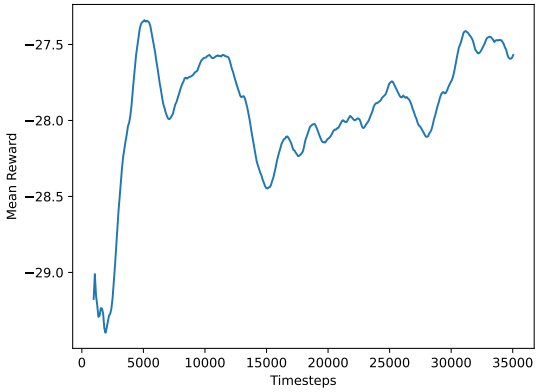

Figure B.6: human k 1.0 mean reward

B.1.2. Zeroed Human Change in Probability (k)

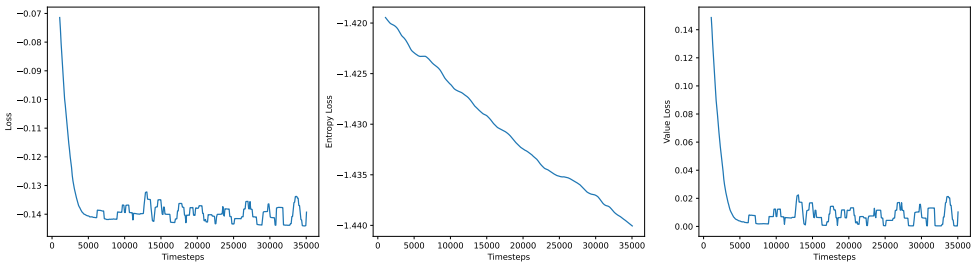

Figure B.7: zeroed probability human k 0.1 losses

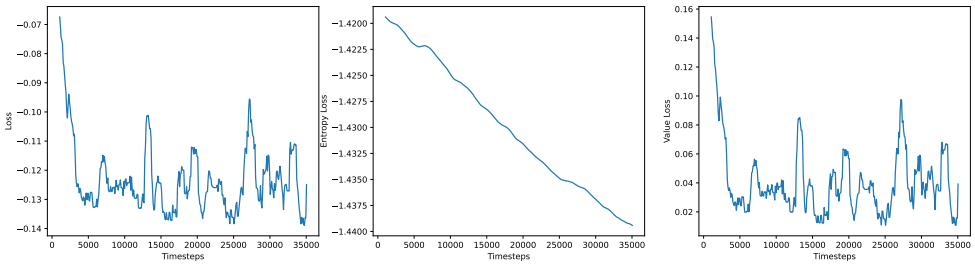

Figure B.8: zeroed probability human k 0.5 losses

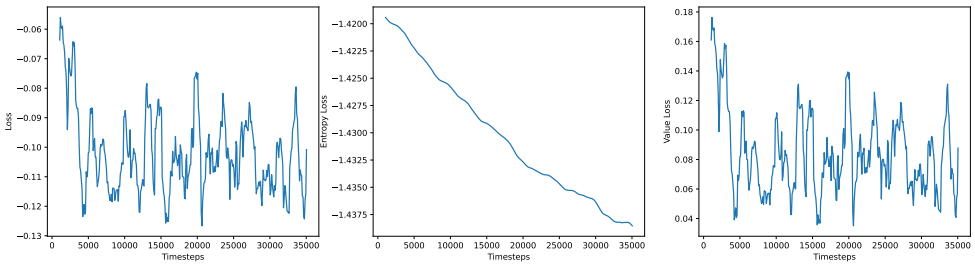

Figure B.9: zeroed probability human k 1.0 losses

