

Master's Programme in Computer, Communication and Information Sciences

# Safe Reinforcement Learning based Optimization of a Cruise Ship Energy Management System

---

**Erald Shahinas**

© 2025

This work is licensed under a [Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/) “Attribution-NonCommercial-ShareAlike 4.0 International” license.



---

**Author** Erald Shahinas

---

**Title** Safe Reinforcement Learning based Optimization of a Cruise Ship Energy Management System

---

**Degree programme** Computer, Communication and Information Sciences

---

**Major** Machine Learning, Data Science, and Artificial Intelligence

---

**Supervisor** Dr. Udayanto Atmojo

---

**Advisor** MSc. Carl Akira King

---

**Date** 29 July 2025

**Number of pages** 53

**Language** English

---

**Abstract**

The cruise ship industry is a large emitter of greenhouse gases. Multiple aspects must be tackled to help make the industry more sustainable and adhere to the net-zero emission goal set by the International Maritime Organization (IMO). One aspect is the optimization of the energy management of the ship, which can lead to large gains in energy savings. Traditional optimization approaches, such as linear optimization struggle to scale to large stochastic problem spaces such as that of a ship energy system. Modern approaches, such as reinforcement learning, are shown to be more effective at the task at hand. However, these approaches often lack a direct focus on safety, thus making them impractical for real-world applications.

This thesis presents work in the development of a safe reinforcement learning (RL) system for ship energy optimization. As part of the system, a ship’s microgrid environment was developed alongside a reinforcement learning agent. The results showcase the potential of RL approaches for energy optimization as opposed to alternative approaches. The inclusion of safety-focused methods, such as utilizing a safety shield and a large language model-generated reward function, was shown to be effective at enhancing safety, reducing the number of safety violations by 87 % when the agent was trained and tested in a surrogate model.

---

**Keywords** Reinforcement learning, large language models , energy optimization , safety , surrogate modeling

---

## **Preface**

I want to thank my family for their unwavering support throughout this journey. I also want to thank my supervisor, Dr. Udayanto Atmojo, who gave me the opportunity to work on this topic, and my advisor, MSc. Akira King for guiding me through the process.

Otaniemi, 29 July 2025

Erald Shahinas

# Contents

<b>Abstract</b>	<b>3</b>
<b>Preface</b>	<b>4</b>
<b>Contents</b>	<b>5</b>
<b>1 Introduction</b>	<b>8</b>
<b>2 Background and relevant work</b>	<b>10</b>
2.1 Inside a cruise ship’s energy management system . . . . .	10
2.1.1 HVAC system . . . . .	10
2.1.2 Energy storage system . . . . .	11
2.2 Control theory . . . . .	12
2.3 Surrogate models . . . . .	13
2.4 Reinforcement learning . . . . .	14
2.4.1 Value-based vs policy-based methods . . . . .	16
2.4.2 Proximal Policy Gradient . . . . .	17
2.4.3 Bounding for continuous action spaces . . . . .	19
2.4.4 Shielding for safe reinforcement learning . . . . .	21
2.5 Large language models . . . . .	21
2.6 System-theoretic process analysis . . . . .	22
2.7 Relevant work in ship energy optimization . . . . .	23
2.8 Relevant work in large language models for reward function generation	23
<b>3 Research methods and implementation</b>	<b>25</b>
3.1 Overview of the methodology . . . . .	25
3.2 Simulation . . . . .	26
3.2.1 Storage . . . . .	30
3.2.2 Generation . . . . .	30
3.3 Black box surrogate model . . . . .	31
3.3.1 Forecasting model approach . . . . .	32
3.3.2 Surrogate model approach . . . . .	33
3.4 Reinforcement learning training environment . . . . .	34
3.5 Reinforcement learning agent . . . . .	34
3.5.1 P3O agent with a continuous action space . . . . .	35
3.5.2 P3O agent with a discrete action space . . . . .	35
3.5.3 PPO agent . . . . .	36
3.6 Reinforcement learning training . . . . .	36
3.7 LLM reward function generation . . . . .	36
3.8 Safety shield . . . . .	37
3.9 White-box simulation model . . . . .	37
3.10 Open loop control . . . . .	38

<b>4</b>	<b>Results and Discussion</b>	<b>39</b>
4.1	Surrogate model . . . . .	39
4.2	Comparing different RL Agents . . . . .	41
4.3	LLM reward function generation . . . . .	43
4.4	Safety shield . . . . .	44
4.5	Incorporating guardrails . . . . .	44
4.6	Results of testing in the white-box environment . . . . .	46
<b>5</b>	<b>Conclusion</b>	<b>47</b>
5.1	Limitations and Future work . . . . .	47
5.2	Conclusions . . . . .	48

# Symbols and abbreviations

## Abbreviations

GHG	Greenhouse Gas
IMO	International Maritime Organization
HVAC	Heating, Ventilation, and Air Conditioning
EMS	Energy Management System
RL	Reinforcement Learning
LLM	Large Language Models
PPO	Proximal Policy Optimization
SAC	Soft Actor-Critic
SC	Safety Constraints
AHU	Air Handling Unit
VAV	Variable Air Volume
FCU	Fan Coil Unit
QoI	Quantity of Interest
DQN	Deep Q Networks
GAE	Generalized Advantage Estimation
P3O	Penalized Proximal Policy Optimization
GA	Genetic Algorithms
NSGA-II	Non-dominated Sorting Genetic Algorithm
DDPG	Deep Deterministic policy gradient
RFG	Reward Function Generator
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
AI	Artificial Intelligence

# 1 Introduction

Maritime transport is an important and growing sector of the global economy, corresponding to 80% of global trade and 2.9% of global greenhouse gas (GHG) emissions as of 2018 [1], [2]. The International Maritime Organization (IMO) believes that the amount of emissions by this industry could be greatly reduced in the following decades and has set a net-zero emission goal by 2050 [1]. In cruise ships, the hotel system, in particular the heating, ventilation, and air conditioning (HVAC) system, consumes almost 40% of the total energy [3], [4]. Optimizing the operation of this system could lead to large gains in energy savings.

Modern cruise ships are large and complex energy systems. Many factors add a degree of uncertainty to their operation, such as weather or passenger behavior. Optimizing the energy management system (EMS) of a ship encompasses controlling the different parameters and control points of the ship in a way that minimizes energy losses due to poor management. However, the complexity of a modern ship's microgrid systems makes it difficult for a human operator to manage it. Traditional optimization techniques such as linear programming are usually employed as they perform better than a human operator. However, they are best suited to smaller deterministic problems. Thus, they may not be able to cope with the uncertainty of a ship's energy system [5]. One promising area of research to resolve this issue is the use of reinforcement learning (RL) to optimize energy management.

Reinforcement learning algorithms can perform well in complex and stochastic environments. The main idea behind reinforcement learning is to teach an agent how to achieve a specific goal or to navigate in a particular environment [6]. The agent learns a policy, which is the set of actions it will take given the state of the agent in relation to the environment. In the case of EMS optimization, the agent would try to optimize the energy consumption while managing the control points inside the ship's control system.

However, real cruise ships are safety-critical systems with explicit regulations regarding the management of the energy system to prevent harm. Standard RL methods are not designed with a focus on safety. Safe RL methods are a subdomain of RL methods concerned with learning policies that adhere to safety constraints. As such, they are better suited to safety-critical systems. Different approaches can be utilized, such as employing safe RL algorithms or utilizing a safety shield that blocks unsafe actions [7], [8].

A crucial part of an RL agent is the reward function, which specifies which behaviors should be rewarded and which should be penalized. The agent will learn a policy that maximizes the reward from the environment. A good reward signal should teach an agent the necessary preliminary steps to achieve a particular goal. However, designing the reward function that will be used by the agent, also known as reward shaping, can be difficult depending on the problem. Reward shaping is often done by a human engineer. However, depending on how large and complex the challenge is, it can be time-consuming and expensive. A novel approach in research is to bypass the human engineer and use large language models (LLMs) to generate the reward function [9]. This removes the technical complexity from reward shaping and greatly

reduces working hours and engineering costs. By feeding an LLM model a description of the challenge and the objective, it can generate an appropriate reward function [9], [10].

The aim of this thesis is the development of a safe RL controller system to optimize the energy management of a cruise ship. To accomplish this objective, it was experimented with different RL algorithms, RL environments, and reward functions. To assess the performance of the methods compared, training and testing data was collected for the RL agents tested in the form of energy consumption and safety violations.

The proposed system consists of the following parts:

- A white box simulation of a ship's energy management system
- A black box surrogate simulation model
- A reinforcement learning agent
- A large language model reward function generator
- A safety shield

The white-box simulation of a ship's energy management system simulates the physical properties of a real system and provides the foundation on top of which all the other parts work. The surrogate model is a lightweight approximation of the white-box model that enables interaction with the RL agent. The RL agent serves as the top controller of the system. The LLM reward function generator generates a reward function that is used during RL training. A safety shield is employed during testing of the RL agent.

This thesis addresses several questions related to the development and use of the system. Firstly, it addresses the effectiveness of utilizing RL methods for ship energy optimization compared to other approaches. Secondly, it explores the capability of safe RL methods in enhancing safety. Finally, it considers the efficacy of LLM-generated reward functions over traditional handcrafted approaches.

This thesis is a part of the SEASHINE project, which aims to achieve safe RL ship energy optimization [11]. As part of the project, safety constraints (SC) related to ship EMS operation were defined. The constraints were used for LLM reward function generation and for the implementation of the safety shield.

The rest of the thesis is organized as follows. Chapter 2 provides context to the topic and gives a review of the current literature related to the methods used. Chapter 3 describes the methodology used in the thesis. Chapter 4 presents the results of the experiments conducted, and finally, Chapter 5 states the key takeaways and conclusions from the thesis.

## 2 Background and relevant work

This chapter provides context for the topic and gives an in-depth explanation of the parts covered. Furthermore, it introduces solutions already found in literature and how this thesis differs from the solutions available.

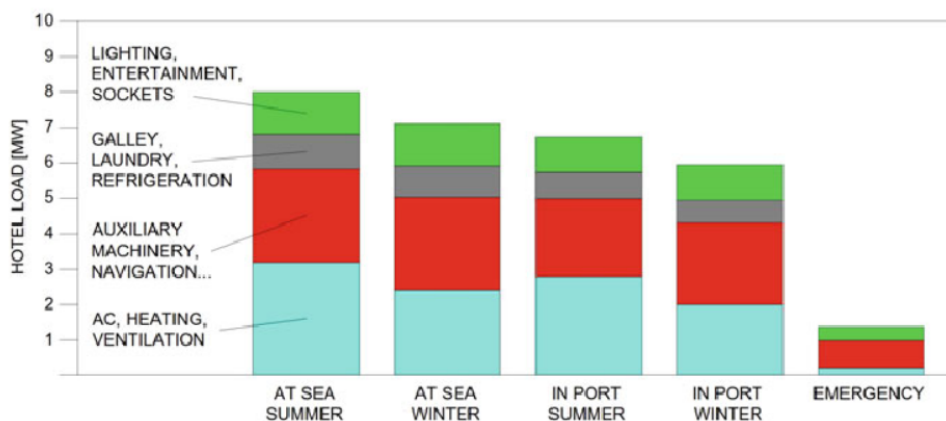
### 2.1 Inside a cruise ship's energy management system

A cruise ship can be best defined as a maritime vessel with the main purpose of entertaining the passengers onboard [4]. Modern cruise ships, regardless of their size, are operated in a similar manner. There are multiple systems inside the ship that make the successful operation of a ship possible. The two main groups of systems are: the machinery systems that are responsible for propulsion, power generation, and essential operation, and the hotel systems that support passenger comfort, similar to a real hotel.

As passenger comfort and entertainment are crucial for a cruise ship, the hotel system is one of the biggest energy consumers on board, using almost half of the total energy consumption [4].

#### 2.1.1 HVAC system

The hotel system's main consumer is the heating, ventilation, and air conditioning (HVAC) system, which consumes approximately 40% of the total hotel load, as can be seen in Figure 1. The HVAC system is the second largest consumer on a cruise ship after propulsion [4].



**Figure 1:** Hotel load of a common cruise ship. The biggest consumer is the HVAC system [4].

The HVAC system is made of multiple parts that aid the comfort of passengers on the ship. Air handling units (AHUs) are responsible for operating air in and out of cabins, hallways, staircases, and other compartments. Multiple air handling units (AHUs) are located inside an air conditioning room, which typically serves a deck or floor of the ship. While fresh air is important, the AHUs also treat the air before

supplying it to the cabins. A cooling coil is used in order to cool and dehumidify the air, and a heating coil is used in order to heat it. In order to provide cold or hot liquid for the coils, the HVAC system relies on chillers and boilers to cool or heat the air, depending on the need. Pumps make it possible to move the treated liquid inside the AHUs.

The air handling units provide constant air in and out of the compartments, however, the need for fresh air varies throughout the day, depending on the temperature inside and also the number of people present. Another part of the HVAC system are the variable air volume (VAV) dampers, which are located inside the air ducts. They open or close in order to regulate the amount of air supplied or returned from the compartments. To reduce energy consumption, inside the cabins there are fan coil units (FCUs) that treat the air inside by recirculating it without supplying fresh air.

The heating, ventilation, and air conditioning system inside a ship not only needs to ensure comfort for the passengers, but it is also imperative that it works according to regulations for ship operation to be safe. Regulations state that cabin temperature must be between 22 and 25 degrees Celsius at all times, and the flow of fresh air inside the cabin must be at least 20 liters per second or 8 liters per person per second [12]. Not following the standards could lead to passenger harm.

