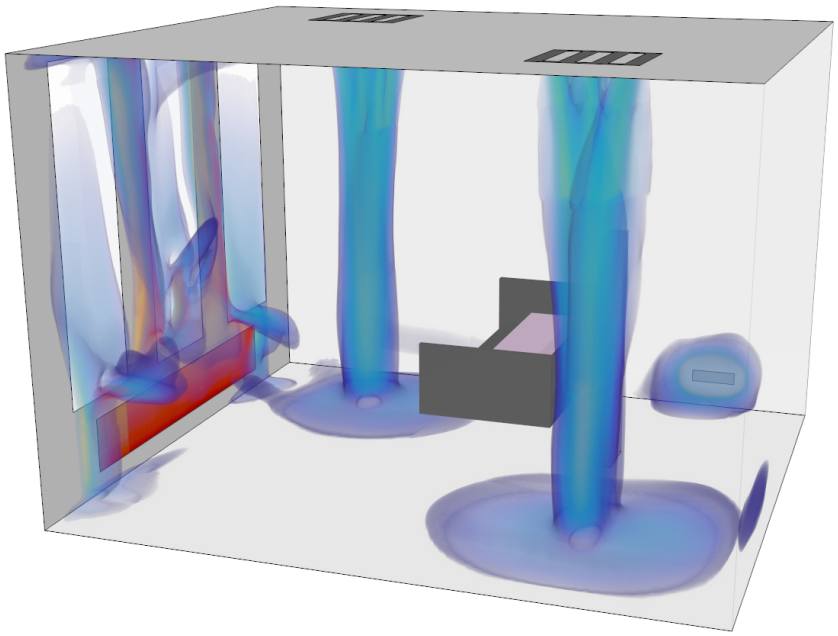


Dynamic Control of HVAC Attributes

Improving Energy Efficiency of HVAC System Using Machine Learning and Computational Fluid Dynamics

Master's thesis in Applied Mechanics

RAJ GOPALAKRISHNA SUBRAMANI VENKATACHALAM

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's Thesis in Applied Mechanics

Dynamic Control of HVAC Attributes

Improving Energy Efficiency of HVAC System
Using Machine Learning and Computational Fluid Dynamics

RAJ GOPALAKRISHNA SUBRAMANI VENKATACHALAM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2025

Dynamic Control of HVAC Attributes
Improving Energy Efficiency of HVAC System Using
Machine Learning and Computational Fluid Dynamics
RAJ GOPALAKRISHNA SUBRAMANI VENKATACHALAM

Supervisor: Anthony Jayanath Vivek, M.Sc.,
Development Lead With AI and CFD,
AFRY.

Examiner: Lars Davidson,
Professor
Department of Mechanics and Maritime Sciences,
Chalmers University of Technology.

Master's Thesis 2025
Division of Fluid Dynamics, Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Volume rendering of velocity flow field inside a ventilated room, illustrating airflow distribution around heating and ventilation sources.

Typeset in LaTeX
Gothenburg, Sweden 2025

Dynamic Control of HVAC Attributes
Improving Energy Efficiency Using Machine Learning and Computational Fluid Dynamics
RAJ GOPALAKRISHNA SUBRAMANI VENKATACHALAM
Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
Chalmers University of Technology

Abstract

Heating, Ventilation and Air-Conditioning (HVAC) accounts for a major share of building energy use. This thesis develops a data-driven HVAC control framework that couples high-fidelity Computational Fluid Dynamics (CFD) with machine learning. Three-dimensional CFD simulation is performed to model the flow field in the room using Star-CCM+. 973 steady state CFD simulations representing realistic boundary conditions were performed to form a comprehensive dataset of temperature and velocity fields. Fourier Neural Operator (FNO) was trained as a surrogate model to CFD. This model reproduces the temperature and velocity flow fields with adequate resolution, with over 90% of predicted temperature and velocity fields across unseen samples deviating within a 5% error, making it suitable for closed-loop use. The error distribution analysis shows that the median temperature error is at 0.0169 °C and median velocity error is 0.0034 m/s, indicating that the surrogate model is reliable to predict temperature and velocity fields.
The surrogate model coupled with a Soft Actor-Critic (SAC) controller, which was designed to regulate inlet air temperature, inlet mass-flow rate, and radiator surface temperature to maximize the reward, that provides an optimal control solution to reduce the energy cost.
The controller was evaluated for a year of weather data with a 20-minute control step and benchmarked against a PID controller. Results show that the SAC consumed 4,836 kWh compared to 7,460 kWh for PID, which corresponds to approximately 35% in energy cost reduction. SAC occasionally produces larger deviations than PID, leading to a higher median temperature error (0.487 °C vs 0.273 °C), but fluctuations beyond ±2.5 °C occurred only 2.3% of the time, indicating that comfort violations remained rare. Seasonal analysis shows SAC controller's energy savings persist across the year and strengthen in the late-year window (36% vs 34% earlier), reflecting adaptive use of outdoor conditions and smoother control.
Overall, the work demonstrates that combining CFD data trained surrogate model with entropy-regularized reinforcement learning can deliver substantial energy savings with acceptable comfort tracking, and provides a practical route to incorporate detailed physics (e.g. radiative gains, occupancy, humidity/CO_2) and more advanced control designs in future studies.

Keywords: HVAC control, Computational Fluid Dynamics (CFD), Fourier Neural Operator (FNO), Surrogate modelling, Reinforcement learning, Soft Actor-Critic (SAC), Thermal comfort, Data-Driven Control.

Acknowledgements

I would like to provide my sincere gratitude to my supervisors, Anthony Jayanath Vivek and Fredrik Jareman from AFRY, for providing me the opportunity to undertake this thesis, for their invaluable guidance and support throughout. Further, I would like to thank Prof. Lars Davidson from Chalmers University, who served as the examiner for this thesis, providing me valuable support through this project.

I am deeply grateful to André Fransson, Blanca González Lozano, Dimitrios Koutsimanis and Viktor Alatalo from AFRY for showing their consistent interest in technical help and emotional support, making the long days lighter and making the thesis stronger and more enjoyable.

Finally, I would like to thank my family and friends for their encouragement and faith in me.

Raj Gopalakrishna Subramani Venkatachalam, Gothenburg, September 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AMR	Adaptive Mesh Refinement
BMS	Building Management System
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy condition
CNN	Convolutional Neural Network
COP	Coefficient of Performance
DDC	Direct Digital Control
DOF	Degrees of Freedom
ELU	Exponential Linear Unit
FFT	Fast Fourier Transform
FNO	Fourier Neural Operator
HVAC	Heating, Ventilation, and Air Conditioning
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
PDE	Partial Differential Equation
PID	Proportional–Integral–Derivative
PI	Proportional–Integral
PLC	Programmable Logic Controller
Re	Reynolds Number
Ri	Richardson Number
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
UA	Overall Heat Transfer Coefficient (U) $\times$ Area (A)

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Parameters

a	Action vector
α	Entropy temperature coefficient (SAC)
c_p	Specific heat capacity [J/kg·K]
COP	Coefficient of performance (coil)
d_{act}	Normalized control action variation
ΔT	Temperature difference [K or °C]
ΔT_{spread}	Temperature stratification across occupied zone [°C]
Δt	Time discretization step [s]
δ	Temporal-difference error
ε	Turbulent dissipation rate [m^2/s^3]
ε	Temperature error (deviation from set-point) [°C]
E_{coil}	Coil energy consumption [kWh]
E_{fan}	Fan energy consumption [kWh]
E_{rad}	Radiator energy consumption [kWh]
E_{tot}	Total energy consumption [kWh]
E_n	Normalized energy penalty
f_i	Body force [N]
G_k	Production term from buoyancy effects
h	Hidden channel width in FNO layers
$\mathcal{H}(\pi)$	Policy entropy
I	Integral term contribution
$J(\pi)$	Policy objective function
K_d	Derivative gain
K_i	Integral gain
K_p	Proportional gain

k	Turbulent kinetic energy [m^2/s^2]
λ_1	Largest eigenvalue of strain-rate tensor
$\mathcal{L}$	Loss function
$\dot{m}$	Mass flow rate [kg/s]
N	Number of training cases / iterations (context-dependent)
n_{modes}	Fourier modes in spectral space
P	Pressure [Pa]
P	Proportional term contribution
P_k	Production term from mean velocity gradients
P_{fan}	Fan power [W]
P_L	Active power of load demand [W]
P_{PV}	Active power from solar generation [W]
Q	Heat source term (J or W depending on context)
$Q_\pi(s, a)$	State–action value function under policy π
Q_{coil}	Coil heat transfer [W]
Q_{rad}	Radiator heat transfer [W]
r	Reward scalar
ρ	Density [kg/m^3]
σ_{ij}	Stress tensor [N/m^2]
σ_{zone}	Standard deviation of occupied zone temperature [°C]
s_{ij}	Mean strain-rate tensor
t	Time [s]
T_{inlet}	Inlet air temperature [°C]
T_{init}	Initial room temperature [°C]
T_{out}	Outdoor (window) temperature [°C]
T_{rad}	Radiator surface temperature [°C]
T_{room}	Indoor mean room temperature [°C]
T_{target}	Target (set-point) temperature [°C]
τ	Polyak averaging coefficient
τ_{ij}	Shear stress tensor [N/m^2]
$u(t)$	Control output at time t
$\vec{u}_i$	Velocity vector and its i-th component [m/s]
U	Internal energy [J]
UA_{rad}	Radiator conductance [W/K]
$V(s)$	State-value function
ν_t	Eddy (turbulent) viscosity [m^2/s]
X	Input tensor (features)