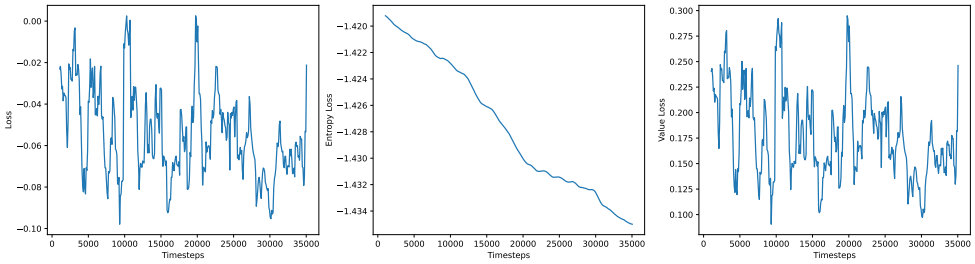

Figure B.10: zeroed probability human k 2.0 losses

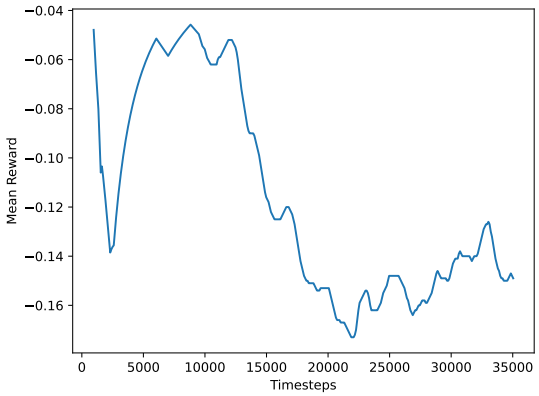

Figure B.11: zeroed probability human k 0.1 mean reward

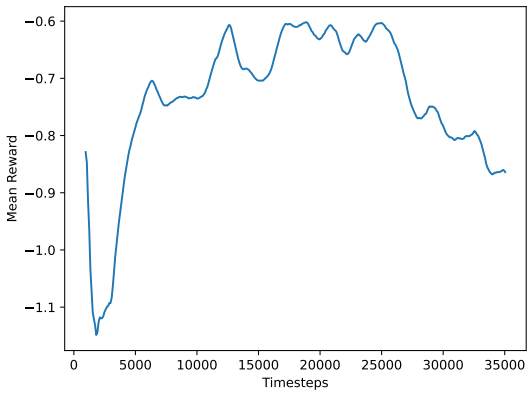

Figure B.12: zeroed probability human k 0.5 mean reward

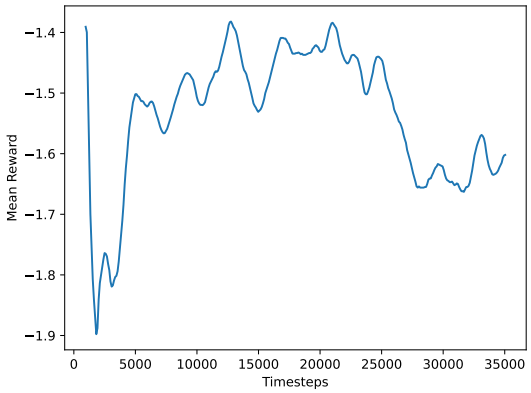

Figure B.13: zeroed probability human k 1.0 mean reward