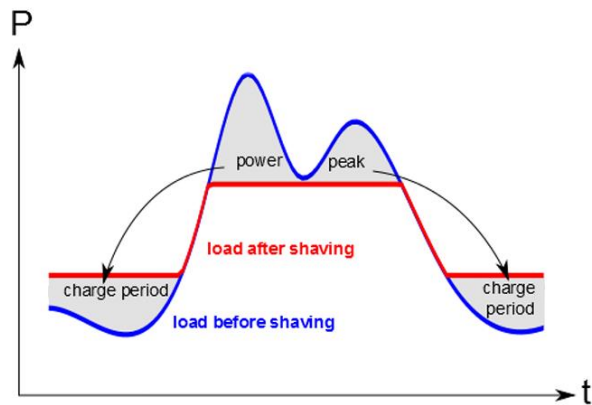
Aside from HVAC, other significant energy consumers of the hotel load are the lighting system and the entertainment system.

### **2.1.2 Energy storage system**

For effective operation, ships also have an energy storage system (ESS). The main roles of the ESS are peak shaving, spinning reserve, and zero emission operation [4]. During peak shaving, the batteries take the excess load off the grid from the generator. If the current load exceeds the maximum output of the generator, an additional generator would need to be started, leading to both generators running at suboptimal efficiency. The storage system can take care of the excess load, making sure the generator runs at optimal efficiency, which reduces fuel consumption. By reducing the number of active generators, the unused generators are used less, leading to lower maintenance costs.

During spinning reserve, the battery system acts as a backup. Cruise ships at sea prefer to have two generators running simultaneously, in case of a blackout by one of them. Restarting a generator is time-consuming, thus having a backup generator up and running is important for the safe operation of the ship. The battery can take the role of the backup generator and free one of the generators. Having only one generator running, increases fuel efficiency, leading to lower energy consumption and emissions.

Zero emission operation consists of using only the battery system while at sea, powering the ship fully electrically. This can be challenging due to the large storage capacity required for the battery. Also, it would be infeasible for large ships or long trips, as the size of the battery system would need to be, in theory, considerably larger than the size of the ship with current battery storage technology. For short trips, however, there are already available electric and hybrid ships that operate on zero or low emissions. Hull 096 is a 130 meter fully electric cruise ship capable of accommodating up to 2200 passengers and 225 vehicles, traveling from Argentina to

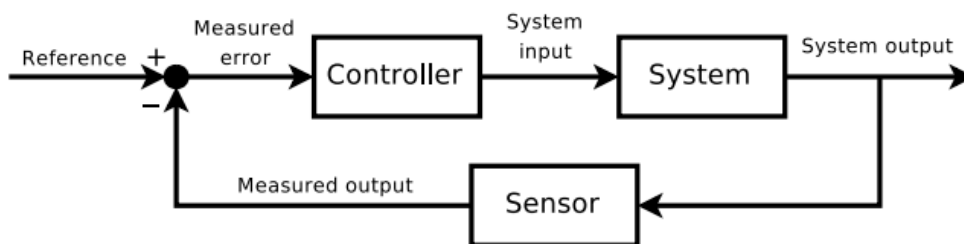


**Figure 2:** Principle behind peak shaving illustrated. The battery is discharged during peak hours and charged during periods of low consumption [13].

Uruguay in a short-distance cruise [14].

## 2.2 Control theory

Control theory deals with the control of dynamic systems. The desired output of a system is called a reference point or a setpoint. The goal of a controller is for the output variable of a system to follow the setpoint over time [15]. In order to accomplish this objective, the controller manipulates the inputs to the system until the desired effect is produced. There are different types of control strategies, with the two main types being open-loop control and closed-loop control, or feedback control. Open-loop control is the simplest type of control. The system input is fixed, and the controller does not actively look at how it is performing. It simply follows the predefined input. Closed-loop control relates to a closed system where the controller actively has feedback on its performance. In closed-loop control, the controller continuously measures its performance through time. To measure the performance of a controller, the error is calculated between the desired setpoint and the output of the system. The controller uses the error term to modify the input. A schematic view of a closed-loop control strategy can be seen in Figure 3.



**Figure 3:** A feedback loop, which controls the output of a dynamic system [15].

An example of a control system is a ship's energy system. To simplify the example, the system only includes a single cabin and a single controller that controls the fan coil unit. The temperature setpoint for the cabin is 23 degrees Celsius, whereas the actual temperature of the air as read from a sensor inside the cabin is 25 degrees. The error term in this case is 2 degrees. The difference would activate the controller of the FCU inside the cabin, which would move the air through a cold coil to decrease the temperature until the sensor reading matches the setpoint.

There are different types of closed-loop controllers depending on how they operate. The most common types of controllers are PID controllers, which use the proportional, integral, and derivative terms with respect to the error to treat the input to the system. Depending on which terms are present in the input calculation for the controller, it can take different names, which are a combination of the letters P, I, and D.

The HVAC system has a central controller that can be accessed from the control room. Through the control room, global setpoints such as temperature and humidity setpoints are set, which control the temperature and humidity of the air exiting the AHUs. The control room can also set inflow setpoints, directly controlling the amount of air that enters the compartments. Different control strategies for the central controller can be set. The simplest strategies would be manual control of the setpoints by a human operator or open-loop control with fixed setpoints. Other types of control strategies are using a PID controller, using the results of a traditional optimization algorithm, or using a reinforcement learning agent. The control room includes real-time readings of sensors inside the ship for the whole system, to aid the controller in making decisions.

Aside from the central controller, the HVAC system also includes multiple smaller controllers that affect the different parts of the system. A VAV controller can directly control the amount of air flowing through the cabins using its own controller and sensor data from the cabin, such as  $CO_2$  levels, humidity, room occupancy, and air temperature. Commonly, passengers can also set temperature setpoints from inside the cabins, which mainly affect the controller of the fan coil units.

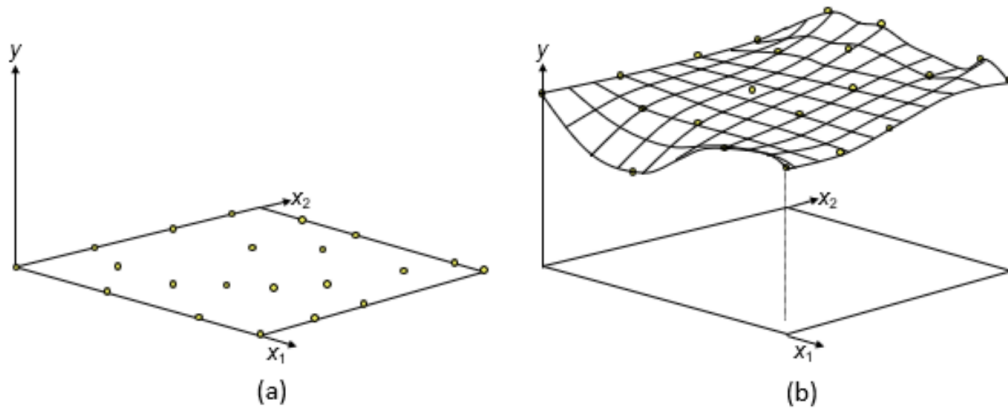
## 2.3 Surrogate models

In engineering, a common challenge is to model a particular quantity of interest (QoI) given a set of input values or parameters. For some phenomena, this can be done analytically. For example, the maximum stress of a beam can be calculated by using the cross-sectional area of the beam, its length, and the location and magnitude of the applied loads. However, for a large number of phenomena, there is no analytical formula you can use. The relationship between the phenomena and the input parameters is governed by complex physics equations and can only be computed using numerical methods [16].

Using numerical methods to simulate the QoI, leads to accurate, high-fidelity data. However, it is usually quite computationally expensive. While running one simulation can be feasible, many engineering challenges require multiple simulations. In engineering processes concerned with optimization, it is common to require multiple simulations in order to find the optimum set of parameters for a particular problem. For example, in design optimization, the challenge is to find a set of design

parameters that optimize a specific goal while the design complies with predefined constraints. Multiple design simulations must be computed while varying the input design parameters. Then, the objective function is computed for each design while making sure it also complies with the constraints [16]. The best-performing set of parameters that is consistent with the constraints is chosen.

A common practice is to develop a black-box approximation of the simulation, also known as a surrogate model [17]. The benefit of developing a surrogate model is a large reduction in computation time for calculating the QoI, in exchange for some loss in fidelity. The idea behind surrogate modeling is similar to curve fitting, as a multiple-dimensional surface is fitted across the input-output hyperplane, as can be seen in Figure 4. The hyperplane is made of the inputs, which are the parameters used for the simulation and the outputted quantities of interest. Thus, it tries to model the relationship between the input parameters and the QoI directly, without relying on numerical methods.



**Figure 4:** Example of parameters in a 2-dimensional domain (a) and the corresponding surrogate model (b) [16].

In order for the surrogate model to be accurate, a sample of initial simulations that spans the hyperplane is needed. These simulations are used as data points for the surface fitting that will form the surrogate model. The surrogate model works by interpolating between the sample of simulations it was fitted on. For a specific set of inputs, the surrogate model outputs the corresponding QoI based on the value of the surface.

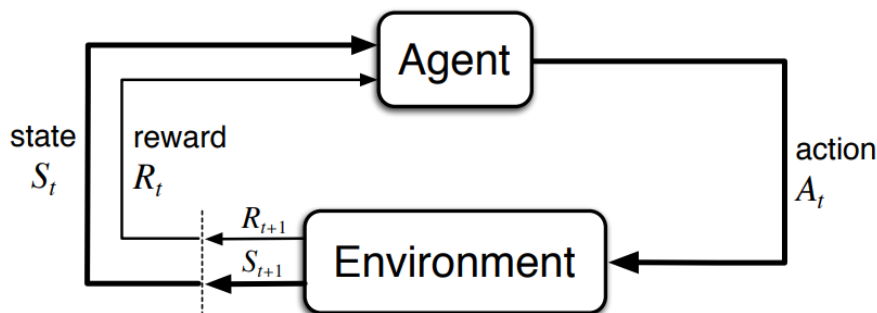
As curve fitting is a common problem in data approximation, many tools can be utilized in order to develop the surrogate model, such as regression models, machine learning models, or neural networks.

## 2.4 Reinforcement learning

Reinforcement learning encompasses a class of solution methods that are used in order to solve reinforcement learning problems. Reinforcement learning (RL) problems involve an agent learning to navigate or accomplish a particular task in an environment

by maximizing a reward signal. There are three main characteristics that RL problems pose, which distinguish them from other machine learning problems according to Sutton and Barto [6]. Firstly, the problems are closed-loop in nature, as each action the agent takes influences the following state it will land on. Moreover, reinforcement learning problems are not supervised problems, and as such, the agent is not told how to act, but must learn on its own. The final characteristic is the ability of the actions chosen by the agent to affect not only the immediate reward but also rewards the agent will receive later on.

In a reinforcement learning problem, the learner is defined as the agent, whereas the environment is everything outside the agent, the agent interacts with. The position of the agent with respect to the environment in a specific timestep is defined as the state of the agent. To move from a state in a timestep to another one in the next timestep, the agent performs an action. Both the state and the action can be discrete or continuous, single values or multi-dimensional, depending on the complexity of the problem. The interaction between the agent and the environment, from an initial state till a goal state or till a maximum number of allowed timesteps, is defined as an episode. During the duration of an episode, the agent learns to interact with the environment as part of a closed system, where the agent is the controller, the environment is the controlled system, and the reward specifies the feedback it receives. This control loop can be seen in Figure 5.



**Figure 5:** The interaction of the agent with the environment in reinforcement learning [6].

Aside from the agent and the environment, there are four main parts of a reinforcement learning problem:

- A policy
- A reward signal
- A value function
- Optionally, a model of the environment

The policy determines the decision making of the agent. In its essence, the policy is a mapping from a state to an action. The agent follows the policy in order to decide

which action to take in a particular state. The policy can be either deterministic or stochastic depending on the challenge the agent is trying to solve. Deterministic policies always map to the same action, whereas stochastic policies sample from a probability distribution. Depending on how complex the RL problem is, the policy can range from a simple lookup table to a deep neural network.

The reward signal or the reward function serves the role of defining the goal of the RL problem and also giving feedback to the agent about how it is performing. After each timestep, the agent receives a reward, which is a continuous value defined by the reward function. The goal of the agent is to maximize the sum of the rewards over the span of an episode. The reward influences the policy. If an action chosen by the policy receives a low or a negative reward, the policy may be updated to better follow the reward.

Similarly to the policy, the reward function can be defined as a mapping, but from a state-action pair to a reward value. This mapping can be either deterministic or stochastic. Unlike the policy, the reward function is not learned through interaction but must be defined by the RL engineer. Defining a reward signal is not always straightforward. In problems where the reward is sparse throughout the timesteps or the goal is too hard to learn for the agent, intermediate goals must be defined. Defining a good reward function takes experience, and it is one of the most important parts of solving a reinforcement learning problem.

The value function specifies how good a particular state is. It is similar to the reward signal as they both provide feedback on how good a particular policy is. However, the reward signal only quantifies the immediate reward from a specific state-action pair, while the value function approximates the long-term usefulness of a particular state. For example, while a state might give low rewards, it might be followed by a state that gives large rewards, and thus it has a large value. Following large value states leads to having a higher cumulative reward at the end of an episode. The value function maps from a state to a value. It is learned from interacting with the environment, and it is shaped by the rewards seen.

Some reinforcement learning methods, also called model-based, include a model of the environment. The role of the model is to learn the dynamics and reward function of the environment while the agent is interacting with it. The model is then used in order to plan for the future over a longer time horizon.

### **2.4.1 Value-based vs policy-based methods**

There is a large amount of research in developing reinforcement learning methods, and there are many alternatives available. One important aspect in which reinforcement learning methods differ is whether they are value-based or policy-based.