Y	Output tensor (targets)
y	Ground truth field
$\hat{y}$	Predicted output field
$z_{\text{one, dev}}$	Zone deviation penalty
$z_{\text{one, std}}$	Zone standard deviation penalty
θ	Trainable model parameters
$\pi(a\|s)$	Stochastic policy distribution
γ	Discount factor
$\mathcal{D}$	Replay buffer
c_{in} or C_{in}	Number of input channels
c_{out} or C_{out}	Number of output channels
η_{ch}, η_{dis}	Charging / Discharging efficiency

Contents

List of Figures

List of Tables

1
Introduction

Heating, Ventilation, and Air Conditioning (HVAC) systems are an essential part of modern buildings, serving multiple purposes, primarily to regulate the indoor environment to ensure thermal comfort for occupants, HVAC systems account for 20-40% of the total energy consumption [1]. As buildings increasingly account for a significant share of the global energy use, optimizing HVAC system performance has become a critical concern in the pursuit of sustainable and energy-efficient architectural practices.

Traditionally, HVAC systems utilize rule-based control strategies which are based on time-of-day schedules, set-point resets, and static schedules. The most commonly used rule-based controllers are PID controllers, and their theoretical background is discussed in Section 2.3.9. While these conventional methods are straightforward to implement and provide acceptable regulation under periodic variations, they are limited in their ability to respond effectively to unpredictable changes in indoor occupancy, external weather conditions, or user preferences [2]. As a result, these control methods require extensive tuning. Consequently, such systems may either compromise comfort or consume excess energy, or both [2].

To address these challenges, recent advances in computational modelling and artificial intelligence have opened promising avenues for improvement [3]. Among these, the integration of Machine Learning (ML) techniques with high-fidelity Computational Fluid Dynamics (CFD) simulations has emerged as a particularly effective approach. Machine Learning algorithms are well suited for capturing patterns from large datasets and can be trained to predict system behaviour under a wide range of conditions [3]. In contrast, CFD simulations offer physically accurate representations of fluid flow, heat transfer, and other thermodynamic phenomena within the spatial layout of a room. By combining these two powerful tools, it becomes possible to create intelligent models that not only learn from data but also retain the physical realism provided by CFD.

This thesis focuses on developing a physics-informed HVAC control framework which couples a surrogate model based on indoor thermo-fluid dynamics with a dynamic controller which is based on reinforcement learning. A broad set of high fidelity CFD simulations for various boundary conditions is used to train a Fourier Neural Operator (FNO) which predicts three-dimensional temperature and velocity fields for a room, faster than CFD. A Soft Actor-Critic (SAC) controller is coupled with this surrogate model to select supply air temperature, mass-flow rate, and radiator set point to balance comfort and energy usage under dynamically varying weather conditions. The trained policy operates in real-time while the surrogate model provides the current room state.

The final outcome of this thesis is expected to be a computational framework that can serve as a real-time closed-loop decision-making model for the HVAC systems. Unlike traditional approaches that require repeated numerical simulations or rely on fixed control rules, this data-driven methodology can quickly adapt to new scenarios, reduce computational burden, and maintain high fidelity in predictions. Moreover, by continuously optimizing

control actions based on learned experience, the system contributes to reducing energy consumption and improving indoor comfort sustainably.
Through this interdisciplinary approach that combines fluid dynamics, machine learning, and control theory, the thesis contributes to the ongoing development of smart building technologies. It seeks not only to demonstrate the technical feasibility of ML-CFD hybrid models in HVAC applications, but also to offer a practical, scalable solution that aligns with the global push towards greener and smarter infrastructure systems.

2
Theory

This section provides insights for necessary theoretical knowledge to understand the methodology and interpretation of results in this thesis. It gives an introduction to the governing equations, physical principles and relevant turbulence models considered and used for the simulations. Further, it gives an overview of machine learning, including the ML model chosen for this thesis and its associated hyperparameters. This establishes a foundation for the simulations and provides necessary support on interpretation of the analysis. The theoretical framework presented in this section is derived from the comprehensive course compendium by Davidson [4], which serves as the primary reference for the discussion of fluid mechanics, turbulent flow, and turbulence modelling.

2.1 Governing Equations

The continuum approach describes the behaviour of fluids as continuous media responding to mechanical forces. While there are several mathematical models to describe fluid flow, the Navier–Stokes equations offer the most accurate representation within the continuum framework. These equations are based on three fundamental conservation laws, described below.

- **Conservation of Mass:** This principle states that the net rate of mass entering a control volume must equal the net rate of mass leaving it. For incompressible fluid flow, the conservation of mass is expressed as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_i)}{\partial x_i} = 0 \tag{2.1}$$

  where,
  - ρ — fluid density
  - t — time
  - u_i — velocity component in the i^{th} direction

- **Conservation of Momentum:** The momentum of the fluid is defined as ρu, per unit volume. This principle is based on Newton's second law of motion, which explains the conservation of momentum in the continuum. It states that the rate of change of momentum is equal to the sum of forces acting on the fluid. This is represented as:

$$\rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = -\frac{\partial P}{\partial x_i} + \frac{\partial \tau_{ji}}{\partial x_j} + \rho f_i, \qquad \tau_{ji} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \tag{2.2}$$

  where,

- P — pressure
- τ_{ij} — viscous shear stress tensor
- f_i — body force per unit mass in the i^{th} direction
- μ — dynamic viscosity of the fluid,

The first term in the left-hand side represents the rate of change of momentum per unit volume in i^{th} direction. The material derivative is then expanded to local acceleration $\left(\frac{\partial u_i}{\partial t}\right)$ and convective acceleration $\left(u_j \frac{\partial u_i}{\partial x_j}\right)$ which corresponds to transport of momentum by velocity field. The first term in the right-hand side of the equation signifies the forces due to the difference in the pressure in the fluid. The second term is the viscous diffusive term, corresponds to the internal frictional forces in the fluid due to viscosity of the fluid. The last term signifies about the external body forces per unit volume on the fluid, such as gravitational or electromagnetic forces.

- **Conservation of energy:** This principle is based on the first law of thermodynamics, which states that the energy cannot be created nor destroyed, but converted from one form to another. This is represented as:

$$\rho \left(\frac{\partial U}{\partial t} + u_j \frac{\partial U}{\partial x_j} \right) = \sigma_{ji} \frac{\partial u_i}{\partial x_j} - \frac{\partial Q_i}{\partial x_i}. \tag{2.3}$$

where,

- U — internal energy per unit mass
- σ_{ji} — stress tensor component
- Q_i — heat flux vector component in the i^{th} direction

The first term on the left-hand side of the equation corresponds to the rate of change of internal energy, per unit volume of the fluid. This also includes temporal change $\left(\frac{\partial U}{\partial t}\right)$ and convective transport of internal energy $\left(u_j \frac{\partial U}{\partial t}\right)$. The first term in the right-hand side of the equation denotes the rate of work done by the normal and shear stresses in the fluid, accounting for the energy added or dissipated due to the deformation. The last term corresponds to net heat flux leaving the control volume.

These governing equations are solved by discretizing the continuum into finite number of volumes (cells / elements) so that the continuous system of PDE can be converted to a set of discretized algebraic equations that are solved by various numerical techniques. In commercial software Star-CCM+, the discretized equations are solved in the following algorithms, which are discussed below.

2.1.1 Segregated Solver

The segregated solver solves the mass and momentum conservation equations sequentially. The momentum equations are solved first to get the velocities, and these values are used to calculate the pressure correction equation to ensure continuity. The pressure and velocity are recoupled based on Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) or SIMPLEC algorithm and using a collocated variable arrangement like Rhie-Chow interpolation to prevent decoupling of velocity and pressure [5]. When the pressure-velocity equations are solved for the current iteration, the segregated energy solver marches the temperature field. This updated temperature field feeds the buoyancy source for the next momentum step, closing the iteration.[5]

The segregated solver is used typically in low-Mach buoyancy driven flows where the characteristic velocity is only few centimetres per second [5]. Under these conditions, the density field influences the momentum equation only as a buoyancy source term and the density variations are not high.

2.1.2 Coupled Solver

Unlike the segregated solver, the coupled solver treats the continuity, momentum, and energy equations as a single system. The primitive variables are updated simultaneously from a single large linear solve. Because the pressure gradient and velocity divergence terms remain implicit, the mass conservation is forced inside each linear solve and no external algorithm such as SIMPLE pressure correction is required.
The tight variable coupling makes this method more robust when the pressure waves, density variations and stiff source terms dominate the physics [5]. Typical examples include compressible flows, and conjugate heat transfer simulations with strong thermal expansion. The simultaneous treatment of continuity and momentum accommodates large pressure gradients without the extensive under-relaxation that a segregated solver would need, and it also converges in a far fewer global iterations, which can be observed in Section 3.2.3. Since the density, pressure, and temperature are all solved together, this solver consumes more memory and takes longer to calculate each iteration compared with the segregated solvers.