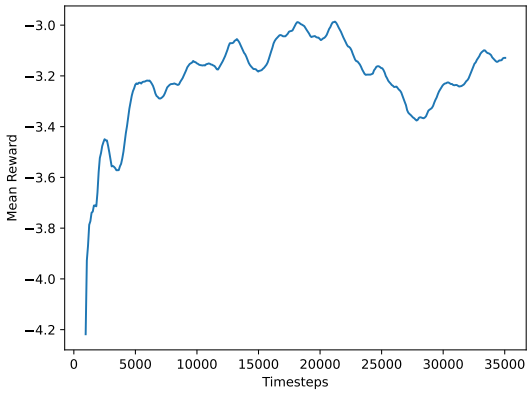

Figure B.14: zeroed probability human k 2.0 mean reward

B.1.3. Change in Epochs

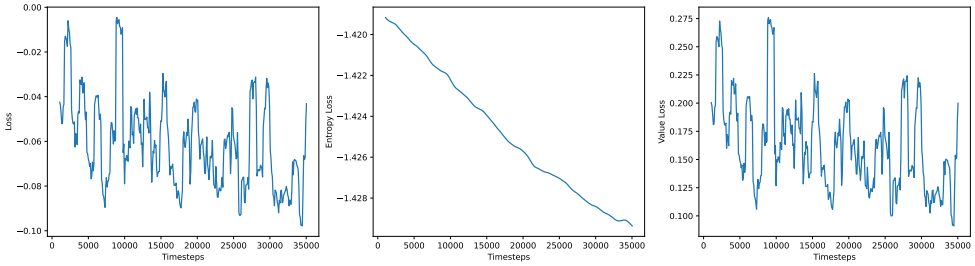

Figure B.15: human 5 epochs losses

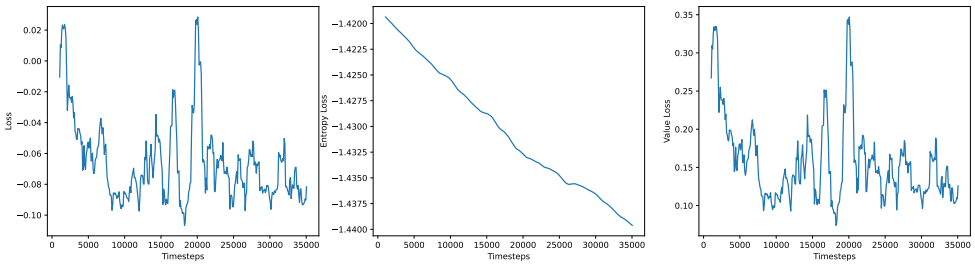

Figure B.16: human 10 epochs losses

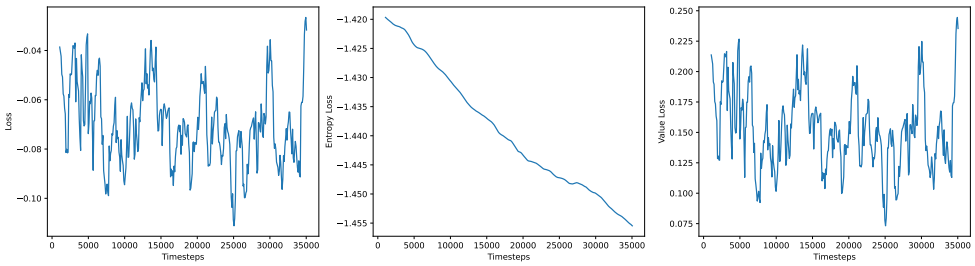

Figure B.17: human 20 epochs losses

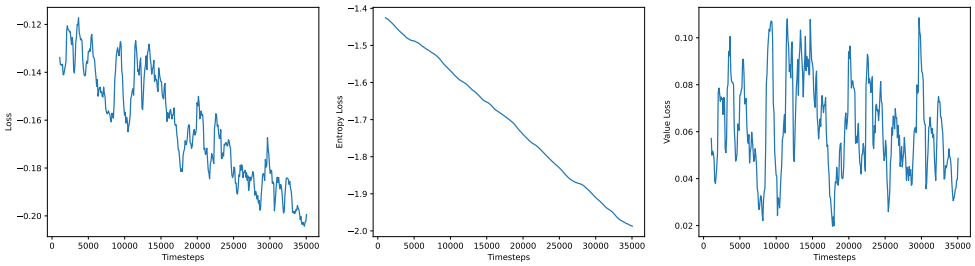