The main focus of value-based methods is learning a good value function. The policy is explicitly derived from the value function by always choosing the action that would lead to a higher value. As such, these methods are always deterministic. Some of the most common value-based algorithms are: Q-learning and Deep Q-Networks (DQN) [18]. The methods are simple and efficient, especially in smaller discrete problems. However, they usually fail at continuous or high-dimensional problems.

Policy-based methods, on the other hand, do not have a value function, instead, they learn the policy directly. The policy is represented by a parametrized distribution, and the agent tries to learn the optimal parameters to maximize the cumulative reward. For example, for a continuous problem, the policy could be a normal distribution with parameters  $\mu$  and  $\sigma$ . The parameters are learned by the agent and can be outputted by a neural network, whereas the action is sampled from the distribution  $\sim \mathcal{N}(\mu, \sigma^2)$ . As the actions are sampled from a distribution, the policy can be stochastic.

The most common pure policy-based method is REINFORCE. The main idea behind REINFORCE is to shift the policy distribution into regions with higher returns gradually. As it is an on-policy method, the current policy is used to collect experience from interactions with the environment in the form of actions, action probability densities, rewards, and states. Action probability densities represent the probability of selecting the action from the current policy distribution. These methods are also called policy gradient methods, as after a certain number of interactions with the environment, they calculate the gradient of the policy with respect to the policy parameters. The policy parameters are updated through backpropagation, similarly to traditional neural network training, where the policy gradient affects all parameters of the network. One drawback of policy gradient methods is that they may require a large number of updates or gradient passes, leading to slow convergence.

Actor-critic methods combine ideas from both policy-based and value-based methods. The critic is a value function that measures the performance of the agent and is learned during training, while the actor represents the policy. Actor-critic methods improve upon policy gradient methods as the inclusion of the critic reduces the variance of the policy gradient, leading to faster convergence. As such, they need a lower amount of updates than pure policy-based methods to achieve good performance. Several RL algorithms use the actor-critic structure. One of the most common, which will also be explored in this thesis, is Proximal Policy Gradient (PPO) [19].

## 2.4.2 Proximal Policy Gradient

The main idea behind PPO is to limit the policy gradient in order to prevent large changes from the old policy ( $\pi_{\theta_k}$ ) to the new one ( $\pi_{\theta_{k+1}}$ ). The current policy parameters  $\theta_k$ , are updated through multiple steps of mini-batches by an optimizer as follows:

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

where  $\theta_{k+1}$  represents the parameters of the updated or new policy, by maximizing the estimate of the particular loss function  $L$  used. The loss function takes as input the parameters of the current policy and trajectory data obtained by following the current policy, such as actions ( $a$ ) and states ( $s$ ). The PPO loss function is made up of multiple terms. The most important term is the surrogate loss as seen in Equation 7, which updates the policy. Other terms include the value loss, which updates the critic. The terms in the loss function are weighted. The weights are hyperparameters and are chosen by the RL engineer.

The main idea behind PPO involves clipping the surrogate loss. The unclipped surrogate loss is defined as:

$$L_{unclipped} = \left( r_t(\theta) \hat{A}(s, a) \right) \quad (2)$$

where the probability ratio  $r_t(\theta)$  is the ratio of the probability density between the distribution defined by the new policy and the one defined by the current policy for the sampled actions:

$$r_t(\theta) = \frac{\pi_{\theta_{k+1}}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \quad (3)$$

The advantage function  $\hat{A}(s, a)$  estimates the difference between the expected reward from the environment given the actions chosen with the average expected reward given all possible action choices for a particular state. It can be positive or negative depending on whether an action is generally better or worse than the average action the agent could choose. Practically, the advantage function is approximated using generalized advantage estimation (GAE) as defined in [20]. Firstly, the temporal difference TD residual ( $\delta_t$ ) for timestep (t) is computed as follows:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (4)$$

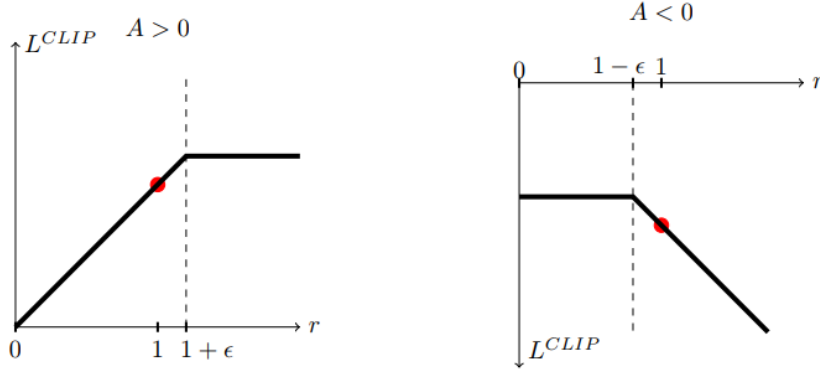
given the immediate reward, a discount factor ( $\gamma$ ), the value estimate of the current state, and the following state. The temporal difference error evaluates how much better or worse a particular action was given the expectations of the agent. Then the GAE for a timestep t is derived as follows:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l} \quad (5)$$

by summing over all future TD errors discounted by a discount factor  $\gamma$  and a hyperparameter  $\lambda$  which adjusts the bias-variance tradeoff. The GAE is similar to a discounted sum of rewards, however, it is much more effective at lowering the variance of the advantage estimate, leading to more stable training and better performance [20].

From Equation 2, the surrogate loss is defined as the probability ratio times the advantage function. The advantage function dictates the direction of the gradient. If the advantage is positive, the chosen action was beneficial, and the parameters should be updated to reflect that. The ratio of probabilities measures how much more or less likely an action is for the current policy given the previous policy where the data was collected. A ratio greater than 1 shows that the action is now more likely than before, whereas a ratio smaller than 1 shows the opposite. Whereas the advantage controlled the direction of the policy gradient, the ratio controls the magnitude of the policy gradient. A ratio much larger than 1 leads to a large change in policy parameters. This could lead to instability during training. This is why in PPO the probability ratio is clipped in between a range  $[1 - \epsilon, 1 + \epsilon]$  with  $\epsilon$  being a small constant usually 0.2.

$$L_{clipped} = \left( \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}(s, a) \right) \quad (6)$$



**Figure 6:** The clipped surrogate loss plotted over the probability ratio. The loss is clipped from growing too large regardless of the direction of the advantage [19].

The clipping is done to prevent the new policy from updating too far from the current policy. The gradient steps become smaller, leading to more stable training. The policy parameters are always updated in proximity to their current values, thus the name Proximal Policy Optimization. The clipped loss is illustrated in Figure 6. The final surrogate loss is defined as:

$$L_{surrogate} = \min(L_{unclipped}, L_{clipped}) \quad (7)$$

Aside from regular PPO, several algorithms build on the idea. One such algorithm is Penalized PPO (P3O) [21]. Penalized PPO is useful for constrained optimization problems where, aside from the objective of the RL, there are also obstacles or constraints that it should avoid. P3O performs better at satisfying constraints than PPO, as shown in [21].

This is achieved by deconstructing the reward function into a reward and a cost. Regular implementations of PPO for solving constrained problems implement the constraints as part of the reward function. As such, both the objective reward and constraint penalty are collapsed into a single scalar. P3O decouples them into a reward and cost function, which is then used to calculate an additional cost loss term.

P3O has an additional network, the cost-value function, which is similar to the critic but it approximates the discounted cost, not reward. The derivation of the cost loss for P3O is identical to the surrogate loss for PPO, with the only change being that the cost advantage ( $\hat{A}_C$ ) is used instead of the reward advantage.

$$L_{cost} = \min \left[ \left( r_t(\theta) \hat{A}_C(s, a) \right), \left( \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_C(s, a) \right) \right] \quad (8)$$

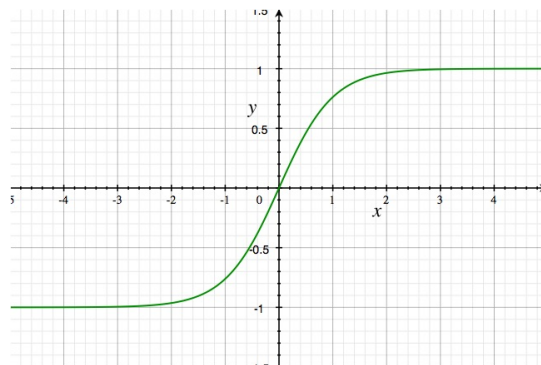
### 2.4.3 Bounding for continuous action spaces

In reinforcement learning problems, the action space can be either continuous or discrete, depending on the needs of the problem. In a discrete action space, the agent selects between discrete action choices, whereas in a continuous action space, the agent selects a real value for the actions. Problems that require a continuous action

space may need to be bounded, thus only accepting action values from a specific range. There are multiple reasons why the agent may benefit from bounding the action space. Firstly, bounding the action space makes it easier for the agent to explore since it does not need to explore an unbounded number of possibilities. Furthermore, extreme action values can lead to extreme or unpredictable environment states, which may confuse the agent and hinder training. Another reason is that the physical system, which is modeled by the RL environment, can have physical bounds on the actions possible. For example, an actuator can be bounded on its range of motion.

PPO agents commonly utilize a neural network for the actor that chooses the policy parameters. Basic neural networks are unbounded in their outputs of policy parameters. Thus, the actions sampled from the policy distribution are also unbounded. To bound the action outputs of the agent, two strategies are usually employed: clipping and applying a squashing function such as the hyperbolic tangent (tanh) [22]. While clipping is effective in preventing extreme actions, it makes learning for the agent harder. All values outside the clipping range are clipped to either the maximum or the minimum of the range, making it impossible for the agent to differentiate between them. For example, if the clipping range is  $[0, 10]$ , both extreme values, such as 1 million, and also values slightly outside the range, such as 10.1, would give the same feedback to the agent, hindering the learning process.

The second technique utilizes a tanh function to squash the actions in the  $[-1, 1]$  range, which are then transformed to the desired range.



**Figure 7:** Tanh activation function.

The place where the tanh transformation occurs is important. Applying a tanh transformation as the last layer of the actor network can lead to numerical issues. Tanh activation function and similar activation functions that squash the output space, such as the sigmoid, are saturating activation functions. If the input to the activation function is above or below a certain saturation point, the result will collapse to either 1 or -1. This is due to numerical constraints in representing floating point numbers with multiple digits after the decimal such as  $0.999...9$ . As such the result of the tanh activation function is collapsed to a horizontal line after the saturation point. The gradient of the neural network for a horizontal line would be zero or vanish. As the gradient is backpropagated through the network, none of the parameters are changed, leading the training to a standstill. The gradient vanishing makes it impossible for

the agent to learn whether the action choice was good or bad. Thus the agent stops learning and produces constant outputs for the action.

A better approach is to apply the tanh activation function after the action has been sampled. Thus, the transformation does not affect the gradient of the neural network. Applying a transformation function to a random variable, in this case the action, changes the probability density function (PDF) of the action [23]. As such, the probability density of the selected action must be corrected. The general formula for computing the PDF of a transformed random variable is:

$$f_Y(y) = F'_Y(y) = \frac{d}{dy}F_X(g^{-1}(y)) = f_X(g^{-1}(y)) \frac{d}{dy}g^{-1}(y). \quad (9)$$

where  $(g)$  is the transformation function and  $(y)$  the transformed variable, and  $f_Y(y)$  is the PDF of the transformed action. For a tanh transformation function and a Gaussian distribution, the PDF of the bounded action  $(a)$  is:

$$f_Y(a) = \mathcal{N}(atanh(a); \mu, \sigma^2) \frac{1}{1 - a^2}. \quad (10)$$

#### 2.4.4 Shielding for safe reinforcement learning

Safe reinforcement learning is a subfield of reinforcement learning concerned with optimal policy learning while complying with safety constraints. One common safe RL method is employing a shield during the interaction of the RL agent with the environment. The purpose of the shield is to block actions that the shield thinks would break safety constraints. A probabilistic safety shield is introduced in [7]. The shield blocks actions that are more likely to violate constraints than a specific threshold. Another work further develops the concept of safety shield into online safety shields [8]. Offline safety shields compute the probability of every combination in the state-action space preemptively, which results in large computational costs. Online shields remove the preprocessing overhead by computing the probability of safety violations occurring in real time. For every action, the shield computes the probability of not violating the safety specifications within a certain number of steps in the future. A key takeaway from the paper is that the safety shield performs best when it is only engaged during inference and not during training. Blocking unsafe actions during training stops the agent from learning the negative feedback it would learn from interacting with these states.

## 2.5 Large language models

Language models are statistical models that predict the next word given a specific context. Modern language models are based on neural network architectures and are trained by feeding on a large amount of textual data. One such architecture, Transformers, are especially good at learning from sequential data such as textual data. These models learn from the connections between words in their training corpora. Increasing the size of the training corpora has led to these models becoming

increasingly good at understanding abstract concepts from their training data and emerging new intellectual abilities [24]. Due to the sheer size of modern language models, the research community refers to them as large language models. These models especially excel at generation by predicting what will follow an incomplete part of text. The partial text can be a prompt, such as a question or a request, alongside background information. The model then completes the prompt by what it thinks should follow it, thus answering the question or the request. This idea has been used in different applications from chatbots to code generation [24].

## 2.6 System-theoretic process analysis

System-Theoretic Process Analysis (STPA) is a hazard analysis method used for validating the safety of a system. STPA consists of 4 main parts or steps, that can be seen illustrated on Figure 8 [25]. Each step consists of smaller tasks and usually produces an outcome.

The first step relates to defining the purpose of the analysis. It is concerned with the types of losses the system will aim to prevent. The second step models the control structure of the system as a set of feedback control loops between the controllers and the controlled entities. The third step is to analyze the control actions and how they could lead to the losses defined from the first step. The unsafe control actions form the basis of the safety and controller constraints of the system.