2.1.3 Turbulence Model

Turbulent flows are inherently unsteady and three-dimensional. In these flows, the large eddies carry the most kinetic energy. These eddies break down into smaller eddies in a cascading process, transferring the energy to the smaller scales. Finally, the smallest eddies dissipate the energy into heat due to the viscosity of the fluid. Due to this, the velocity at a point in a turbulent flow consists of two components: mean and fluctuating part such that.

$$v_i(x,t) = \overline{v}_i(x) + v_i'(x,t), \tag{2.4}$$

where $\overline{v}_i(x)$ is the mean velocity component and $v_i'(x,t)$ fluctuating velocity component in i^{th} index. Using these decompositions in the Navier-Stokes equation gives,

$$\rho \underbrace{\frac{\partial \overline{v}_i}{\partial t}}_{\text{transient term}} + \rho \underbrace{\overline{v}_j \frac{\partial \overline{v}_i}{\partial x_j}}_{\text{convection term}} = \underbrace{-\frac{\partial \overline{P}}{\partial x_i}}_{\text{mean pressure gradient}} + \underbrace{\mu \frac{\partial^2 \overline{v}_i}{\partial x_j \partial x_j}}_{\text{molecular viscous stress}} - \underbrace{\frac{\partial}{\partial x_j}\left(\rho \overline{v_i' v_j'}\right)}_{\text{Reynolds-stress divergence}} . \tag{2.5}$$

The last term, which contains the Reynolds stress tensor, accounts for the momentum transport by the turbulent eddies. Eq.2.5 is not closed because the stress tensor consists of six additional unknowns. This makes the equation to have more unknowns than number of equations, thus causing a closure problem. Therefore, the Reynolds stress tensor must be modelled to overcome this closure problem. One such approach is Boussinesq approximation which replaces the Reynolds stress tensor with the turbulent viscosity, ν_t with an assumption that the anisotropic stress tensor is proportional to the mean strain rate. But this approach does not give enough information on how to calculate turbulent viscosity. The standard K-ε model is introduced to overcome this issue.

2.1.3.1 Standard k-ε Model

Since the Boussinesq approximation [4] does not provide enough solution for turbulent viscosity, ν_t. The Standard k-ε model supplies two additional transport equations apart from continuity and momentum equations, one for turbulent kinetic energy Eq.2.7 and another for dissipation rate ε, Eq.2.8. By solving these equations, the turbulent viscosity is calculated from Eq.2.6, thereby closing the system.

$$\nu_t = C_\mu \frac{k^2}{\varepsilon}, \quad C_\mu = 0.09 \tag{2.6}$$

$$\frac{\partial k}{\partial t} + \overline{v}_j \frac{\partial k}{\partial x_j} = 2\nu_t \, \overline{s}_{ij} \, \overline{s}_{ij} + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G_k - \varepsilon \tag{2.7}$$

$$\frac{\partial \varepsilon}{\partial t} + \overline{v}_j \frac{\partial \varepsilon}{\partial x_j} = \frac{\varepsilon}{k}\left(c_{\varepsilon 1} P_k - c_{\varepsilon 2}\varepsilon + c_{\varepsilon 3} G_k\right) + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] \tag{2.8}$$

where,

- k — turbulent kinetic energy.
- ε — viscous dissipation rate.
- $\overline{s}_{ij}$ — mean strain-rate tensor.
- $P_k = 2\nu_t \, \overline{s}_{ij} \, \overline{s}_{ij}$ — production term due to mean velocity gradients.
- G_k — production term due to buoyancy effects.
- ν_t — eddy (turbulent) viscosity.
- $(C_\mu, C_{\varepsilon 1}, C_{\varepsilon 2}, \sigma_k, \sigma_\varepsilon) = (0.09, 1.44, 1.92, 1, 1.3)$ are the empirical constants.

Physically, the ratio of specific turbulent kinetic energy, k and dissipation rate, ε defines the characteristic eddy time scale and the constant C_μ represents the efficiency with which they transport the momentum of the eddies. But this approach can sometimes yield negative stresses especially on stagnation or in recirculation regions which violates the realizability constraint $\overline{v_i' v_i'} \geq 0$ [4].

2.1.3.2 Realizable k-ε model

The realizable k- ε model is introduced to mitigate the shortcomings of the standard k-ε model. The Boussinesq assumption is defined as in Eq.2.9, where λ_1 is the largest eigenvalue of the mean strain tensor. For a stagnation or recirculating area of the flow, this stress tensor can go negative if s_{11} becomes large. One such remedy is to identify the largest eigenvalue and substitute it for s_{11}.

$$\overline{v_i' v_i'} = \frac{2}{3}k - 2\nu_t \overline{s}_{11} \tag{2.9}$$

By rearranging Eq.2.9, an explicit upper limit for turbulent viscosity is obtained, as shown in Eq.2.10

$$\nu_t \leq \frac{k}{3|\lambda_1|} = \frac{k}{3}\left(2 \, \overline{s}_{ij} \, \overline{s}_{ij}\right)^{1/2} \tag{2.10}$$

This upper bound refinement is purely algebraic, so the computational overhead does not differ from the standard $k-\varepsilon$ model, yet the model exhibits markedly improved behaviour in the stagnation or recirculation zones and plume-dominated natural convection flows in the fluid domain. Therefore, the realizable $k-\varepsilon$ model is utilized in this thesis, combining numerical robustness of a two-equation eddy viscosity model with a level of physical consistency that is essential for predicting buoyancy driven stratification or recirculation accurately.

2.1.4 Heat Transfer

This section provides a short introduction to basic concepts of heat transfer. Heat transfer occurs because of temperature difference between two points. There are three ways in which heat transfer can occur between two points. The discussions in this section are followed as mentioned in [6]

- **Conduction** — Heat transfer within a medium when a temperature gradient exists in a stationary medium.
- **Convection** — Heat transfer between a moving fluid and a stationary fluid or two moving fluids when there is a temperature difference.
- **Thermal Radiation** — Transfer of heat energy through in the form of electromagnetic waves emitted by the surface at higher thermal energy.

2.1.4.1 Richardson Number

The Richardson number is a non-dimensional parameter that is used to identify if the flow is natural convection or inertia-dominated. It characterizes the importance of the buoyancy forces to the inertial and shear forces in the flow field. The Richardson number is defined as:

$$Ri = \frac{g\,\beta\,\Delta T\,L}{U^2} \tag{2.11}$$

where,

- β – coefficient of thermal expansion $(1/K)$,
- ΔT – characteristic temperature difference (K),
- L – characteristic length scale (m),
- U – characteristic velocity scale (m/s).

For $Ri \ll 1$, the flow is dominated by inertia and shear forces. When $Ri \gg 1$, buoyancy forces are dominant over shear forces in the flow. If $Ri \approx 1$, the flow is under a mixed convection regime, where there is the presence of both inertia and buoyancy forces are significant.

2.2 HVAC: Energy and Control

Understanding the power consumption of the HVAC unit is important to optimize the performance and reduce the energy consumption. Modern HVAC units contain numerous

components that serve specific purposes in heating, cooling and ventilation cycles, and each component contributes to the overall energy consumption of the system.
The prominent energy-consuming process in an HVAC unit is to heat or cool the volume of the air in the room to the target temperature. The target temperatures are often based upon the thermal comfort or operational needs. The amount of thermal energy required to heat the room to the set temperature is determined by Eq.2.12

$$Q = \dot{m} \cdot c_p \cdot \Delta T \tag{2.12}$$

where,
Q – heat energy required (W)
$\dot{m}$ – mass flow rate of the air (kg/s)
C_p – specific heat capacity of the air (J/kg $\cdot$ K)
ΔT – temperature difference between supply air and the set temperature

If the HVAC unit is cooling, Q is removed from the system and for heating, it supplies Q energy to the system. This energy forms the fundamental basis on how much electrical energy is consumed, depending on the efficiency of the heater, pump, or chiller. For electric resistance heaters, the electrical energy consumed is almost equal to the thermal energy required. For the case of chillers, the electrical energy consumption is calculated using the coefficient of performance (COP), which is defined as,

$$P_{\text{electrical}} = \frac{Q}{\text{COP}} \tag{2.13}$$

where the COP represents how many units of heat is removed per unit of electricity consumed. Generally, for a HVAC unit serving a room size of $60m^3$, the COP is in the range of 3.2 to 4.0 [21].
Aside from heating/cooling the room, the air must be recirculated throughout the space. This is performed by the ventilation fan. As the fan operates, it creates a pressure drop across the fan, allowing the air to be drawn in and pushed out. Thus, to maintain a certain airflow against the pressure drop, the fan must do work and this is measured as power. The power required by a fan to push the air is given by,

$$P_{\text{fan}} = \frac{\Delta P \cdot \dot{V}}{\eta} \tag{2.14}$$

In real systems, this pressure drop depends on flow rate, and it is not constant and nonlinear. The fan curve defines the relationship between the mass flow rate and the static pressure drop. Fig.2.1 illustrates the fan curve for an office building HVAC system by Mitsubishi Electric [22].
Using Eq.2.14 and the fan curve, the energy consumption due to fan for the specified mass flow rate.
Conversion from Air Change Cycle (ACH) to mass flow rate: The ACH represents how many times the total volume of air is replaced in an hour. The mass flow rate is calculated as:

$$\dot{m} = \frac{\rho \cdot V \cdot \text{ACH}}{3600} \tag{2.15}$$

where V is the volume of the room in m^3. The ACH values in this thesis were chosen to represent different ventilation scenarios. The resulting flow rates are used as input boundary condition values for inlet vents.