Figure B.18: human 250 epochs losses

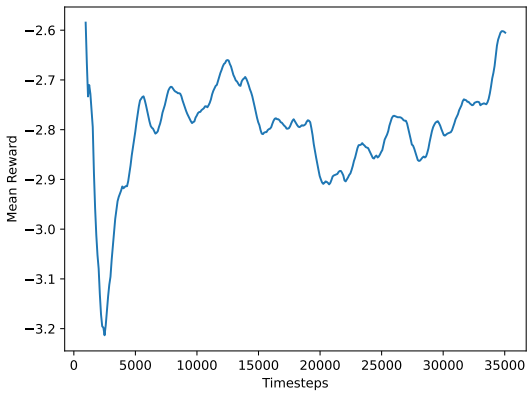

Figure B.19: human 5 epochs mean reward

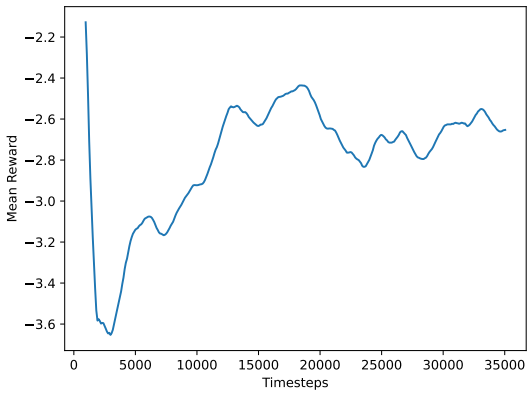

Figure B.20: human 10 epochs mean reward

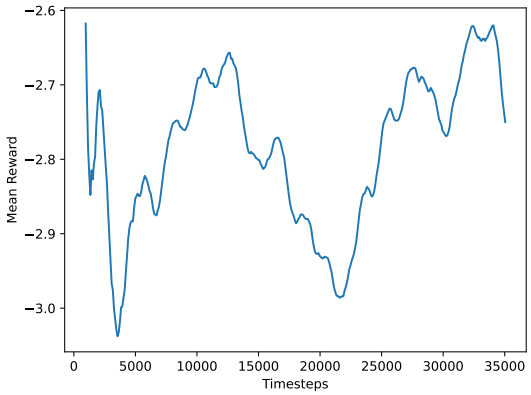

Figure B.21: human 20 epochs mean reward

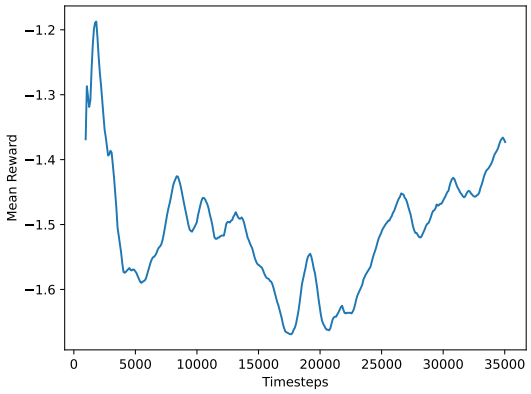

Figure B.22: human 250 epochs mean reward

B.2. Testing

B.2.1. Change in Probability (k)