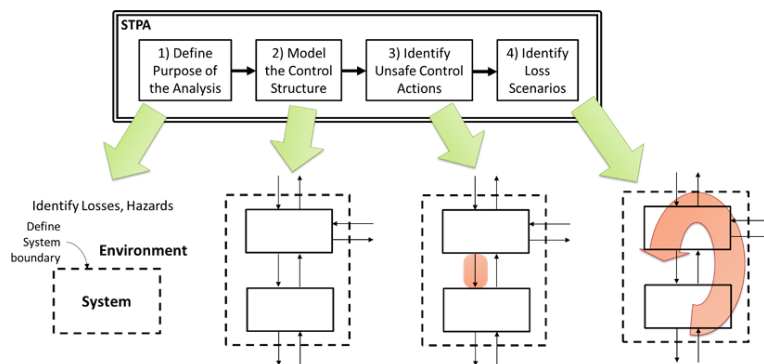


Figure 2.1: Overview of the basic STPA Method

**Figure 8:** Overview of the STPA method [25].

Safety constraints define rules that the system must abide by in order to prevent unsafe or hazards states. For example a safety constraint can state that cabin temperature must not exceed 25 degrees Celsius. Controller constraints are similar, although they also place limits on the controller. As an example, a controller constraint may state that not only does cabin temperature need to be below a certain temperature, but if that is the case the controller must give a temperature setpoint that is lower.

The fourth step identifies why unsafe control actions might occur in the system.

## 2.7 Relevant work in ship energy optimization

There is existing prior research on the topic of ship energy optimization. Traditional optimization techniques such as linear programming or genetic algorithms (GA)s are commonly employed to the task at hand and have shown to be effective. In one paper, researchers demonstrated using Non-dominated Sorting Genetic Algorithm (NSGA-II) for propulsion control under severe sea conditions [26]. Their results show improvements in energy consumption with a 21.9% decrease in fuel consumption per nautical mile. Another paper also utilizes NSGA-II for optimizing a hybrid energy ship's power system [27]. They report an improvement in the Energy Efficiency Operational Indicator (EEOI), which measures  $CO_2$  emissions per unit of work, of 17.53%. While effective for some scenarios, traditional optimization methods have drawbacks. They do not scale well to large and complex environments or stochastic environments. A ship's energy system is large and complex, with many uncertainties that affect it, such as passenger behaviour and weather conditions. As such, more novel methods have been employed to the task of ship energy management.

Reinforcement learning methods are not limited by the size of the search space. They are also able to deal with real-time challenges and unpredictable scenarios. Existing literature explores the use of reinforcement in the optimization of ship energy consumption and greenhouse gas emissions. In [5] the authors employ a reinforcement learning algorithm, namely Proximal Policy Optimization (PPO), and an expert-crafted reward function for their agent. The agent interacts with a simulation of a ship energy system. Experiments involved comparing RL performance with traditional approaches for 5 different types of vessels: cruise ships, car carriers, oil tankers, bulk carriers, and container ships. The authors report that they were able to achieve significantly better results than traditional optimization methods, such as linear programming and genetic algorithms. In another paper, authors analyze the effectiveness of different deep RL algorithms for green ship energy management [28]. They compare three different RL algorithms: a deep deterministic policy gradient (DDPG) algorithm, a PPO algorithm and a deep Q network (DQN) algorithm. Although existing literature highlights the effectiveness of reinforcement learning for ship energy optimization, there is a lack of focus on safety.

The research in this thesis differs from the existing literature in RL for ship energy optimization by focusing on safe RL approaches. An LLM-based reward function generator was used in order to generate a safe RL reward function, and a safety shield was employed to block unsafe actions from the agent.

## 2.8 Relevant work in large language models for reward function generation

There is a large and growing amount of research related to Large language model generation. Here, two state-of-the-art methods that utilize LLM generation for reward shaping are introduced.

Text2Reward is an algorithm that, given a goal written in natural language, can produce interpretable, dense reward codes written in Python [9]. It utilizes a goal

in natural language and a Pythonic representation of the environment to generate a reward function. To make sure that the reward generated by the LLM is consistent with the goal, it utilizes human feedback in order to evaluate the reward function. The human operator observes how the agent moves in the environment and gives feedback accordingly to push the agent in the direction of the stated goal.

“Language to reward” is a similar paper that achieves LLM-based reward shaping [10]. Their method relies on turning a natural language objective first into an intermediate motion description of the intended motion and then encoding the description into a low-level code for the agent to follow. This method also relies on human feedback to evaluate and optimize the reward function generated.

Both papers describe algorithms that are quite similar to what was developed in this thesis, and they were used as a basis. This thesis differs from the aforementioned papers because the scope follows a ship energy management system environment, which has not been researched before. Both introduced papers are tested on small, tight-knit control environments with few parts, and they can be described fully in natural language or in a Pythonic manner. A ship’s energy system environment is more complex, with many more controllable parts and complex underlying physical equations mapping control actions to changes in the environment. Thus, correctly describing the environment and condensing it into a system prompt will be part of the novelty of the research.

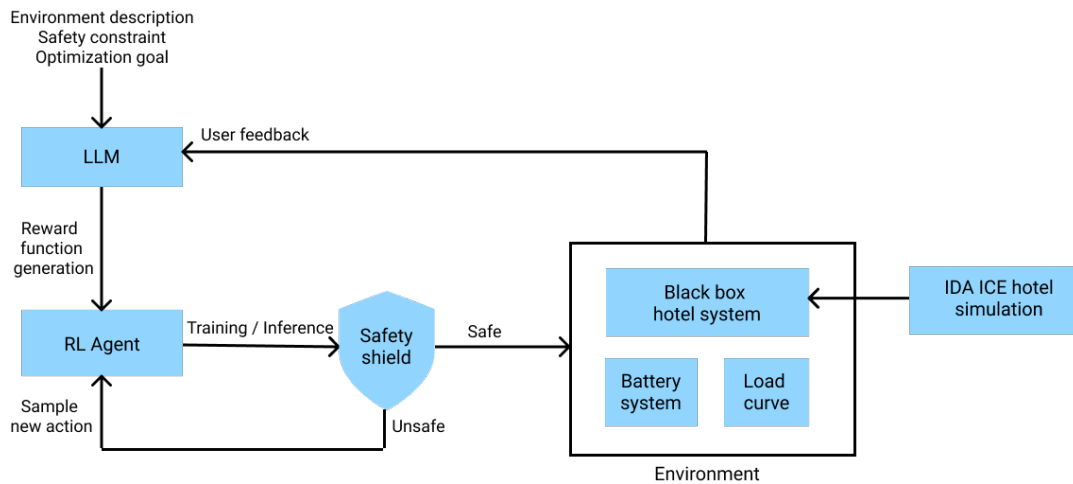
### 3 Research methods and implementation

This chapter describes the methodology of the thesis and the experiments that were conducted.

#### 3.1 Overview of the methodology

The methodology of the thesis relates to the development and testing of a safe reinforcement learning controller system for a cruise ship environment. As part of the methodology, several parts were developed and integrated. Figure 9 demonstrates the integration of the different parts. The full list of parts that were developed are:

- A white-box simulation of a cruise ship’s energy system, which consists of: consumption, storage, and generation models
- A black-box surrogate model that approximates the white-box model and is computationally faster
- A reinforcement learning training environment
- A reinforcement learning agent
- A large language model reward function generator (RFG)
- A safety shield



**Figure 9:** Methodology of the work presented in this thesis [29]

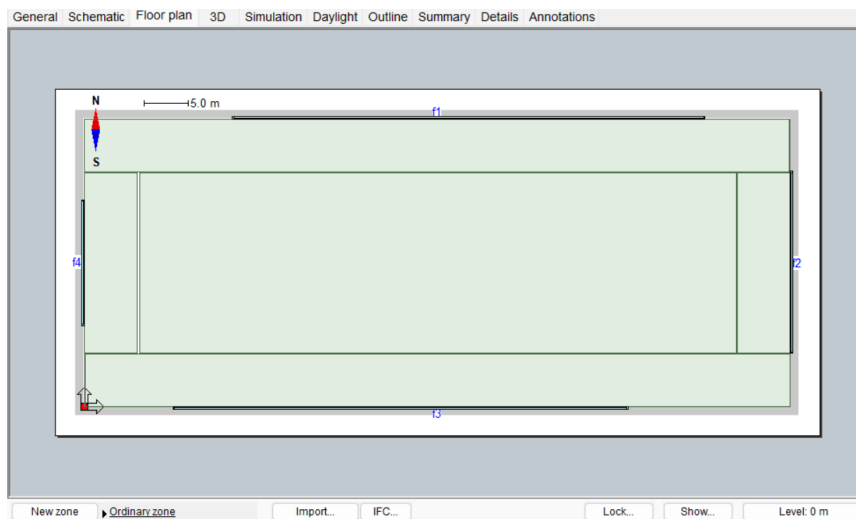
The environment in which the system will run is a simulation of a ship’s energy microgrid. To have an accurate representation of a real-world ship, a white box physical simulation was developed. The simulation was developed in IDA ICE, a building simulator software, and models the physical properties of a ship’s hotel system, such as HVAC, lighting, weather patterns, passenger movement, and energy consumption.

Alongside the hotel simulation, a battery model was used in order model energy storage, and a load curve was used in order to model generation.

Although the white-box simulation model is accurate, it is computationally expensive to run. A black-box model was trained to learn and approximate the physical phenomena modeled by the white-box model. The black-box model is significantly faster to run and was used to train a reinforcement learning agent on top of it. Before the agent starts training, a reward function is generated by the reward function generator (RFG). The RFG takes as a prompt a description of the environment and problem at hand, the optimization objective, and a list of safety constraints. The reward function is then integrated into the agent. While the agent is training, the safety shield keeps track of safety violations. During inference, the safety shield blocks actions that lead to unsafe states, resulting in a safer policy. After the agent has finished training, performance metrics and natural language feedback are added to the reward function generator in order to develop a better reward function. The iterative process goes on until the RFG is unable to develop a better reward function.

### 3.2 Simulation

The RL system requires a cruise ship environment to be available in order to train and test the RL agent. In this thesis, the IDA ICE simulation software, developed by EQUA Simulation AB, was used to simulate the hotel system of a cruise ship, the HVAC system, and the lighting system [30]. The IDA ICE simulation tool is focused on simulating buildings, and as such, there are several limitations to using it. While there are many similarities between a real hotel and the hotel section of a cruise ship, this thesis acknowledges two main drawbacks of utilizing this tool for the cruise ship hotel simulation.



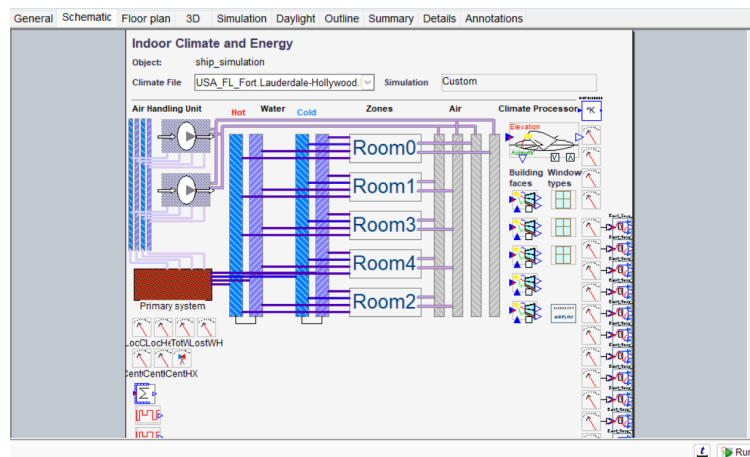
**Figure 10:** Aerial view of the simulated cabins. The simulation model includes 4 cabins and the hallway between them.

Firstly, changes in weather patterns due to the movement of the ship, from one

location to another, cannot be simulated in IDA ICE as a building is fixed in one location. Secondly, the simulated hotel cannot be rotated, as a building is fixed in place. In a real ship, the rotation affects the amount and location of radiation the ship receives.

Simulating the whole ship system and every compartment inside would be overly complex and redundant. It is common in engineering solutions to simulate a representative part of the system from which it can be inferred about the system as a whole. The simulation model used in this thesis includes 4 cabins and the hallway in between, as shown in Figure 10. The model is based on an already available building model, modified to fit the context of a cruise ship [31]. The power consumption is multiplied in order to approximate the energy consumption of a mid-size reference cruise ship with 300 cabins.

The simulation includes a comprehensive HVAC system. A schematic view of the HVAC system can be seen in Figure 11. There are two air handling units, one connected to the cabins and one to the hallway. Both AHUs are connected to an electric chiller, which provides cold liquid to the coils, and a boiler, which burns fuel in order to heat the coil liquid. Each compartment has a VAV system, which controls the amount of air that flows inside and is controlled by a central controller in the engine room. Each compartment has its own fan coil unit, which treats the air by recirculating it and is controlled by temperature and humidity setpoints in the cabins. There are sensors inside each compartment that are connected to the engine room, and they provide information regarding air temperature, CO<sub>2</sub> levels, and humidity inside the compartment.



**Figure 11:** The HVAC system as part of the simulation model.

Human activity is also simulated. The two smaller cabins on the left and right, as seen in Figure 10 accommodate 2 people, while the cabins on the front and back accommodate 4 people. The people freely move between their cabin and the hallway, and their activity is defined by a set schedule. Each of the five compartments also includes lights, which make up the lighting system and electronics, which make up the entertainment system. The consumption for both the lighting and entertainment systems throughout the day is based on the activity level of the passengers.