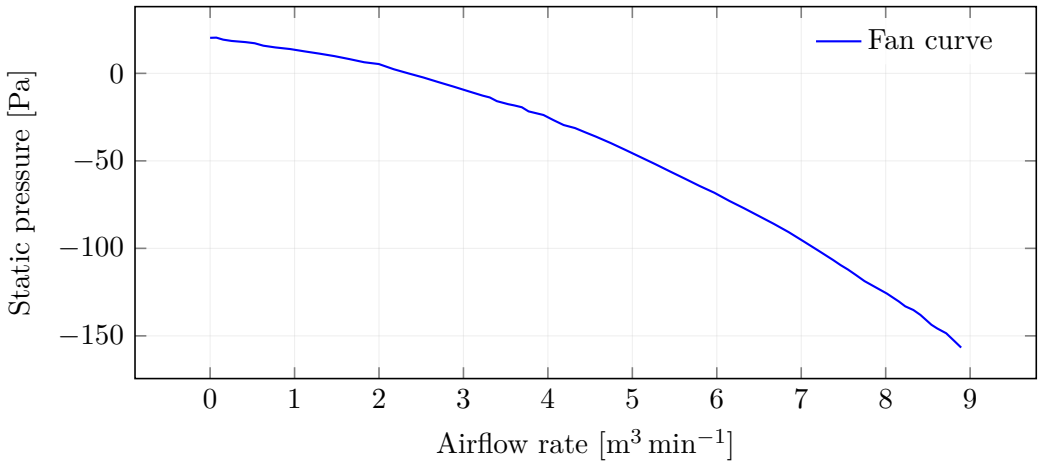


Figure 2.1: Fan performance curve showing the decrease in static pressure with increasing airflow rate [22].

2.3 Machine Learning

Machine learning is part of the Artificial Intelligence, consisting of algorithms which systematically improve their performance through experience. The core ideology behind machine learning involves training algorithms to identify the patterns and distribution within the training data, enabling predictions without any explicit programming [7]. Machine learning algorithms are commonly classified as supervised learning, unsupervised learning, and reinforcement learning.

2.3.1 Supervised Learning

Supervised learning is a class of machine learning method which is used to build a predictive model based on previous simulations or experimental data. It involves in training the model, which contains an input feature vector along with its corresponding target value or an output vector. The model generates initial predictions for the given inputs, and these predictions are compared with the untrained ground truth data. The difference between the prediction and ground truth is calculated by a loss function, which is then used to evaluate the performance of the current parameters. An optimization algorithm, such as gradient descent, tunes the model parameters such as weights and bias of nodes to minimize the error [8]. Through back-propagation, the error is propagated backwards through the network using chain rule. The chain rule calculates the gradients of the loss with respect to each parameter. This process is repeated until prediction values are closer to the ground truth.
The input features can include geometric specifications, boundary conditions for the simulation or turbulence parameters and their corresponding target can be velocity fields, pressure distribution or temperature gradients [9]. The objective of supervised machine learning is to build a computational model that correlates the input vectors and their corresponding target vectors. Thus, suitable hyperparameters such as number of nodes, hidden layers, batch size etc. should be chosen for machine learning model to yield the outputs that are closer to the known targets or ground truth [10]. Supervised learning can be broadly classified into two main categories:

- **Classification:** Classification model assigns the data points into predefined, dis-

crete classes or groups for a given input data. Some of the common classification algorithm include decision trees, support vector machines (SVM), Convolutional Neural Networks (CNN) logistic regression, random forests etc., The classification algorithms are commonly used in image classification, disease detection based on imaging data and spam email filters [11].

- **Regression:** Unlike classification algorithms, regression models predict continuous numeric outcomes such as temperature inside a room, pressure distribution across a turbine blade or drag force prediction of a bluff body. Commonly used regression methods include polynomial regression models and various neural network architectures such as Convolutional Neural Network (CNN), Fourier Neural Operators (FNO) and fully-connected neural networks. These neural networks can be trained on CFD generated data that can rapidly predict flow fields of complex geometries, significantly reducing the computational resources needed for repetitive numerical simulations [12].

2.3.2 Multi-Layer Perceptron

The Multi Layer Perceptron (MLP) is one of the most commonly used architectures in neural networks [18]. It is the building block of many deep learning models. MLP is a type of feed-forward neural network which contains multiple layers of nodes, also called as neurons. These nodes are fully connected to the nodes in the next layer. A fully connected layer is one in which every node in a layer has a weighted connection to every node in the subsequent layer, allowing the data to be passed through all the nodes. MLP is often used in regression and classification problems, and they are also used as an immediate component for CNNs and FNO [9] [11]. An MLP is made up of an input layer, one or more hidden layers and an output layer. Each layer consists of multiple nodes in which each node performs linear transformation, followed by a non-linear activation function. The output of the first layer is computed as:

$$h_1 = \sigma\left(W_1\vec{x} + b_1\right) \tag{2.16}$$

where,

$\vec{x}$ – Input vector to the neural network layer.
W_1 – Weight matrix of the first layer.
b_1 – Bias vector of the first layer.
σ – Nonlinear activation function (e.g., ReLU, sigmoid, tanh).
h_1 – Output of the first hidden layer after applying activation.

The key theoretical result for MLP is the universal approximation theorem, which states that, given a sufficient number of nodes in a hidden layer, the neural network can approximate any continuous function with desired accuracy. In operating frameworks like FNO, MLP is used as a lifting layer to transform low-dimensional input to a high-dimensional representation of the input, and map high-dimensional features back to the output space. The lifting layer projects the raw input into a richer feature space [9]. This helps the FNO to capture the localized and pointwise features in the data.

2.3.3 Activation Function

Activation functions determine the output of the individual nodes in a layer. This helps the machine learning model approximate non-linear relationships between input and target

data. Without any activation function, the machine learning model behaves in a purely linear way, which largely limits the ability of neural networks [7]. Some of the commonly used activation functions for regression machine learning models are discussed below.

- **ReLU (Rectified Linear Unit):** This is one of the most commonly used activation functions in neural networks [18]. This is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{2.17}$$

  This function gives the output of the nodes if the values are positive, or returns zero otherwise. This helps to avoid the "vanishing gradient" problem [19]. Vanishing gradient occurs during backpropagation, where the weights are updated back through the neural network. The gradients can diminish to very small values as they move towards the input layer. This slows down the training process significantly. However, this activation function can lead to "dead nodes", because its output of the activation function is zero for all negative input values to that node.

- **Leaky ReLU:** This is a slight modification of the ReLU. Instead of zeroing all the negative outputs, this function allows a smaller negative slope of a specified value. The Leaky ReLU function is defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{2.18}$$

  where,

  – x – Input to the activation function.

  – α – Small positive constant which controls the slope for negative input values.

  This modification eradicates the dead-node issue observed from the ReLU activation function. Thus, the nodes can still learn even when they receive negative inputs.

- **ELU (Exponential Linear Unit):** ELU aims to resolve the issue of dead neurons, similar to Leaky ReLU. In contrast with Leaky ReLU, ELU introduces a smooth exponential curve in the negative region instead. The ELU function is defined as:

$$\text{ELU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases} \tag{2.19}$$

  This smoother transition enables smoother gradient flow, which helps gradient propagation. ELU is more computationally expensive than the other activation functions, but it provides better performance in the neural networks [20].

2.3.4 Loss Function

The loss function is one of the most critical components in the machine learning methods. It provides a measure on how close the machine learning model's predicted output closer with the ground truth. This helps in guiding the learning process by informing the optimizer how to adjust the model parameters. The objective of the training of a machine learning model is to find a set of model parameters, θ, that minimize the loss over the dataset.

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i) \tag{2.20}$$

where,

θ	– Model parameters to be optimized
θ^*	– Optimal parameters that minimize the loss
N	– Total number of training samples
y_i	– Ground truth for the i-th sample
$\hat{y}_i$	– Predicted output for the i-th sample
$\mathcal{L}(y_i, \hat{y}_i)$	– Loss function measuring the difference between y_i and $\hat{y}_i$

Two commonly used loss functions which serve specific purposes, and they are discussed below:

- **Mean Squared Error (MSE):** This is one of the most commonly used loss functions in the regression-based machine learning methods. It calculates the average of squared differences between the predicted value and the ground truth. The formulation of the loss function is mentioned below:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.21}$$

  The squared error term grows quadratically. It penalizes the total loss disproportionately due to larger loss values. This makes MSE an ideal choice when large deviations are undesirable.