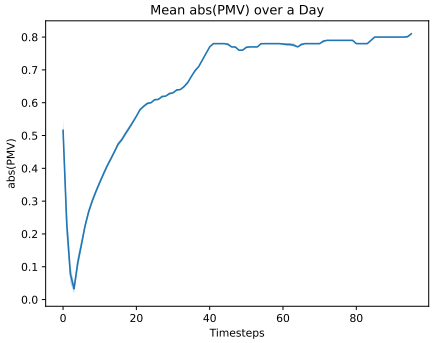

Figure B.23: human k 0.2 mean abs(PMV) over day

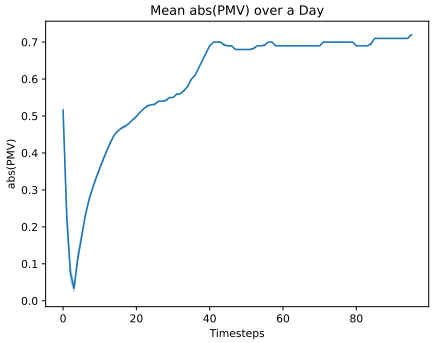

Figure B.24: human k 0.5 mean abs(PMV) over day

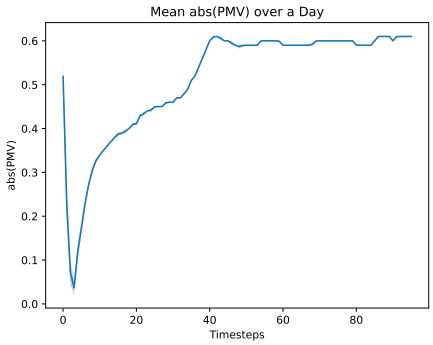

Figure B.25: human k 1.0 mean abs(PMV) over day

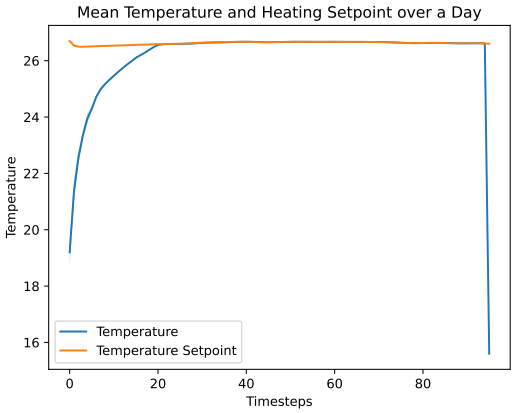

Figure B.26: human k 0.2 mean heating setpoint and temperature over day

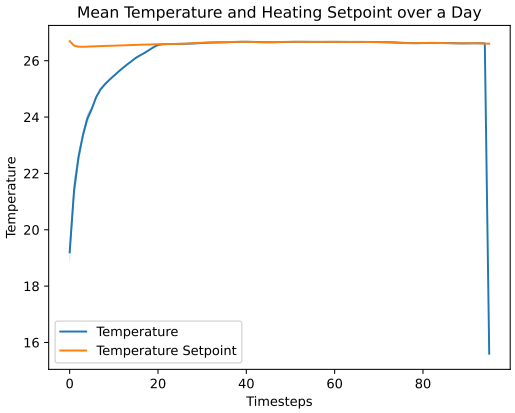

Figure B.27: human k 0.5 mean heating setpoint and temperature over day

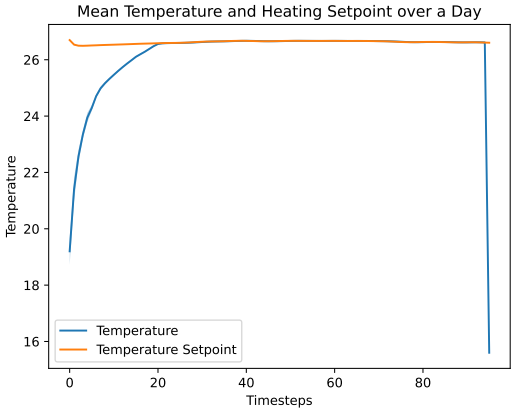

Figure B.28: human k 1.0 mean heating setpoint and temperature over day

B.2.2. Zeroed Human Change in Probability (k)