The simulation software makes it possible to change different parameters in the cabins or the HVAC system, and run simulations. As part of the simulation process, a specific location and a weather file must be chosen, as well as a time frame for the simulation. During the simulation, energy meters keep track of the energy consumption of different parts such as the chiller, the boiler, the AHU fans, the FCU fans, the pumps, the lights and the entertainment system.

In order to communicate with the IDA ICE simulation software, the API provided by the same company makes it possible to set specific inputs and to get outputs from each simulation.

The inputs that will be set during operation setting are the control inputs, parameters the RL agent should have access to in order to optimize the control of the EMS such as: temperature, humidity and airflow setpoints. On the other hand the outputs that are extracted from each simulation are energy consumption data from the energy meters, sensor data from each compartment such as temperature, humidity and CO<sub>2</sub> levels, and weather information such as outside temperature and humidity. The inputs are single values while the outputs are timeseries, with a timestep of 6 minutes between each data point. Tables 1 to 4 give a full list of the inputs and outputs for each simulation.

<b>Input parameters</b>	<b>Unit</b>	<b>Explanation</b>
AHU temperature setpoint	°C	Control setpoint for the temperature of the air that will leave both AHUs.
AHU humidity setpoint	%	Control setpoint for the humidity of the air that will leave both AHUs.
Compartment heating setpoint	°C	Control setpoint for the minimum temperature of the air inside of a compartment before heating takes place.
Compartment cooling setpoint	°C	Control setpoint for the maximum temperature of the air inside of a compartment before cooling takes place.
Compartment inflow	$L/(m^2 * s)$	Control setpoint for the inflow inside each compartment in liters per meter squared per second. The absolute inflow depends on the size of the compartment.

**Table 1:** Simulation control input parameters

<b>Output data</b>	<b>Unit</b>	<b>Explanation</b>
Cooling	Watts	Chiller power
Heating	Watts	Boiler power
Conditioner Fans	Watts	The power of the AHU fans, which are responsible for moving air in and out of a compartment
Fans	Watts	FCU fans power
Pumps	Watts	The power of the pumps which move coil liquid from the chiller and the boiler to the AHU
Hvax Auxiliary	Watts	Power of auxiliary processes of the HVAC system, for example, humidifying/dehumidifying.

**Table 2:** Simulation output consumption data for a single timestep

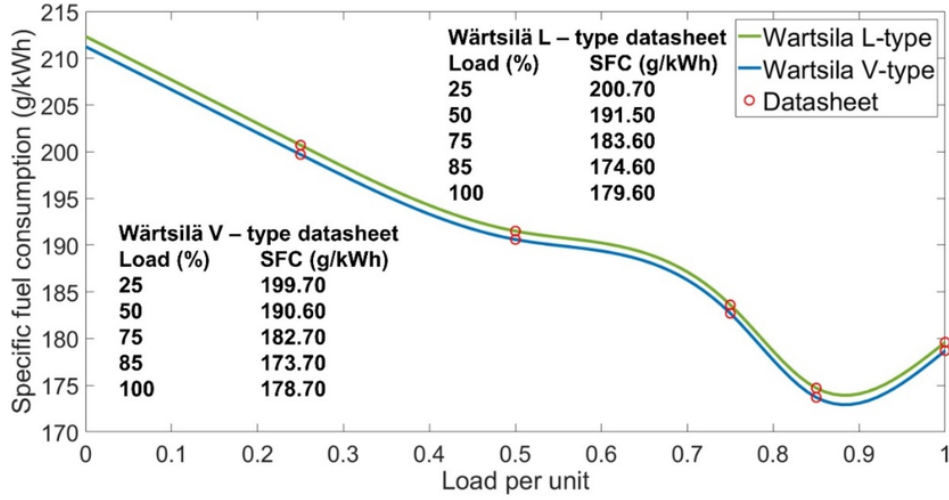
<b>Compartment data</b>	<b>Unit</b>	<b>Explanation</b>
Air temperature	°C	Average air temperature
Air humidity	%	Average air humidity
CO <sub>2</sub> levels	parts per million (ppm)	CO <sub>2</sub> concentration of the air
Air age	hours	Average amount of time since air particles have entered a compartment

**Table 3:** Simulation output sensor data for one compartment for a single timestep.

<b>Weather data</b>	<b>Unit</b>	<b>Explanation</b>
Dry bulb air temperature	°C	Ambient temperature.
Humidity	%	Relative outside air humidity
Direct solar radiation for each compartment	W/m <sup>2</sup>	Energy emitted directly from the sun. Measured for each compartment
Diffuse solar radiation for each compartment	W/m <sup>2</sup>	Energy emitted from the sun and scattered by the atmosphere. Measured for each compartment.
Wind speed	m/s	Average wind speed
Wind direction	Degrees	The direction of the wind with 0° indicating wind blowing from north towards south.

**Table 4:** Real weather data for the Helsinki region used by the simulation.





**Figure 13:** Specific fuel consumption (SFC) curve of Wärtsilä 46F V and L type marine engines. The curve maps fuel consumption with engine load [33].

$$\text{SFC}(p) = \begin{cases} -1.0272p^3 - 46.336p + 212.3, & p \in [0, 0.25] \\ 158.74p^3 - 119.82p^2 - 16.38p + 209.8, & p \in (0.25, 0.50] \\ -704.32p^3 + 1174.8p^2 - 663.67p + 317.69, & p \in (0.50, 0.75] \\ 4106.5p^3 - 9649.6p^2 + 7454.6p - 1711.9, & p \in (0.75, 0.85] \\ -1826.6p^3 + 5479.9p^2 - 5405.5p + 1931.8, & p \in (0.85, 1] \end{cases} \quad (11)$$

where  $p$  is the load of the generator.

The generation model simulates a single generator being always in use. The start-up phase of the generator is not taken into account.

### 3.3 Black box surrogate model

The IDA ICE simulation software can be time-consuming in producing a simulation. This is particularly true in the case of using it to train an RL agent. The control inputs can only be set once before the start of the simulation. Thus, the only way for the RL agent to interact with the simulation model, by changing the control inputs, is to run a new simulation at each time step. However, simulating even a short interval of 6 minutes takes several seconds of computing time, as there is preprocessing involved. For the use case described in this thesis, each training episode would require at least hundreds of timesteps, making training on the simulation model unfeasible.

In order to solve this drawback, this thesis develops a black box surrogate model of the IDA ICE simulation. The surrogate model is an approximation of the real simulation. While some fidelity is lost in the surrogate model it is much faster to run, making it possible to use for reinforcement learning training.

At first, a sample of simulations was generated from the IDA ICE simulation model. The sample includes simulations with different control inputs. The choices for

the control inputs were chosen so that they span across the domain of possible values that could be seen during RL training. Two strategies were conducted for selecting the control inputs. First is a grid search where each input gets its value from a selected list of possible inputs. IDA ICE also gives the possibility to select a daily schedule for the cabin temperature and inflow setpoints. A schedule makes it so the temperature setpoint changes during the day depending on the predefined setpoint for every hour. For the second strategy, a schedule was selected for the compartment temperature setpoints and the compartment inflow setpoint.

After the simulations finished running, the ones that had crashed were removed from the sample, along with the ones that had extreme values. The final sample consists of 143 simulations, each simulating a year worth of energy consumption for a hotel system. This sample makes up the dataset used to train and test the surrogate model.

Each of the simulations can be thought of as a table or a multidimensional timeseries. Each simulation consists of 87600 timesteps or rows of data and 70 features or columns. All of the features together represent the state of the system for each time step. Each feature represents a different aspect of the simulation such as: weather conditions, time of day, current control action, current sensor data reading for each compartment, current electrical power of each system.

The purpose of the surrogate model is to learn the relationship between input features such as as time of day, weather conditions and control actions and output quantities of interest such as electrical power usage and sensor data reading for a specific timestep. In order to train the surrogate model the dataset was divided into a training dataset comprised of 80 % of the simulations and a testing dataset.

To develop the surrogate model two different strategies were explored. The first one was developing an auto-regressive forecasting model and the second was developing a traditional surrogate model. Going forward the model developed using the first approach will be refereed to as the auto-regressive model or first model and the model developed using the traditional surrogate approach will be refereed to as simply the surrogate model or second model.

### **3.3.1 Forecasting model approach**

The initial idea for building a surrogate model was to use an auto-regressive approach. The model was fed the previous timestep or previous set of timesteps and input features from the current timestep to calculate the value for the output quantities of interests of the current timestep. As the model trains, it uses past predictions to predict new ones, and as such, it is auto-regressive.

The strength of this particular approach is the ability to have inertia. Each output is affected by the prior state of the system, thus it retains temporal information similar to a real system. The main weakness of this approach is that it is complex. Forecasting data, especially in multiple dimensions, is hard and not always accurate.

The structure of the model is as follows. A neural network was trained on the training dataset while the mean squared error (MSE) was used as a loss between the outputs of the model and the real output values from the simulations. The neural

network is made of an LSTM layer, which distills information from a set of previous timesteps, in this case 50 previous timesteps. The output of the LSTM layer is averaged over the temporal dimension and concatenated with the input features of the current timestep. The architecture is then followed by a rectified linear unit (ReLU) activation function and a linear layer. The inputs to the model were standardized by reducing the mean and dividing by the standard deviation.

In order to train the model a GPU was used. Auto-regressive models are sequential as each output depends on the output of the previous timestep. As such they fail to take advantage of the parallelism performance boost offered by the GPU. The training of the model was modified in order to take opportunity of parallelism. Batches were constructed by combining data points that were shifted by a constant  $x$  from each other. As such each batch contained data that immediately followed the previous batch in time. During training, a tensor is used to keep track of the output of the model for the previous set of batches. Figure 14 gives a representation of how a the training data was structured.

Batch 1	Batch 2	...		
0	1	...	49	10000
50	51	...	99	10050
100	101	...	149	10100
...	...	...	...	...
9900	9901	...	9949	19900
9950	9951	...	9999	19950

**Figure 14:** The structure of the training dataset for the auto-regressive model. Each column represents a batch and each point represents the location of the data point in the timeseries.

### 3.3.2 Surrogate model approach

The second approach follows a traditional surrogate modeling approach. The model interpolates between the different simulations in its training dataset in order to output predictions for a new set of inputs.

The strength of this approach is that is simpler and widely used in engineering applications. However, during training the time series data is sliced and the model can only view the data one slice at a time. As such, it has less temporal awareness than the first approach.

Different hyperparameters were tweaked, such as the model's depth, width, or the loss used to develop the most accurate surrogate model. The final architecture is a

neural network, composed of 5 linear layers with ReLU activations in between. The loss function consists of the MSE loss. The inputs were standardized similarly to the previous approach.

### 3.4 Reinforcement learning training environment

The surrogate model was used as the basis for the RL training environment. The training environment was developed using the Gymnasium library developed by OpenAI. The surrogate model, models the dynamics of the environment, thus how the state of the environment changes from one timestep to another given a specific action.

Both the action and observation spaces are continuous. The actions the environment receives and the states it outputs are normalized. The environment is responsible for denormalizing both actions and states to store debugging information and performance metrics. The action consists of a 6-dimensional tensor where each action component corresponds to a control action for the surrogate model. The state consists of a 66-dimensional tensor where components correspond to energy consumption data, compartment sensor data for each compartment, state of the environment such as time and weather conditions and battery state of charge. Table 5 and 6 give a full list of action and state features.

<b>RL action components</b>
AHU temperature setpoint
AHU humidity setpoint
Compartment heating setpoint
Compartment cooling setpoint
Compartment inflow
Battery charge/ discharge command

**Table 5:** The components of the action the RL agent outputs

To prevent any extreme behavior from the environment, wrappers were added to the environment. Each wrapper adds additional functionality. The first wrapper normalizes the reward in a rolling window fashion after each. Other wrappers clip the action and observation spaces.

### 3.5 Reinforcement learning agent

An RL agent was developed to interact with the environment with the goal of optimizing power consumption, while following safety constraints. This thesis experiments with different agents, all based on the PPO algorithm. Namely, the three agents that are compared are a traditional PPO agent with a discrete action space, a Penalized PPO (P3O) agent with a discrete action space, and a P3O agent with a continuous action space.

All agents are based on the structure of the PPO algorithm defined in [19]. Both the actor and the critic are neural networks. It is experimented with both a continuous

<b>RL state components</b>
Cooling power
Heating power
Conditioner fans power
FCU Fans power
Pumps power
Hvax auxillary power
Air temperature for each compartment
Air humidity for each compartment
CO <sub>2</sub> levels for each compartment
Air age for each compartment
Time of day
Activity level of the occupants
Outside temperature
Outside humidity
Solar radiation for each compartment
Battery state of charge

**Table 6:** The components of the state the RL environment outputs

action space and a discrete action space for the agent. The P3O agents decouple the reward and the cost from the reward function and receive both of them separately from the environment. Alongside the surrogate and value loss terms, the P3O agents have also a cost surrogate loss.

### 3.5.1 P3O agent with a continuous action space

The main reason behind experimenting with continuous action space is to give the agent more flexibility in the action choices. As the agent can specify the actions to a greater precision, larger optimization gains are achievable in theory.

The policy of the agent is parametrized as a Gaussian distribution. The actor network outputs values for the mean of the distribution given the current state of the agent. To prioritize exploration early on and exploitation as the agent learns, the standard deviation of each action component is large early on and gradually decreases. The agent samples actions from the policy distribution. To make sure that the actions chosen by the agent fall between a reasonable range both clipping a tanh function squashing are applied.

### 3.5.2 P3O agent with a discrete action space

Although a continuous action space gives more flexibility to the agent, it can make learning harder due to a larger selection of possible actions. Another approach is to have discrete action bins for each action component.