- **Mean Absolute Error (MAE):** Unlike MSE, MAE computes the average of absolute differences between predicted values and the ground truth. The formulation of the MAE is defined as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{2.22}$$

  This loss function linearly penalizes absolute errors. This function, however, is not differentiable at zero, which means less stable and slower than MSE.

2.3.5 Convolution Operator

The convolution operation is a mathematical operation that performs linear transformations acting on functions, producing a third function which reflects the interaction between two input functions. This operation characterizes on how one function is modulated by the influence of another function, called kernel function. This operation is typically used in image analysis and deep learning architectures such as CNNs and FNO [11]. For a continuous function f and g, the convolution, the convolution h is defined as:

$$h(t) = (f * g)(t) \tag{2.23}$$

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)\, g(t - \tau)\, d\tau. \tag{2.24}$$

One of the fundamental theorems for convolution operation is that the convolution of two functions in the physical domain is equivalent to their element-wise multiplication in Fourier space.

2.3.6 Fourier Neural Operator

The theory provided in this section is primarily based on the work of Li et al. (2020) [9]. The main objective of the Fourier Neural Operator (FNO) is to learn the solution operator of a partial differential equation. Instead of solving the PDE for each new input, the FNO learns the function that correlates the input vectors, such as boundary conditions and spatial locations, directly to the target vectors. This is done in a way that it generalizes across different spatial resolutions.

The PDE equation defines a rule that transforms one function, such as initial or boundary condition, into another function such as the PDE solution. This process of transformation is called an operator. The primary goal of the FNO is to train this operator using training data consisting of input to output pairs.

Consider a spatial domain, D which is represented by Cartesian coordinates X, Y and Z if this is a three-dimensional space. The input function is $a(x)$, where, $x \in D$ and the output solution is $u(x)$. $G^{\dagger}(a)$ is the operator that maps a to u. The FNO then approximates this operator by a neural network model, G_{θ}, where θ represents the parameters learned during training. A loss function is utilized to measure how far the predictions $G(a)_{\theta}$ are from the ground truth $G^{\dagger}(a)$, and the network is trained by minimizing the losses in the loss function. This optimization helps the model to generalize to new and unseen inputs [9].

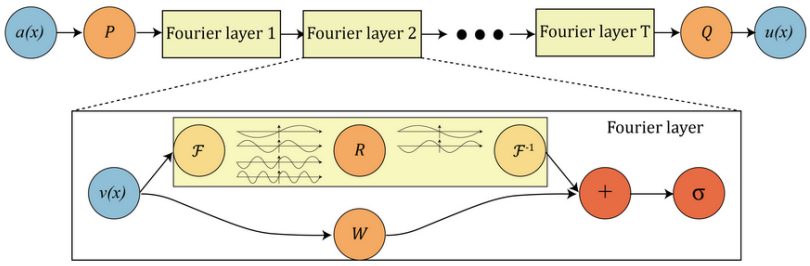


Figure 2.2: top: The architecture of the neural operators; bottom: Fourier layer. [9]

The architecture of FNO processes the input function in multiple stages, each designed to update and refine an internal representation. Each layer in the neural network performs two main operations:

Local Transformation: In FNO, the local transformation is the key operation performed at each layer and at each spatial location. This corresponds to the pointwise linear transformation independent of the global spatial domain. Consider the current feature representation at layer t is defined as $v_t(x) \in \mathbb{R}^{d_v}$ where the spatial location, $x \in D$. Here, $v_t(x)$ is the feature vector of d_v dimensions in real space. The trainable weight matrix which is shared across all spatial locations is defined as $W \in \mathbb{R}^{d_v \times d_v}$. The linear transformation can be expressed as $\sigma W v_t(x)$, where σ is the activation function for the Fourier layer.

Global Interaction: In conventional deep learning methods such as Convolutional Neural Networks (CNN), the output is computed based on small neighbourhood of inputs. This approach works well with image processing, but cannot model the physical systems governed by PDE, where long range dependencies are often critical [9]. The global interaction in FNO is achieved by using the integral operator. This operator updates the representation of the solution at each point by averaging the data from all the points in the domain. Unlike CNNs, the update on the location x depends on a weighted combination of the representations $v_t(x)$ at all other points in the domain. This process is expressed in Eq.2.25

$$(\mathcal{K}(a;\phi)v_t)(x) = \sigma \int_D \kappa_\phi(x, y, a(x), a(y))\, v_t(y)\, dy. \tag{2.25}$$

where $\kappa_\phi(x, y, a(x), a(y))$ is the kernel function which provides the internal representation of the solution of each point in the domain. The kernel function is a weightage function which learns from the data, and it is capable of adapting to its spatial coordinates and input parameters. The kernel is parametrized as a neural network and, it is trained alongside the FNO model, allowing it to capture the complex spatial interactions.
To enable faster computation, FNO makes a critical assumption that the kernel function is translation - invariant, which means that the interaction between the points depends only on their relative distance. This allows the Eq.2.25 to be simplified into a convolution:

$$(\mathcal{K}v_t)(x) = \int_D \kappa(x - y)v_t(y)\, dy = (\kappa * v_t)(x). \tag{2.26}$$

The '*' in Eq.2.26 denotes the convolution operation between κ and $v_t(x)$. This convolution operation calculated in Fourier space using the convolution theorem. This process is computationally efficient, as the convolution operation is equivalent to multiplying in Fourier space. Utilizing this theory, the FNO performs this global operation by transforming the features into Fourier space using FFT. The learned weight tensor (R), is applied to the transformed features and the results are transformed back to the physical space using inverse FFT. Thus, Eq.2.26 can be written as:

$$(\mathcal{K}v_t)(x) = \mathcal{F}^{-1}\left(R \cdot \mathcal{F}(v_t)\right)(x). \tag{2.27}$$

here, $\mathcal{F}$ and $\mathcal{F}^{-1}$ corresponds to the Fourier transform and inverse Fourier transformation. R is the learned complex valued weighted function for each Fourier mode. The FNO utilizes only low-frequency modes in the Fourier domain. These low frequency modes contain information about the overall structure of the target solution. The model can still recover the higher frequency information by the use of non-linear activation function and additional Fourier layers through mode interactions [9].
Together with the local linear transformation and the global transformation, the update for the solution space for the given layer is written as:

$$v^{t+1}(x) = \sigma\left(Wv^t(x) + \int_D \kappa(x, y)v^t(y)\, dy\right) \tag{2.28}$$

Since FNO's ability to handle functions of different resolutions, the model can be trained on coarse grids and still perform well on fine grids.

2.3.7 Reinforcement Learning

Reinforcement learning is specifically geared towards active control and optimization tasks. Unlike supervised learning methods, reinforcement learning develops decision-

making strategies through continuous interaction between an "agent" (a controller) and an "environment" (the fluid domain or flow configuration). The RL agent explores possible actions, such as modifying the boundary conditions or applying control parameters to actuators [13]. The environment then responds by transitioning into new fluid states, generating outcomes such as pressure and velocity distributions, and subsequently providing feedback in terms of rewards or penalties based on performance criteria. The Reinforcement Learning algorithms are broadly classified into two categories [14] based on how they utilize received rewards, as discussed below.

- **Policy Based Methods:** The policy in reinforcement learning refers to the strategy or decision-making rules that the agent uses to select the appropriate actions for the current state. These policies are typically represented in mathematical functions or in neural networks which directly map the current state of the environment, such as pressure fields or temperature distribution inside a room. The learning process in policy based algorithms generally tunes the parameters in the policy network to maximize the cumulative reward received during interactions. Some of the prominent algorithms are Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO). Policy based methods optimize policies directly through repeated simulation trials; they usually require more computational resources and significant amount of simulation time [15].

- **Off-Policy Based Methods:** In contrast with policy-based methods, these algorithms explicitly learn from past interactions through buffer memory of past actions and the corresponding states of the environment, rather than depending on the interactions guided by the current policy. This ability to reuse of buffer data reduces computational cost significantly and improves learning efficiency. These algorithms typically involve in learning action-value functions or Q-functions. This Q-function is then used to predict the cumulative reward an agent would receive by providing a specific action in a given state and then following an optimal policy afterwards. The agent can identify optimal decisions effectively without necessarily running additional expensive simulations repeatedly [23]. Some off-policy methods are Q-learning, Deep Q-Networks and Soft Actor Critic method [16] [17].

2.3.8 Soft Actor Critic (SAC)

Soft Actor-Critic (SAC) is a model-free RL algorithm which is used to solve for continuous action problems instead of choosing from set of discrete options. This is also called a model-free algorithm, meaning that it does not have to learn the exact model of the environment. The theoretical foundation of SAC discussed in this section are based on the work of Haarnoja et al. [23]
The primary theory behind SAC is the principle of **maximum entropy reinforcement learning,** which means the algorithm tries to remain as unpredictable as possible rather than trying to find a way to earn the most reward. This is achieved by introducing an entropy term into the reward function, and it makes the model to explore a wider range of actions [23]. This enables the model to perform well in complex environments where they are uncertain and noisy by making the training process more stable, leading to more robust policies. The entropy term also balances exploration and exploitation of the action values. Exploration is when the agent tries new random actions and evaluate the outcomes. This helps in the model to find better solutions for the environment. In contrast, exploitation

is when the agent predicts the action values based on past learning experience, producing more deterministic outputs.