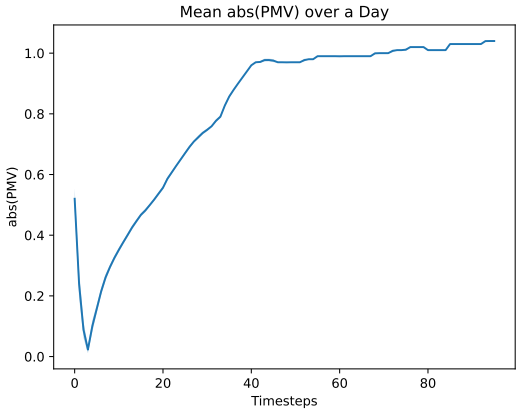

Figure B.29: zeroed probability human k 0.1 mean abs(PMV) over day

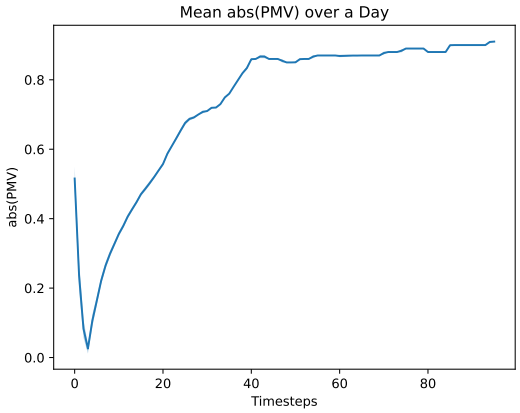

Figure B.30: zeroed probability human k 0.5 mean abs(PMV) over day

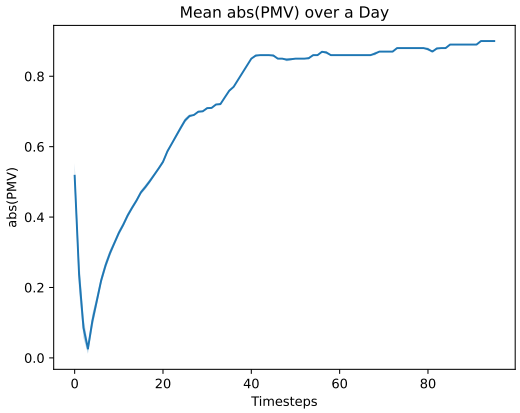

Figure B.31: zeroed probability human k 1.0 mean abs(PMV) over day

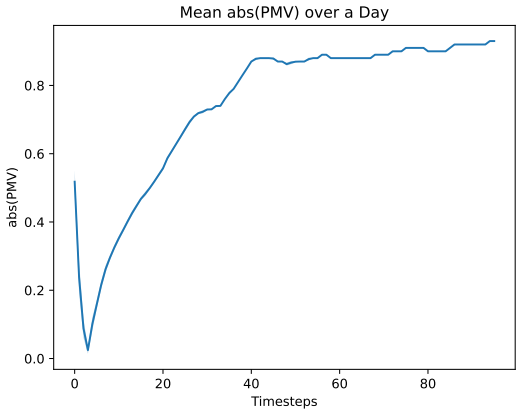

Figure B.32: zeroed probability human k 2.0 mean abs(PMV) over day

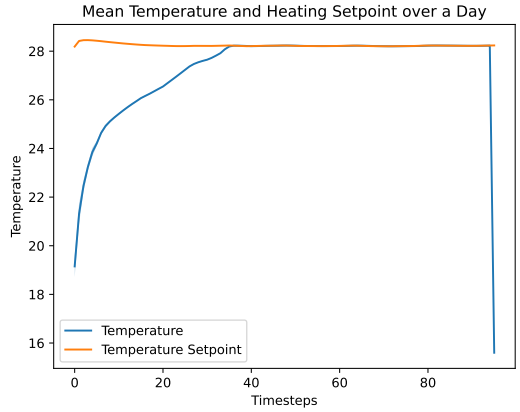

Figure B.33: zeroed probability human k 0.1 mean heating setpoint and temperature over day

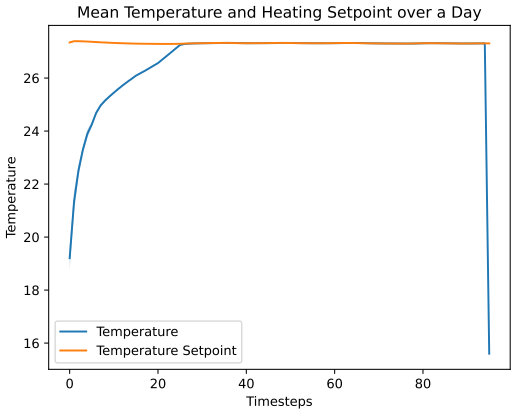

Figure B.34: zeroed probability human k 0.5 mean heating setpoint and temperature over day