The agent can choose between ten choices for each action component. The action

choice is sampled from a categorical distribution, which is parametrized by the actor network. The network outputs logits or unnormalized probabilities for each action choice.

### **3.5.3 PPO agent**

The third agent that was experimented with is a traditional PPO agent. The PPO agent has a discrete action space similar to the previous agent with a categorical distribution policy.

## **3.6 Reinforcement learning training**

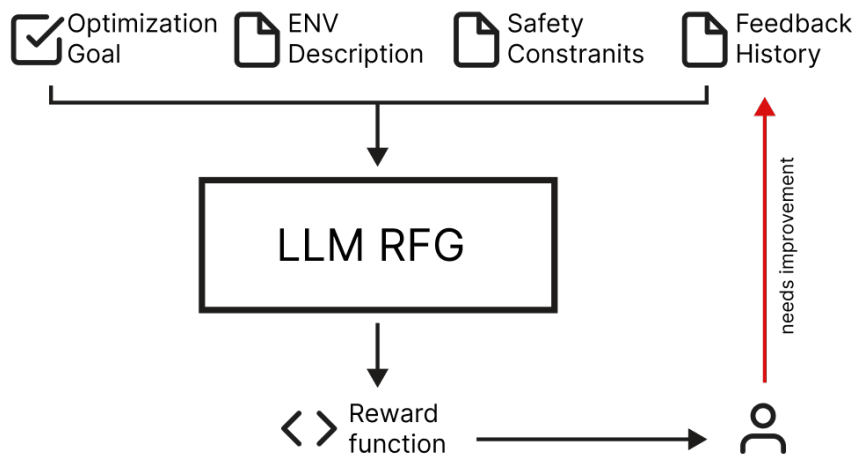
In order to compare the performance of the three agents developed, they were trained for 1000 episodes. Each episode represents a 24-hour interval of time in the simulated cruise ship HVAC system. Each agent was trained three times with 3 different seeds in order to dampen any stochasticity that arises during training.

## **3.7 LLM reward function generation**

In order for the RL agents to learn and to be trained they need a reward function so that they understand the goal they are trying to achieve. In this thesis two methodologies for reward shaping are compared, handcrafted and LLM generated reward functions. The handcrafted reward function was designed given knowledge of the task at hand. The LLM generated reward function was developed through an iterative methodology. During each iteration a prompt was given to the LLM containing: a description of the environment, an optimization goal, safety constraints and a history of the feedback received.

The environment description gives a detailed outline of the challenge. It includes a natural language description of the ship energy system environment, the Pythonic structure of the environment, the role of the LLM as an RL engineer tasked with reward shaping, the structure of the action and observation spaces, the format of the reward function as a python function, the input parameters, and outputs. The optimization goal defines the goal the agent will optimize, namely reducing energy consumption. The safety constraints define safety and controller constraints of the environment, and were produced in an STPA analysis of a general cruise ship HVAC system.

The outputted reward function was used to train and test the RL agent. The performance metrics of the interactions between the agents and the environment were prompted as feedback to the LLM in order to generate a new reward function. As such, they were part of the feedback history. Alongside performance data, the feedback history also includes natural language cues regarding the performance of the agent. The iterative process continues until the LLM is unable to generate a better reward function.



**Figure 15:** The methodology behind LLM reward function generation

### 3.8 Safety shield

As part of the safe reinforcement learning methodology of this thesis a safety shield was also employed. The role of the safety shield is two fold. The safety shield is responsible for keeping track of safety violations both during training and testing. A violation occurs if during a timestep a safety or controller constraint is broken. The safety shield uses conditional logic to detect if the conditions of the safety constraints uphold or not.

The second role of the safety shield is to block unsafe actions from occurring. All PPO-based agents have a stochastic policy and as such, the shield blocks and resamples actions if they lead to safety violations. The way the shield checks if an action is safe or unsafe is by calling the surrogate model directly and then checking if the output state violates the safety conditions. This can be repeated for  $k$  steps in the future, to make sure that the current action does not lead to unsafe states in the future.

However, this implementation of the shield does lead to a conflict. If the agent with the shield is tested on the surrogate environment, the shield would have full information of the state that the agent would land on. For this reason, this thesis experiments with testing the shield on the surrogate model and also directly on the white box model to check the validity of the shield.

This thesis also experiments with utilizing the safety shield both during training and inference, and only during inference. Literature points out that the shield hinders performance when it is used during training [8]. The experiment validates whether this is true or not for this use case.

### 3.9 White-box simulation model

In order to validate the performance of the RL agents more robustly, the final RL agent was also tested directly in the IDA ICE simulation software. The training is done on the surrogate model, which is computationally less expensive to run on, and the inference is done on the white-box simulation environment.

Experiments were conducted to compare the performance of the trained agent on the surrogate model and the white box model for the same timeframe. The role of the experiment is to validate that the performance in the white-box model does not differ significantly.

The effectiveness of the safety shield was also tested in the white-box model. An experiment was conducted to compare the performance of the RL agent in the white-box model with the safety shield on and off.

### **3.10 Open loop control**

To assess the effectiveness of the reinforcement learning methodology, it needs to be compared to other control methods. In this thesis, alongside the reinforcement learning approach, an open-loop control strategy was employed.

During open-loop control, a predefined set of parameters is fed to the white-box simulation without utilizing feedback from the system. The values for the parameters are chosen so that they are safe and that they consume low amounts of energy. To ensure that the setpoints are safe, they were chosen so that they lay comfortably within the range defined by the safety regulations. From experimentation, it was found that controlling the inflow parameter had a large effect on energy consumption. Also, above a certain limit, the inflow parameter did not have any noticeable effect on other modeled parameters such as  $CO_2$ , temperature, or humidity. As such humidity was set to the humidity setpoint was set to the low end of the safe range

## 4 Results and Discussion

This chapter presents and interprets the results of the experiments.

### 4.1 Surrogate model

Two approaches were tested for developing the surrogate model, which led to two different models. The first one is an auto-regressive model, and the second is a traditional surrogate model. To compare the performance of the two models, the root mean squared error (RMSE) was used as a performance metric. The reason for using RMSE is that it is a robust metric for comparing the performance of two different models on the task of multidimensional curve fitting. Also, both models were trained using a mean squared error loss. Thus, they were both trained on the same objective, to lower the mean squared error, which is similar to the RMSE. To calculate the RMSE, the outputs of both models were treated as a timeseries and compared to the outputs of the test simulations outputs given the same set of inputs. The RMSE was computed in the normalized space to prevent domination of any variable. The results of experimenting with surrogate models can be seen in Table 7. The mean RMSE was computed over each dependent variable and test simulation.

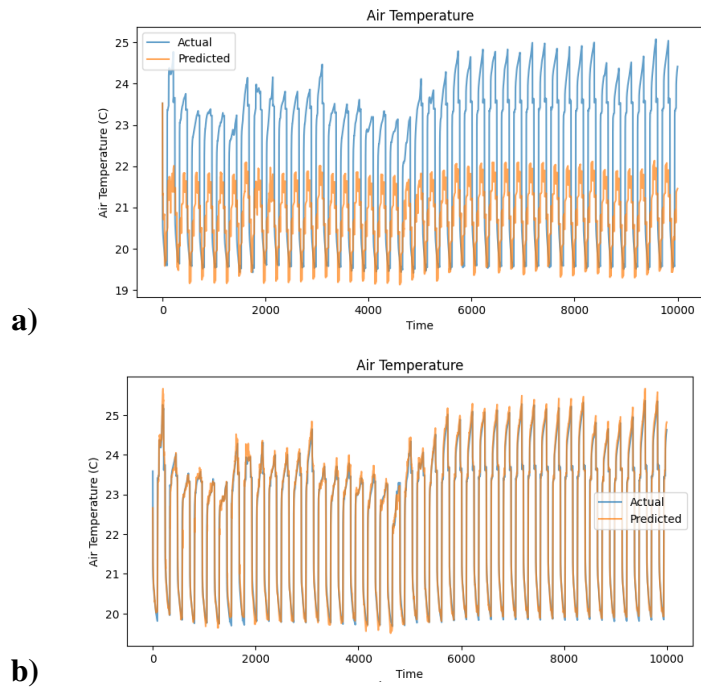
**Table 7:** RMSE score comparison between two approaches. Partly from [29].

Auto regressive model	Traditional surrogate model
0.73	0.061

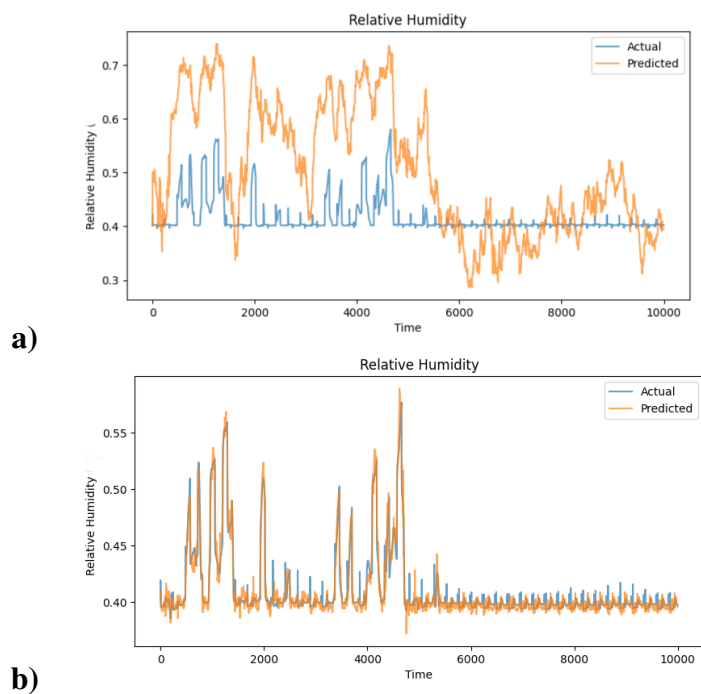
From the results in Table 7 it can be seen that the second approach of using a traditional model was much more accurate. The RMSE score of 0.06 can be interpreted as the model being on average 0.06 standard deviations away from the correct output for any single output.

I believe that the reason the first approach led to poorer results is that it was trying to solve a harder problem than the second model. The first model was trying to learn the dynamics of how the state of the system changes with time. The relationship between the current state of a system and the previous one in time is governed by complex physical equations that are difficult for the neural network to learn. Another issue is the fact that the model is always building upon its past predictions. As such, even small errors get compounded over time, which leads the outputs to stray away from the correct range.

The second approach solved a much simpler problem. It learned the relationship between input features such as outside weather conditions, time of day, and control inputs with the state of the system. As the problem the model solved is easier, it is much more accurate. Furthermore, it is considerably smaller in size, and it is much faster to train and use for inference. Thus, the second model is the chosen surrogate model, which was used by the other parts of the system developed in this thesis.



**Figure 16:** Visual comparison of the two surrogate modeling approaches in modeling the air temperature of a compartment vs the actual data from the simulation software. a) forecasting model, b) surrogate model

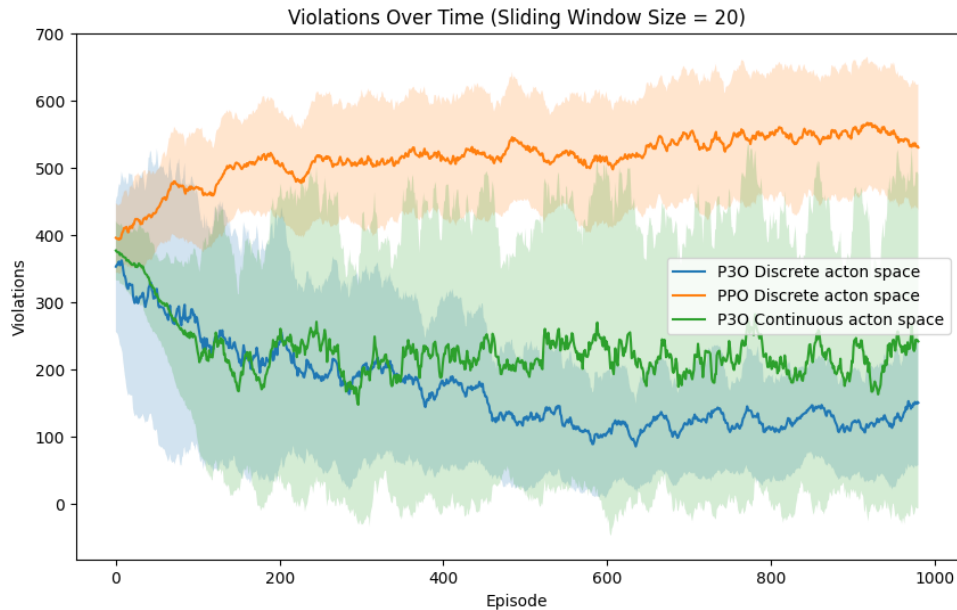


**Figure 17:** Visual comparison of the two surrogate modeling approaches in modeling the air humidity of a compartment vs the actual data from the simulation software. a) forecasting model, b) surrogate model

Figures 16 and 17 give a visual representation of the results of both models. A representative test simulation was used for the graphs. Plotted are the air temperature and humidity levels for one of the cabins for a 10000-timestep period corresponding to almost 42 days. The blue line represents the real temperature and humidity of the cabin as a timeseries, whereas the orange line represents the predicted values. From both images, it can be seen that the second model performs better than the first model, as the outputs of the model are closer to the real values.

## 4.2 Comparing different RL Agents

Multiple RL agents were trained on the surrogate training environment, and their performance on managing a ship energy system was compared. The agents compared are a PPO agent, a P3O agent with a continuous action space, and a P3O agent with a discrete action space. The agents were compared on their ability to reduce energy consumption while being safe. Figure 18 and 19 show the training curves for each agent over 1000 training episodes and 3 runs each. To reduce variance in the plot, the running average was plotted over 20 episodes. Each episode represents 24 hours in the simulated environment. Figure 18 plots the violations per episode for all agents.