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\, r(s_t, a_t)\right] \tag{2.29}$$

where,

$J(\pi)$	– Total score (expected return) of a policy π
$\mathbb{E}_{\tau\sim\pi}$	– Expectation over trajectories τ (sequences of states and actions) sampled from policy π
γ^t	– Discount factor raised to time step t, where $\gamma \in [0, 1]$
$r(s_t, a_t)$	– Reward received at time t after taking action a_t in state s_t
s_t	– State of the environment at time step t
a_t	– Action taken by the agent at time step t

For the SAC algorithm, Eq.2.29 is modified by adding an entropy term to the reward. The new score for the policy becomes:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))\right)\right] \tag{2.30}$$

where,

α	– Entropy temperature coefficient that balances reward and entropy
$\mathcal{H}(\pi(\cdot \mid s_t))$	– Entropy of the policy at state s_t, encourages exploration
$\pi(\cdot \mid s_t)$	– Probability distribution over actions given state s_t

The temperature coefficient in the entropy term controls the trade-off between exploration and exploitation for predicting the action values. In optimization of high reward and high entropy, SAC, updates the policies to be adaptable to dynamically varying situations. This leads to greater training ability and enhances the robustness of the control policy. SAC utilizes actor-critic architecture where the actor is the policy, π, which determines the actions to be taken for the current environment. The critic network evaluates the performance of the action in terms of the soft Q-function, $Q^\pi(s, a)$ which includes both expected reward and the entropy term, to encourage exploration of the model. The soft Bellman Q-function is defined as:

$$Q^\pi(s, a) = r(s, a) + \gamma\, \mathbb{E}_{s'\sim p,\, a'\sim\pi}\left[Q^\pi(s', a') - \log\pi(a' \mid s')\right] \tag{2.31}$$

where.

$Q^\pi(s', a')$	Estimated future return from next state-action pair.
$\log\pi(a' \mid s')$	Log-probability of action a' under policy π; penalizes deterministic behaviour.

The SAC algorithm uses three neural networks to approximate the policy and value functions. a policy network (actor), two Q value critic network and its corresponding target network. The minimum of the Q values of the two critic networks when computing the target for Bellman backup to mitigate the overestimation bias which is faced in single Q-network updates. The Q - networks are trained based on Mean Squared Error (MSE) of the trained Q value and the target Q-value. For each critic, $i \in \{1, 2\}$ the loss function is calculated as,

$$J_Q(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_{\theta_i}(s, a) - \hat{Q}(s, a)\right)^2\right] \tag{2.32}$$

and the target Q-value is defined as,

$$\hat{Q}(s,a) = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi} \left[\min_{j=1,2} Q_{\theta_j}(s', a') - \alpha \log \pi(a'|s') \right] \tag{2.33}$$

The actor network is trained to maximize the expected return, while it is able to increase its entropy. This is accomplished by reducing the difference between entropy term and minimum of critic loss. This can be expressed as:

$$J_\pi(\phi) = \mathbb{E}_{s \sim \mathcal{D},\, a \sim \pi_\phi} \left[\alpha \log \pi_\phi(a|s) - \min_{j=1,2} Q_{\theta_j}(s,a) \right]. \tag{2.34}$$

In general, two critic functions are used and the minimum of two functions is used to update the policy and value function. Based on the loss values, the weights in target value function ($V_{\bar{\psi}'}$) of critic networks are then updated using exponential moving average which is represented in Eq.2.35

$$\bar{\psi} \leftarrow \tau \psi + (1 - \tau) \bar{\psi} \tag{2.35}$$

where,

- $\bar{\psi}$ is the parameter of the target network,
- ψ is the parameter of the current (online) network,
- $\tau \in (0, 1)$ is the soft update rate (typically a small value like 0.005),

This rolling update method, utilization of twin Q - functions and stochastic policy coupled with entropy - regularized objective, makes SAC to provide stable control solutions along with good balance between exploration vs exploitation even in complex environments, making it well suited for real-world reinforcement learning application.

2.3.9 PID Controller

The proportional - Integral - Derivative (PID) controller is widely used in automation systems for regulation. It is a feedback-based control algorithm which uses error, defined as the difference between set point and measured value of the process variable. The controller reduces the error over each time step by providing an adjusted control input to the system. The PID controller in general has three distinct components-proportional, Integral and Derivative and each of these components contributes to the control input in their own way, as described below. The theoretical foundation of SAC discussed in this section are based on the work of Li.[25]

2.3.9.1 Proportional Term

The proportional component of the controller estimates the control action proportional to the instantaneous error. This term is expressed as:

$$u_P(t) = K_p \cdot e(t) \tag{2.36}$$

where,

- $u_P(t)$: Proportional control signal
- K_p: Proportional gain

- $e(t) = r(t) - y(t)$: Error between the reference signal and the process output.
- $r(t)$: Target value
- $y(t)$: System Response

This term provides an immediate response to the current magnitude of the error. However, this also causes unsteadiness in the controller, which the other term helps to reduce.

2.3.9.2 Integral Term

The integral term estimates the cumulative error over time, which helps the controller to identify the steady-state error. The integral term is given by

$$u_I(t) = K_i \int_0^t e(\tau)\, d\tau \tag{2.37}$$

Where K_i is the integral gain term. Excessive integral output can lead to instabilities and overshoot, especially in a system with slower dynamics.

2.3.9.3 Derivative Term

The derivative component introduces control input based on the rate of change of error. The derivative term is expressed as:

$$u_D(t) = K_d \frac{de(t)}{dt} \tag{2.38}$$

Where K_d is the derivative gain. The derivative term helps in dampening the oscillations in the response curve and slowing the rate of change of error over time. This helps in stability of control system. Excessive derivative influence will lead to erratic control actions when it is not filtered appropriately.

2.3.9.4 Combined PID Control

The complete PID control output is the summation of all the three control components, which is expressed as:

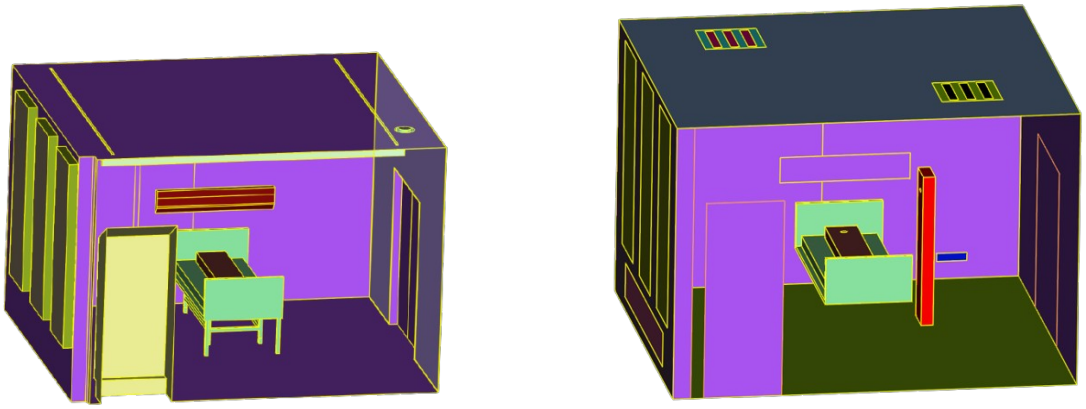
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{2.39}$$

PID controllers are effective for systems where the dynamics are well understood and that can be approximated by linear models. They are the most commonly used control system algorithm due to their ease of implementation and wide range of applications [24].

3
Methods

3.1 Room Geometry and Physical Specification

The computational domain represents a hospital room, which is used to simulate the thermal and airflow behaviour using CFD and machine learning. The initial geometry is provided by AFRY which is illustrated in Fig.3.1a. This geometry had detailed features, including multiple small openings and intricate structures. While these intricate details seem realistic, they do not significantly influence the flow field inside the room. However, these details increase the computational cell count and the overall simulation time significantly. Therefore, to improve the mesh quality and reduce the cell count, simplifications were made to the domain. It can be noted from the Fig.3.1b, minor details such as door and window frames, ceiling fixtures, vents projection and legs of bed were removed to avoid generating smaller cells.