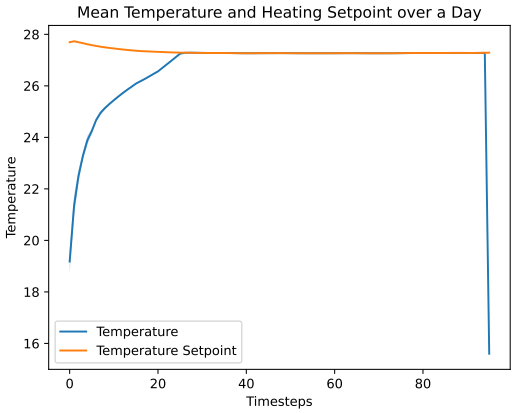

Figure B.35: zeroed probability human k 1.0 mean heating setpoint and temperature over day

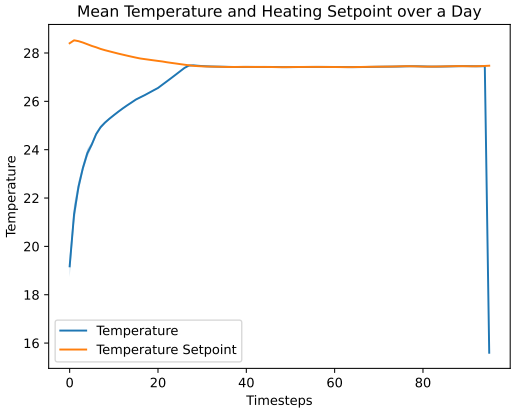

Figure B.36: zeroed probability human k 2.0 mean heating setpoint and temperature over day