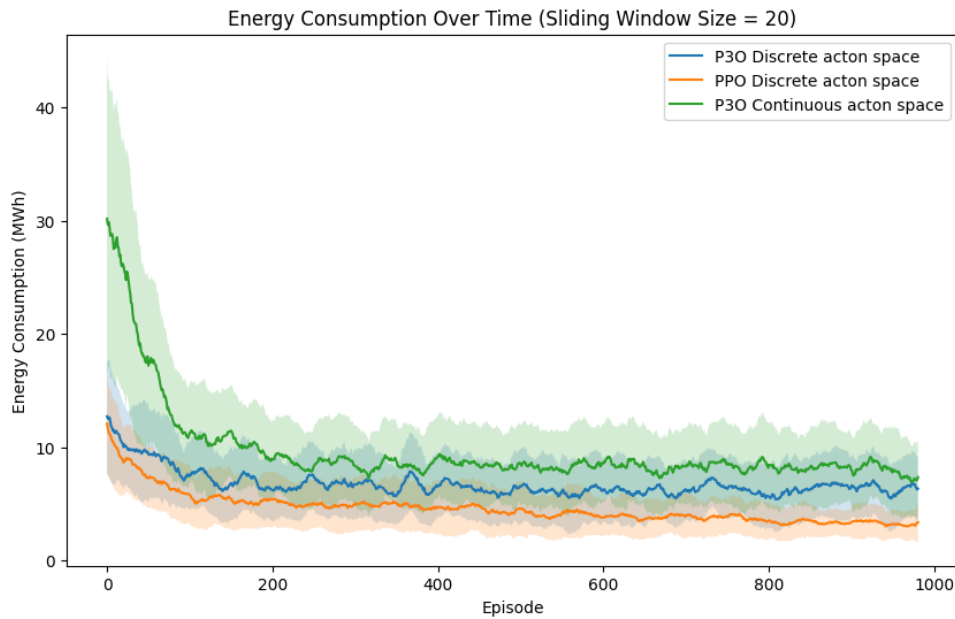


**Figure 18:** Safety violations training curve for the three agents compared. Plotted are the mean and standard deviation over 3 seeds. The running average is plotted over 20 episodes.

While the number of violations might seem high, as it is in the hundreds, the precise number of violations is not of interest. The number of violations depends on how the safety and controller constraints were set up. As such, the comparison between different agents is more important than the number of violations itself. From Figure 18, it can be seen that the PPO agent is the only one that does not learn to

optimize safety as safety violations increase as the agent is training. Both P3O agents perform better as they learn to lower the number of safety violations. The discrete P3O agent performs the best, as it has the lowest mean at the end of training, and it has lower variance than the other P3O agent.

Figure 19 plots the performance of all three agents in energy consumption during training. The energy consumption figure measures the total amount of energy consumed by the hotel system for an episode or 24 hours. All three agents learn to lower energy consumption as training goes on. The PPO agent performs the best of the three as it has a lower mean at the end of training. The two P3O agents perform similarly.



**Figure 19:** Energy consumption training curve for the three agents compared. Plotted are the mean and standard deviation over 3 seeds. The running average is plotted over 20 episodes.

After training, all three agents were tested on 100 test episodes. The results of testing are available in Table 8. The mean and standard deviation over the test episodes are reported. The RL agents were also compared with an open-loop control strategy. The results show that the reinforcement learning control strategy was effective, as two out of the three RL agents beat the open-loop control strategy in energy consumption. The PPO agent achieves the best performance in energy consumption, almost halving the open-loop control strategy. However, it does not improve the safety of the system.

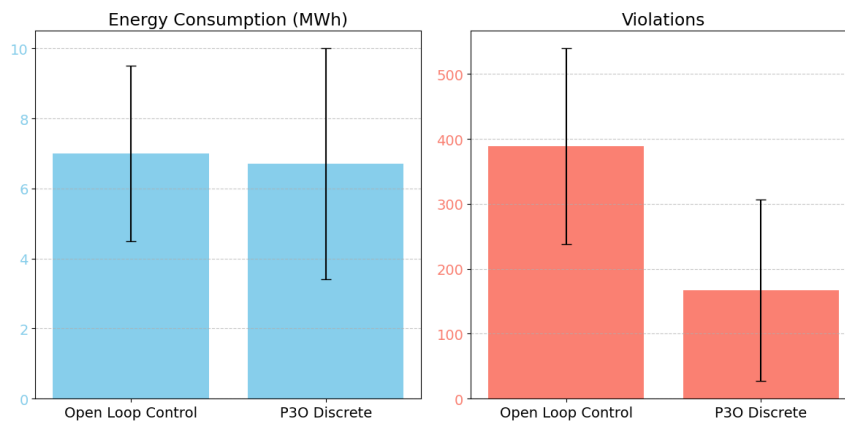
To understand the performance of the PPO agent, compared to the other agents, we must look at its architecture. The PPO agent combines and codifies two conflicting objectives, energy and safety, into a single scalar reward value. As such, how each objective is weighted determines the focus of the agent. An equal weighting might be hard to construct, as the two parts that make the reward scalar are always changing. The reward function used for this PPO agent was handcrafted, and the weighting chosen favors energy consumption optimization. Thus, the strategy the PPO agent

chosed can be interpreted as the agent focusing on optimizing energy consumption while ignoring safety. As such, it was able to achieve large gains in energy savings over the simpler open-loop strategy.

**Table 8:** Comparing different control strategies in power consumption and safety violations. Partly from [29].

Algorithm	Power consumption	Safety violations
Open loop control	$7.0 \pm 2.5$ MWh	$388.5 \pm 151$
PPO	$3.6 \pm 1.4$ MWh	$405.6 \pm 121.5$
P3O Continuous	$7.8 \pm 3.4$ MWh	$201.0 \pm 194.9$
P3O Discrete	$6.7 \pm 3.3$ MWh	<b><math>167.0 \pm 139.8</math></b>

Both P3O agents improve on the safety of the system over the PPO agent and the open-loop control. The P3O agent with a discrete action space performs the best, being the safest agent. The discrete agent performing better than the continuous agent supports the claim that discrete agents may learn faster due to fewer possible action choices. The discrete P3O agent is the best-performing agent overall, reducing both power consumption and safety violations. The agent reduces the number of safety violations by 57% over the open-loop control strategy. It also improves on the energy consumption, unlike the continuous agent, with a 4% reduction in energy usage. This is the agent that was used for the experiments presented further on. Figure 20 plots the performance of the Discrete P3O agent with the alternative control strategy that is open-loop control.



**Figure 20:** Bar plot comparing the performance of the open-loop control strategy with the discrete P3O agent. The error bars quantify the standard deviation.

### 4.3 LLM reward function generation

This thesis also explored the use of Large Language Models for RL reward function generation. In order to comply with the P3O algorithm, the LLM was instructed to

design a reward function that returns both a reward and a cost. The LLM-based reward function (RF) was compared with a handcrafted RF. The handcrafted reward function was developed based on knowledge of the system. The reward chosen is the negative of the power consumption, scaled down to prevent numerical overflow issues. Thus, the more energy the agent consumes in a single timestep, the more negative the reward. The environment normalizes the reward before it is collected by the agent. The cost represents the constraints of the problem. The chosen cost is computed by adding the excess residuals of the sensor values from their safe ranges. Thus, if the state of the system for a particular sensor, such as temperature, is far away from the safe range, the agent would incur a large cost.

The LLM used is GPT4. From the experiments, it was observed that LLMs are generally quite capable of shaping a reward function. The outputted reward function was functionally correct from the first iteration. Additional iterations of code generation were conducted by adding feedback. The feedback added was minor. After the first iteration, the LLM was given agent performance data and told to come up with a better reward function. After the second iteration, the LLM was informed that safety violations are tracked and that the outputted RF should try to lower them. The final reward function was generated after 3 iterations.

#### 4.4 Safety shield

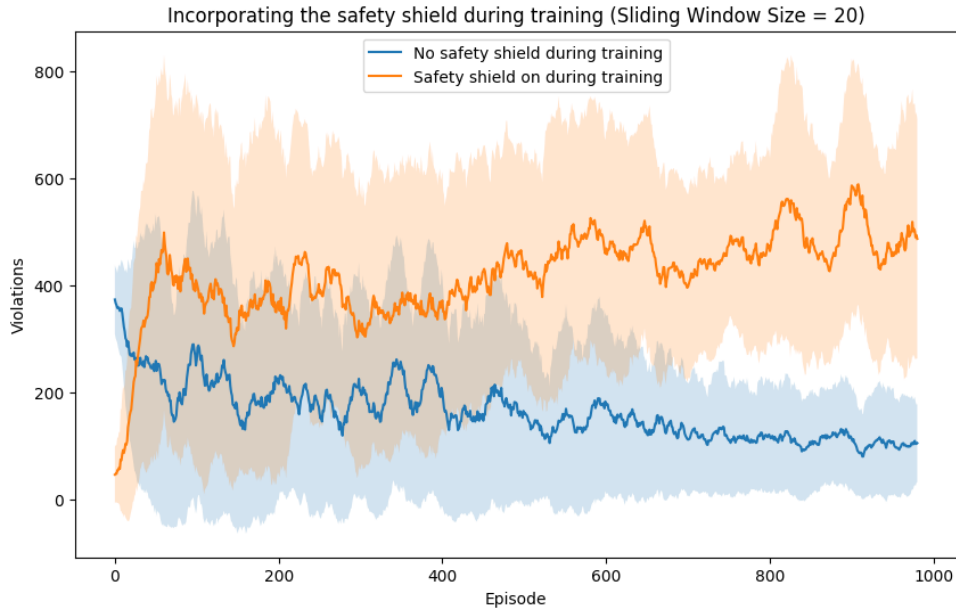
A safety shield was developed to enhance the safety of the system. The safety shield gets information about the current state of the agent and the action the agent is taking [34]. An experiment was conducted to explore the use of the safety shield during training. Two identical agents were trained for 3 different seeds, one with the shield on and one with the shield off. The training curves of both agents are plotted in Figure 21.

From the training curve, it can be observed that the safety shield has a large negative effect on the performance of the agent during training. The reason for the drop in performance may be that the shield limits the exploration of the agent. This is especially true for unsafe states that would give the agent large negative feedback. The outcome supports the literature [8].

#### 4.5 Incorporating guardrails

Both the safety shield and the LLM-generated reward function serve as guardrails, preventing the system from derailing into unsafe states. Table 9 presents the results of incorporating both of them. The results show the mean and standard deviations over 100 test episodes and 3 seeds. The rows that do not mention the LLM RF have a handcrafted reward function.

The results show that both guardrails are effective at reducing violations. Utilizing the shield reduces safety violations by 61% over the unshielded agent. Incorporating the LLM RF reduces violations by 56% over the handcrafted RF. Combining both leads to the safest result with a 70% reduction over pure P3O. While incorporating guardrails enhances safety, it comes at a cost. The system is more constrained in the actions it can choose, and as such, it will choose actions that are less optimal regarding



**Figure 21:** Training curves of the RL agent with the shield active and inactive during training. The mean and standard deviation over 3 seeds are plotted [34].

energy consumption. This is especially true in the case of utilizing the LLM RF. While it improves on safety, the power consumption grows to a large degree. The shielded P3O agent is an outlier as it is able to increase energy efficiency over regular P3O. However, the two experiments represent different agents that were trained from scratch and not the same agent with and without the shield. As such, stochasticity could play a role in the results. The best-performing agent is the one that is able to optimize both conflicting objectives, energy performance, and safety. While the safest result is achieved by the last experiment with shielded P3O and LLM RF, it led to a large performance drop in energy efficiency. I believe this is because it was too constrained. The best result overall is achieved by the shielded P3O agent with a handcrafted reward function, which balances safety and performance.

**Table 9:** Results of integrating the safety shield and the LLM-based reward function

Algorithm	Power consumption	Safety violations
P3O	$6.7 \pm 3.3$ MWh	$167.0 \pm 139.8$
P3O + LLM RF	$10.9 \pm 4.6$ MWh	$72.1 \pm 83.7$
<b>P3O + shield</b>	<b><math>6.4 \pm 3.0</math> MWh</b>	$64.4 \pm 55.5$
P3O + shield + LLM RF	$11.5 \pm 4.8$ MWh	<b><math>48.9 \pm 47.7</math></b>

## 4.6 Results of testing in the white-box environment

The RL agents were trained and tested in the surrogate model. The surrogate model is a neural network approximation of the physical model, and as such, it is less accurate. To evaluate the performance of the agent, it is also tested on the white-box model. Running the white-box model is much more computationally expensive, thus, experiments on it focused on single episodes. A single episode took 4 hours of real time to run on a regular laptop.

Table 10 evaluates the effectiveness of the safety shield during inference of the agent on the whitebox model. Each row represents the results of a single episode. The agent is P3O with LLM-generated RF. Results suggest that the shield has no negative effect on energy consumption, while it managed to reduce violations by 14.2%.

**Table 10:** Results of testing the agents in the white-box environment. Each experiment represents a single episode.

Algorithm	Power consumption	Safety violations
P3O	5.6 MWh	77
P3O + shield	5.5 MWh	66

## 5 Conclusion

This chapter describes the limitations of the thesis, the main conclusions, and key takeaways.

### 5.1 Limitations and Future work

There are several limitations to the experiments conducted that must be addressed in order to have a detailed view of the results shown.