(a) Initial Room Geometry **(b)** Simplified Room Geometry

Figure 3.1: Comparison of Initial and Final Room Geometries

The geometry is modified with simplified block representations in the ANSA pre-processing software as shown in Fig.3.1b. Additionally, rectangular box is introduced to represent a second human figure. This allows to simulate the thermal obstruction and possibilities to simulate more complex CFD simulations by including radiation generated from the people and CO_2 emissions. The ceiling-mounted air inlet vent was simplified in shape to avoid excessively refined mesh regions, thereby speeding up the CFD simulations. Additionally, smaller gaps that were found in the initial geometry were closed to eliminate the need for thin skewed cells during meshing.
Fig.3.2 represents different boundaries in the room and the size of the domain. The room has the dimensions of 4.055 m along X-axis, 4.66 m along the Y-axis and ceiling

height of 3.2 m in Z-direction. On one of the walls, three evenly spaced windows provide an interface with the environment and simulate weather changes in the environment. A radiator is positioned directly below the windows, acting as a localized heat source within the room. An outlet vent is placed behind the bed to extract the air from the room and ensure proper ventilation. These features are necessary to provide meaningful analysis of HVAC performance inside the room.

3.2 CFD Simulation Setup

This section provides the methodology used to simulate the airflow and thermal conditions within the domain using commercial CFD software, STAR-CCM+. The objective of the simulation is to generate reliable flow field data using CFD simulations to generate necessary data for the machine learning model. The overall approach is to capture essential flow dynamics and temperature fields inside the room while finding a good balance between numerical accuracy and computational speed. Both steady-state and transient simulations are performed to analyse the steady-state solutions and time dependent thermal behaviours. This part of the methodology includes defining the boundary conditions to the domain, preparing the mesh and setting up the solver to obtain converged solutions.

3.2.1 Boundary Condition

This is one of the crucial steps in performing a CFD simulation. Appropriate boundary conditions need to be set to represent realistic HVAC operation, and also to ensure numerical stability and physical consistency in the simulation. The boundary surfaces are assigned a PID (Property ID) in ANSA and this is imported in STAR-CCM+ to assign corresponding physical conditions. This ensures a structured framework for assigning the boundaries. The boundary condition specifications for each surface are detailed below

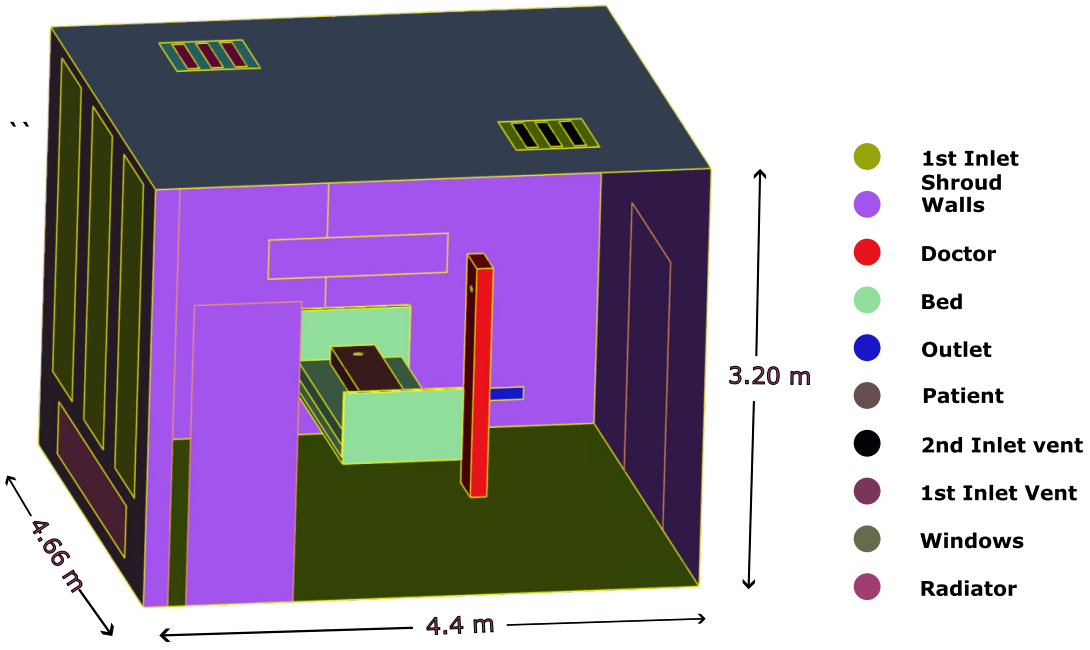


Figure 3.2: Room Geometry with Boundary Labels

- **Walls (bed, doctor, patient, shrouds, and room walls)**
  - Defined as wall boundary condition.
  - Treated as no-slip surfaces to have zero velocity at the boundary surface.
  - Wall surfaces are treated as adiabatic, to simplify the implementation.
- **Heating element (radiator)**
  - Defined as a wall boundary condition.
  - Defined a constant temperature heat source.
  - Temperature values are specified based on different radiator operating conditions.
- **Windows**
  - Also defined as wall boundary condition.
  - Defined constant surface temperatures corresponding to various outdoor environmental conditions.
- **Inlet vents**
  - Configured as mass flow inlet boundaries.
  - Specified with mass flow rates in kg/s, corresponding to different air change per hour (ACH) scenarios.
  - Inlet air temperature is prescribed according to defined HVAC supply conditions as mentioned in Table 3.6.
- **Outlet vent**
  - Defined as a pressure outlet boundary.
  - The outlet pressure is set to atmospheric pressure, allowing free outflow from the domain.

3.2.2 Mesh Generation

The quality of the mesh directly influences the resolution of flow features, computational cost, and overall performance of the simulation. The mesh for this simulation is generated using STAR-CCM+, with the primary objective of creating a mesh with enough resolution to resolve key flow structures, especially thermal gradients and near-wall phenomena, without introducing excessive numerical diffusion or instabilities in the solver. A structured workflow was followed to define appropriate global and local mesh sizes, apply local refinements to resolve gradients in the flow field and ensure smooth transition between different cell sizes.

In this project, trimmed cell meshes are used to capture internal airflow. Polyhedral cells have multiple faces (typically: 14), allows room for meshing for complex geometries with fewer cells than tetrahedral cells. However, the polyhedral cells have greater solver overhead because of their unstructured nature and irregular face arrangements. Depending on the geometry, they also reduce orthogonality of the cell faces near boundary layers, which leads to reduced accuracy in solutions due to numerical diffusion [5].

In contrast, the trimmed cell approach produces hexahedral cells (6 faces), trimmed near the boundaries to fit the geometry. These meshes provide a good balance between numerical accuracy and computational efficiency. Because of their structured nature, the trimmed cells increases the orthogonality of the cell faces and this helps in improved solver performance and enabling faster convergence of the simulations [5]. Because of the advantages of the trimmed cell meshes and rectangular domain without any complex shapes, trimmed-cell mesher is chosen for the simulation. Fig.3.3 represents the sectional view of the mesh used for the simulation.

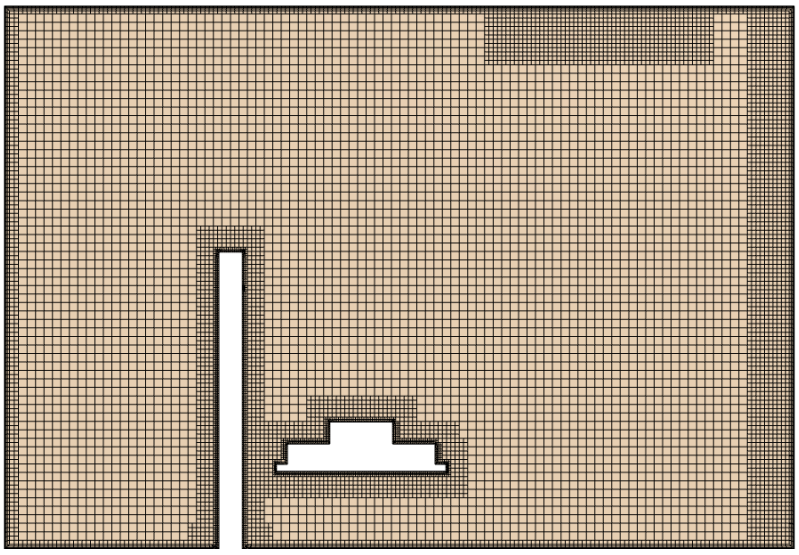


Figure 3.3: Section View-XZ plane of the Mesh

Local mesh refinements are introduced to capture higher convective flows and steep gradients. The base cell size in these local refinements is chosen as even integer multiples of the global base size. This ensures smooth cell transition and prevents abrupt changes in the cell aspect ratio, which can negatively affect the solution accuracy. The base cell size of the window and heating element region was set to 50% of the global base cell size. The region with the highest momentum gradients, outlet vents and inlet vents, the base size is defined as 25% of the global base cell size. Fig.3.4 represents the details of the mesh near the walls and the vent.