B.2.3. Change in Epochs

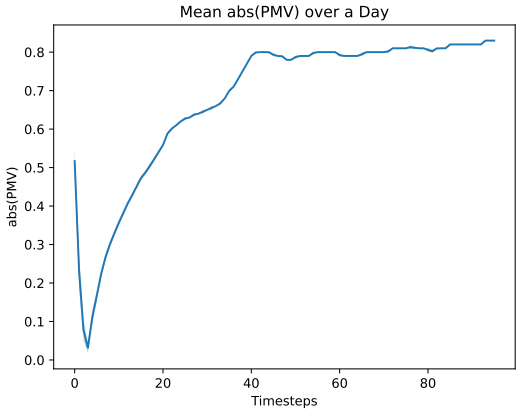

Figure B.37: human 5 epochs mean abs(PMV) over day

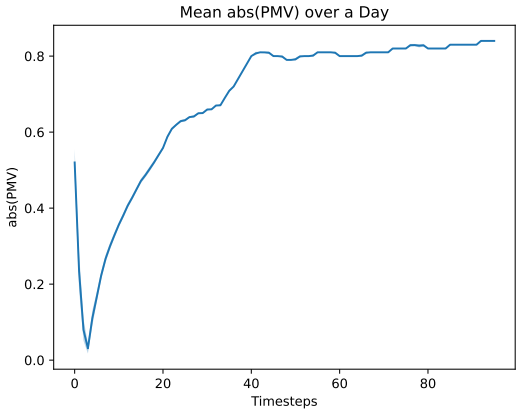

Figure B.38: human 10 epochs mean abs(PMV) over day

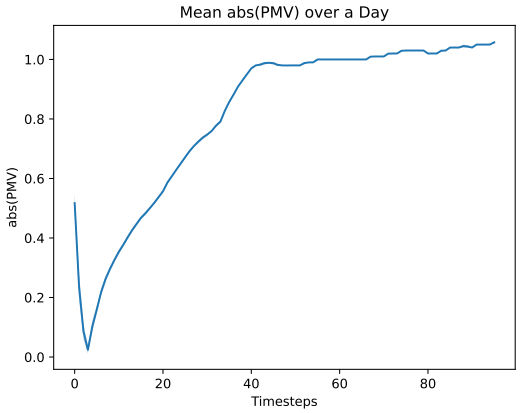

Figure B.39: human 20 epochs mean abs(PMV) over day

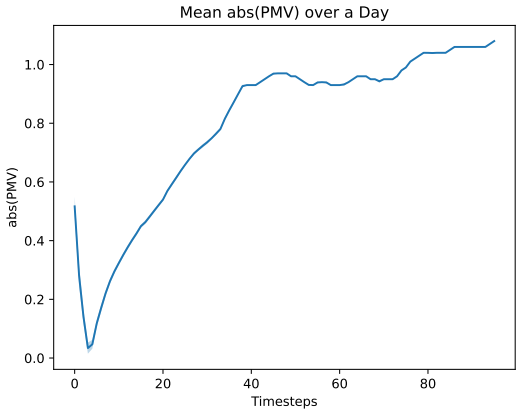

Figure B.40: human 250 epochs mean abs(PMV) over day

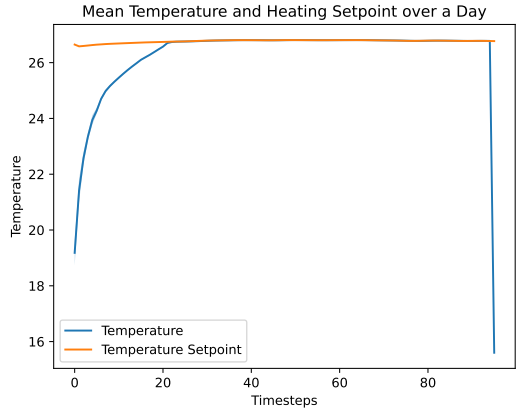

Figure B.41: human 5 epochs mean heating setpoint and temperature over day

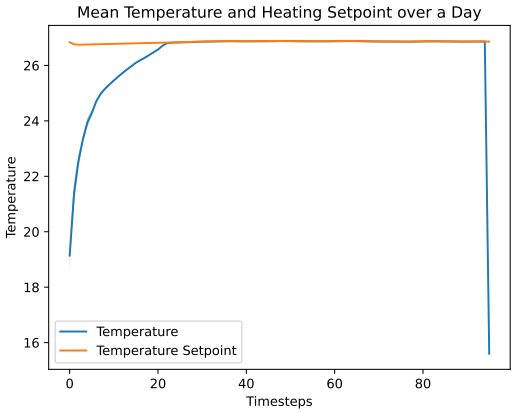

Figure B.42: human 10 epochs mean heating setpoint and temperature over day

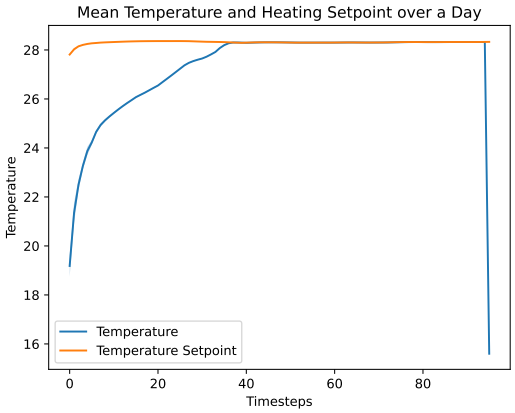

Figure B.43: human 20 epochs mean heating setpoint and temperature over day

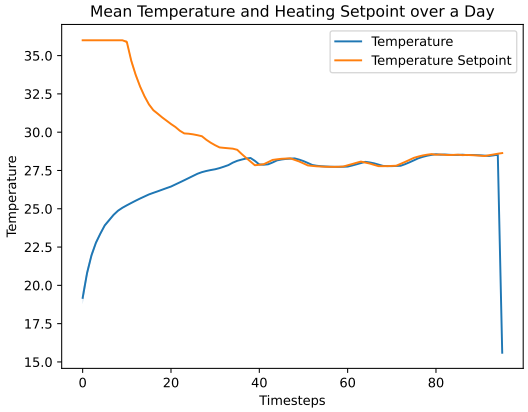

Figure B.44: human 250 epochs mean heating setpoint and temperature over day