The biggest limitation is in the environment used. Firstly, the software used, IDA ICE, is limited in its ability to portray a ship simulation environment due to it being primarily a building simulator. Secondly, the simulation used approximates a real ship system, and as such, loses some amount of fidelity to the real system. Furthermore, there was no experimentation involved in comparing the white box simulation with a real-life system to validate the performance of the simulation. The black box surrogate model further approximates the environment. Thus, while the methodology used is valid in the environments it was tested on in this thesis, future work on more comprehensive systems may be needed in order to fully validate the methodology.

Other limitations are related to the training of the RL system. The RL agent was trained on a neural network surrogate model. While neural networks are highly accurate, as they are able to mimic a complex system like an EMS, their outputs are affected by noise or approximation error. The outputs of a white box simulation are produced by physical equations and, as such, produce a smooth curve. The outputs of the surrogate model instead are affected by approximation error due to their nature and produce a less smooth or bumpy curve. The outputs of the surrogate model make it harder for the agent to learn due to the increased incidence of local optima. This makes the training of the RL agent and the results produced much more susceptible to stochasticity. Another limitation of the surrogate model, as discussed in the results section, is the limitation in modeling temporal changes. The state of the system at a timestep has a lesser effect on future conditions when compared to a white box model. As such, the agent may wrongly learn that their actions have little effect on the future and may focus on short-term gains. Future work on dealing with this limitation could include further research into surrogate modeling or working with a different type of model entirely, such as a white-box model or a gray-box model that combines features of both models.

Another limitation of the results is computational or software-related limitations. The IDA ICE software requires specific libraries to run and can only be run on a Windows system with a graphical interface. Also, the software is quite computationally expensive, as mentioned before. Since it cannot be run on a terminal-only or Linux system, it is not possible to take advantage of a computer cluster to speed up computation. Due to this limitation, only a small number of simulations were produced as training data for training the surrogate model. A larger pool of simulations could have led to more accurate results. Furthermore, testing on the white-box model was slow, and only a single episode for each experiment was produced. Due to the high variance of RL methods, one episode is a low sample for producing a good comparison.

A potential limitation of the thesis is the alternative control strategy that was compared with the RL controller. The open-loop control strategy, while effective, is relatively simple and not the best-performing choice when considering traditional methods. Literature has shown that other control strategies, such as PID controllers or optimization algorithms, perform better at the task given. However, these strategies are more complicated to implement and would have taken away from the scope of the project. Future work could implement other traditional controllers for a fairer comparison to the RL agent.

While not a limitation, the scope of the thesis was rather broad as it tackled different domains such as reinforcement learning, surrogate modeling, and large language models. Each domain was necessary to develop the parts of the RL controller system and their integration. Future more exhaustive work on specific parts of the system could also be insightful. LLM reward function generation could be explored more in depth by providing comparisons over a large array of different LLMs. Their performance could be compared based on model parameter size or release date. Also, the effectiveness of the iterative methodology for RFG could be explored in more detail. This can be achieved by keeping track of the performance of the system over multiple iterations. Another part of the system that can be explored in more detail is the safety shield. Experimentation could compare the performance of the shield over different branching depths and resampling thresholds for action selection. Also, the effectiveness of the shield on the white box model could be assessed in more detail.

## 5.2 Conclusions

This thesis explored the development and implementation of an RL-controlled system for ship energy management optimization. The research in thesis attempts to assess the efficiency and safety of such system especially compared to existing methods. To assess the efficacy of the system as a whole, the performance of the different parts of the system was evaluated. The key areas this thesis looks into are: the performance of the RL controller compared to other control strategies, the added safety of incorporating safety mechanisms and guardrails, and the viability of generative AI methods for reward shaping.

Experiments showed that reinforcement learning was an effective control strategy. Multiple agents were compared to one another and an alternative control strategy, namely open-loop control. Results showed that the RL agents were able to effectively learn how to manage the energy system of the ship. Two out of the three agents tested were able to outperform the alternative control strategy in energy consumption. The RL agents also learned how to act safely. The best performing agent overall was a P3O agent with a discrete action space, which reduced the power consumption by 4% and safety violations by 57%.

It was found that safety-focused methods indeed add to the safety of the system. The first safety-focused method utilized is the P3O reinforcement learning agent. The agent was able to improve safety over the regular PPO agent by reducing the number of safety violations by 58.7 % over the PPO agent. Furthermore, a safety shield and an LLM-generated reward function were incorporated as guardrails to prevent the system

from derailing into unsafe states. Both methods achieved an improvement in safety violations of 61% and 57%, respectively. Incorporating all three safe RL methods led to the safest agent, which achieved 87% fewer violations than the open-loop control strategy. While the safest implementation was not the most energy-efficient, the large gains in safety enhancement validate the effectiveness of safety-based methods for enhancing the safety of RL-based systems.

Experimenting with large language models showed that generative AI is capable of developing safe and robust reward functions for reinforcement learning problems. The LLM-generated reward function was able to outperform the handcrafted RF in reducing safety violations, but it failed in reducing consumption, leading to a higher energy usage. This suggests that LLM-generated RF are not yet able to outperform handcrafted RF. However, the fact that they achieved similar performance while reducing the amount of engineering time and engineering know-how to develop them, suggests that they are a viable alternative to handcrafted rewards.

While the results of experiments run on the white-box model directly are not exhaustive due to low sample size, they do add to the validity of the thesis. They show that a model trained on a simpler approximated surrogate model can perform well on the physical simulation as well. Thus, it is reasonable to believe that the methodology could also be transferable and effective on a real ship system.

## References

- [1] European Commission, *Reducing emissions in the shipping sector*, [https://climate.ec.europa.eu/eu-action/transport/reducing-emissions-shipping-sector\\_en](https://climate.ec.europa.eu/eu-action/transport/reducing-emissions-shipping-sector_en), EU Climate Action, n.d.
- [2] European Commission, *International cooperation and coordination in maritime transport*, [https://transport.ec.europa.eu/transport-modes/maritime/international-cooperation-and-coordination\\_en](https://transport.ec.europa.eu/transport-modes/maritime/international-cooperation-and-coordination_en), EU Transport, n.d.
- [3] ABB. “HVAC - system retrofits to lower energy consumption - Energy Efficiency”. Online; accessed 2025-04-08. (2025), [Online]. Available: <https://new.abb.com/marine/energy-efficiency/hvac>.
- [4] M. Aarnio, *Cruise Ship Handbook*. Cham, Switzerland: Springer, 2022.
- [5] M. H. Alshareef and A. F. Alghanmi, “Optimizing maritime energy efficiency: A machine learning approach using deep reinforcement learning for eexi and cii compliance”, *Sustainability*, vol. 16, p. 10 534, 2024. DOI: [10.3390/su162310534](https://doi.org/10.3390/su162310534).
- [6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd. Cambridge, MA, USA: MIT Press, 2018.
- [7] N. Jansen *et al.*, “Safe reinforcement learning using probabilistic shields”, in *Proceedings of the 31st International Conference on Concurrency Theory (CONCUR 2020)*, 2020.
- [8] B. Könighofer *et al.*, “Online shielding for reinforcement learning”, *Innovative Systems and Software Engineering*, vol. 19, 2023.
- [9] T. Xie *et al.*, “Text2reward: Automated dense reward function generation for reinforcement learning”, in *International Conference on Learning Representations (ICLR), 2024 (07/05/2024-11/05/2024, Vienna, Austria)*, 2024.
- [10] W. Yu *et al.*, *Language to rewards for robotic skill synthesis*, arXiv:2306.08647, 2023.
- [11] U. Atmojo *et al.*, *Safe intelligent agent to optimize ship energy management*, <https://research.aalto.fi/fi/projects/safe-intelligent-agent-to-optimize-ship-energy-management>, Aalto University, Department of Electrical Engineering and Automation. Accessed: May. 12, 2025, 2025.
- [12] International Organization for Standardization, *Iso 7547:2022 - ships and marine technology – air-conditioning and ventilation of accommodation spaces – design conditions and basis of calculations*, Standard, Geneva, Switzerland: International Organization for Standardization, 2022.
- [13] G. Karmiris and T. Tengnér, “Peak shaving control method for energy storage”, ABB AB, Corporate Research Center, Västerås, Sweden, Tech. Rep., 2025.

- [14] L. Cox. “‘world’s largest’ electric ship measuring 130 metres launched by tasmanian boatbuilder”. Accessed: 2025-06-18, The Guardian. (May 2, 2025), [Online]. Available: <https://www.theguardian.com/australia-news/2025/may/02/worlds-largest-electric-ship-measuring-130-metres-launched-by-tasmanian-boatbuilder>.
- [15] BasicKnowledge101, *Control theory*, Accessed: 2025-06-18. [Online]. Available: <https://www.basicknowledge101.com/pdf/control/Control%20theory.pdf>.
- [16] N.-H. K., *Surrogate Modeling and Optimization: Theories, Applications, and Limitations*, 1st ed. Hoboken, NJ: Wiley, 2025, ISBN: 9781394245819.
- [17] W. Bianca and C. Selen, “Novel tool for selecting surrogate modeling techniques for surface approximation”, in *Computer Aided Chemical Engineering*, Elsevier, 2025. DOI: [10.1016/B978-0-323-88506-5.50071-1](https://doi.org/10.1016/B978-0-323-88506-5.50071-1).
- [18] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Playing Atari with deep reinforcement learning”, in *NIPS Deep Learning Workshop*, arXiv:1312.5602 [cs.LG], 2013. DOI: [10.48550/arXiv.1312.5602](https://doi.org/10.48550/arXiv.1312.5602). [Online]. Available: <https://arxiv.org/abs/1312.5602>.
- [19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms”, 2017. arXiv: [1707.06347](https://arxiv.org/abs/1707.06347) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1707.06347>.
- [20] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation”, in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
- [21] L. Zhang, L. Shen, L. Yang, *et al.*, “Penalized proximal policy optimization for safe reinforcement learning”, in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, L. D. Raedt, Ed., Main Track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 3744–3750. DOI: [10.24963/ijcai.2022/520](https://doi.org/10.24963/ijcai.2022/520). [Online]. Available: <https://doi.org/10.24963/ijcai.2022/520>.
- [22] M. Andrychowicz, A. Raichuk, P. Stańczyk, *et al.*, “What matters in on-policy reinforcement learning? a large-scale empirical study”, *arXiv preprint arXiv:2006.05990*, Jun. 2020, Google Research, Brain Team.
- [23] D. S. Watkins, *Transformations of random variables*, Technical report, University of Arizona, Available at <https://math.arizona.edu/~jwatkins/f-transform.pdf>, Sep. 2009.
- [24] W. X. Zhao *et al.*, *A survey of large language models*, <https://arxiv.org/pdf/2303.18223>, 2023.

- [25] N. G. Leveson and J. P. Thomas, *Stpa handbook*, Accessed: 2025-06-18, MIT Partnership for Systems Approaches to Safety and Security (PSASS), Cambridge, Massachusetts, USA, 2018. [Online]. Available: [https://psas.scripts.mit.edu/home/get\\_file.php?name=STPA\\_handbook.pdf](https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf).
- [26] Z. Yang, W. Qu, and J. Zhuo, “Optimization of energy consumption in ship propulsion control under severe sea conditions”, *Journal of Marine Science and Engineering*, vol. 12, no. 9, p. 1461, 2024. DOI: [10.3390/jmse12091461](https://doi.org/10.3390/jmse12091461). [Online]. Available: <https://doi.org/10.3390/jmse12091461>.
- [27] X. Wang, H. Zhu, X. Luo, S. Chang, and X. Guan, “A novel optimal dispatch strategy for hybrid energy ship power system based on the improved nsga-ii algorithm”, *Electric Power Systems Research*, vol. 232, p. 110385, 2024. DOI: [10.1016/j.epsr.2024.110385](https://doi.org/10.1016/j.epsr.2024.110385). [Online]. Available: <https://doi.org/10.1016/j.epsr.2024.110385>.
- [28] Y. Zhao, S. Wen, Q. Zhao, B. Zhang, and Y. Huang, “Deep reinforcement learning-based energy management strategy for green ships considering photovoltaic uncertainty”, *Journal of Marine Science and Engineering*, vol. 13, no. 3, p. 565, 2025. DOI: [10.3390/jmse13030565](https://doi.org/10.3390/jmse13030565). [Online]. Available: <https://doi.org/10.3390/jmse13030565>.
- [29] E. Shahinas, A. King, and U. D. Atmojo, “Safe reinforcement learning for ship energy management optimization with llm-based reward shaping”, To appear at IEEE ETFA 2025.
- [30] E. S. AB, *Ida ice 5: Indoor climate and energy simulation tool*, Accessed: 2025-04-10, 2025. [Online]. Available: <https://www.equa.se/ida-ice>.
- [31] S. Sierla *et al.*, *Adaptive multi-energy virtual power plant for a complex of buildings*, <https://research.aalto.fi/en/projects/multi-energy-vpp>, Aalto University, Department of Electrical Engineering and Automation. Accessed: Apr. 13, 2025, 2025.
- [32] H. Aaltonen *et al.*, “Bidding a battery on electricity markets and minimizing battery aging costs: A reinforcement learning approach”, *Energies*, vol. 15, no. 14, p. 4960, 2022. DOI: [10.3390/en15144960](https://doi.org/10.3390/en15144960).
- [33] C. Nuchturee, T. Li, and H. Xia, “Design of cost-effective and emission-aware power plant system for integrated electric propulsion ships”, *Journal of Marine Science and Engineering*, vol. 9, no. 7, p. 684, Jun. 2021. DOI: [10.3390/jmse9070684](https://doi.org/10.3390/jmse9070684).
- [34] A. King, E. Shahinas, and U. D. Atmojo, “Exploring safe reinforcement learning using safety shields derived with system-theoretic process analysis: A case-study on a cruise ship hotel system”, To appear at IEEE ETFA 2025.