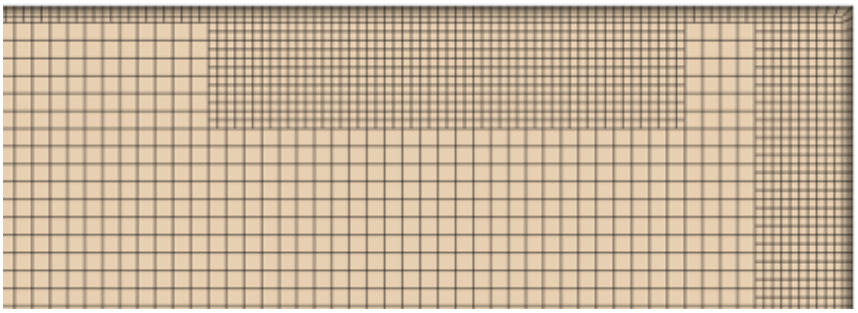


Figure 3.4: Section View-XZ plane of the Mesh
(Magnified view of mesh near the vents)

To capture near-wall behaviour, prism layers are utilized on all the surfaces in the domain. Resolving boundary layers is necessary to calculate shear stress and heat transfer near the

walls. Since Realizable $k-\varepsilon$ turbulence model and two-layer y^+ wall function is utilized, it is sufficient to maintain near-wall $y^+ > 30$. This is because the wall function blends viscous sublayer $y^+ \leq 5$, buffer region ($5 < y^+ < 30$) and lograthmic layer ($y^+ > 30$) to ensure smoother transition across the boundary layer. As the flow around the objects such as the bed, doctor and patient mannequins are not the focus and has limited influence on the overall flow field, the number of prism layers is reduced by half compared to other regions. A comprehensive mesh refinement study is performed to determine the cell sizes, which is detailed in Section 4.1.1. Table 3.1 summarizes the overall mesh settings used for the simulation.

Table 3.1: Mesh configuration parameters applied in the simulation setup.

Parameter	**Value**
Base Cell Size	0.05 m
Surface Growth Rate	1.2
Volume Growth Rate	1.2
Prism Layer Count	4
Boundary Layer Thickness	0.03 m
Near-Wall Thickness	0.0045 m
Total Cell Count	1.8 million

3.2.2.1 Adaptive Mesh Refinement (AMR)

Adaptive Mesh Refinement (AMR) is used to dynamically adjust the mesh resolution based on local flow features during the simulation. Rather than applying uniformly fine mesh throughout the entire domain, AMR selectively refines the mesh where stronger flow gradients are present. This allows for efficient resolution of features such as shear layers, jets, and thermal driven flow. The AMR is employed using STAR-CCM+'s built-in adaption features.

The adaption mode is set to midpoint subdivision, where the refinement operation splits each selected cell into small cells by introducing new cell centres at the midpoint of the parent cell edges. The transition width is set to 3 which controls the number of layers the mesh is refined between fine and coarse mesh regions. This avoids abrupt changes in cell size, which leads to numerical diffusion. The minimum cell size is limited to 50% of the global base mesh size to achieve the required refinement while balancing computational cost. The adaption criteria are defined by two field functions based on the curl of pressure and velocity, which capture rapid variations in flow field. The adaption is defined based on the following equations:

$$\textbf{Pressure Criterion:} \quad |\nabla P| \cdot \Delta x$$
$$\textbf{Velocity Criterion:} \quad |\nabla \times \vec{V}| \cdot \Delta x$$

where,

- $|\nabla P|$ – Magnitude of the pressure gradient
- $|\nabla \times \vec{V}|$ – Magnitude of the vorticity
- Δx – Adaption cell size

The mesh cells are refined when the criterion exceeds the defined threshold and coarsens when it falls below the minimum threshold and remains unchanged when within acceptable

limits. Adaption cell size is the local cell size that is undergoing refinement or coarsening during the AMR cycle. The AMR cycle will calculate only the gradients when it is not scaled with the adaption cell size. This makes the AMR continuously refine in the regions where the gradients grow steeply, producing infinitely small or large cells if no lower or upper limits are applied. This can lead to extremely large or small cell sizes, which affect the solver stability. This criterion will be normalized when scaled with adaption cell size, preventing unnecessary refinement in resolved regions, maintaining a balanced, efficient distribution.

3.2.3 Solver Settings

This section outlines the solver configurations used to perform CFD simulations in STAR-CCM+. The physical settings for CFD simulation were configured to model buoyancy-driven indoor airflow under HVAC operation. The air inside the domain is modelled as an ideal gas, which provides a suitable approximation of air under natural convection flows. These assumptions enable the density to vary with temperature under ideal gas law, making it essential to capture buoyancy effects, which are driven by temperature-induced gradients.
The simulations are carried out under steady-state conditions, which resolves the flow fields in equilibrium. Compared to transient simulations, steady-state modelling is computationally more efficient and appropriate for establishing time-independent distribution of airflow and thermal fields under constant boundary conditions.
Realizable $k - \varepsilon$ turbulence model is chosen for the given nature of the flow inside the room, which is characterized by moderate turbulence and wall-bounded recirculation flows. Gravity is enabled in the simulation to model buoyancy-driven convection accurately. In indoor environments, heat sources, and colder boundary regions can cause temperature variations that interact with gravity to induce vertical motion. The reference pressure is defined at the outlet surface under atmospheric conditions, since the solver calculates for the pressure differences relative to the fixed point where the pressure is constant. This means that all the upstream pressure rises are measured above that datum point.
To evaluate the influence of the solver settings on simulation performance and solution accuracy, both segregated solver and coupled solver were implemented and tested in STAR-CCM+. For direction comparison between solvers, the exact same geometry, boundary conditions, and mesh settings were maintained across both configurations for consistency. The coupled solver is set up with automatic Courant–Friedrich's–Lewy (CFL) number, allowing the solver to adjust the solution update based on the convergence behaviour. The key settings for the coupled solver are listed in Table 3.2
The segregated solver is set up based on SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm. The points below discusses the segregated solver settings:

- Segregated solver with segregated temperature model
- **Pressure–velocity coupling scheme:** SIMPLE (Implicit)
- **Reference pressure location:** Outlet surface (at atmospheric pressure)
- **Convection scheme for momentum:** 2nd-order upwind
- **Convection scheme for temperature:** 2nd-order upwind

Table 3.2: Solver and CFL control parameters applied in the simulation.

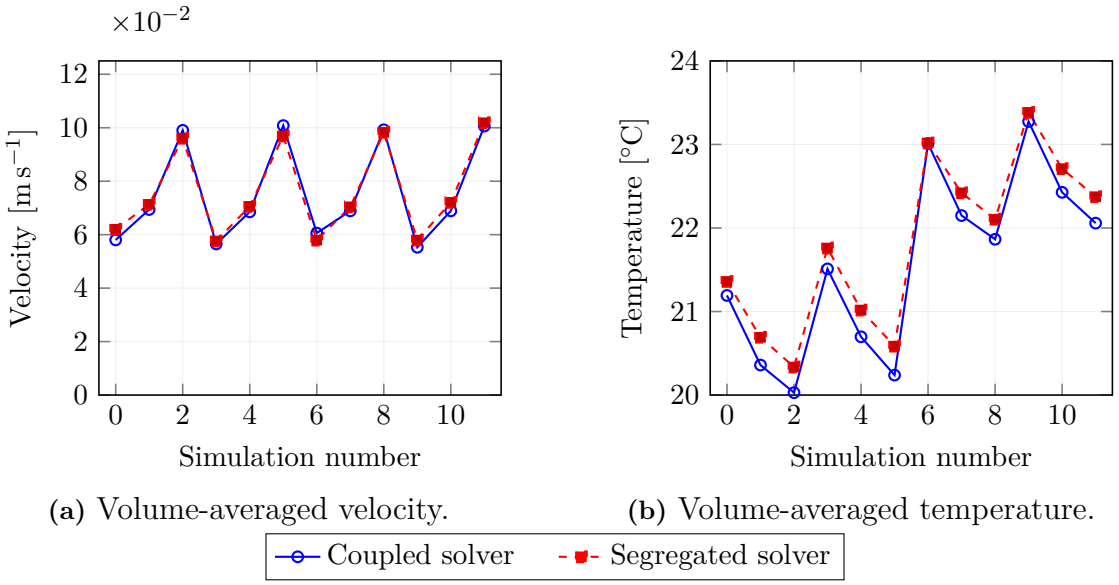
Parameter	Value / Setting
CFL Control	Automatic
Initial CFL Number	1
Minimum CFL Number	0.1
Maximum CFL Number	500
Target AMG Cycles	4
Max Iterations per Multigrid Level	50
Convergence Tolerance	5.00×10^{-4}
CFL for Multigrid Acceleration	5
Continuity Convergence Accelerator	Enabled
Enhanced Stability Treatment	Enabled
Enhanced Mass Imbalance Calculations	Enabled

Fig.3.5b represents the volume average temperature values obtained from coupled and segregated solvers across twelve different simulation cases with different boundary conditions. It can be noted that both the solvers produce consistent trend in temperature values but with slight deviation. The segregated solver predicts slightly higher values compared with coupled solver, with an average temperature difference of 0.246 °C. Table 3.3 represents the minimum and maximum volume average temperatures for each solver setting.
Fig.3.5a represents the comparison of volume averaged velocity in similar to the volume average temperature calculation. Across all the simulations, both solvers follow the same trend, with minor variations in the magnitude. The difference in velocity between coupled and segregated solvers is 6.0×10^{-4} m/s on average. Table 3.4 shows the minimum and maximum volume average velocity values for each solver setting. These findings further suggests that segregated solver is capable of resolving flow fields of internal flows with negligible deviation.

Table 3.3: Comparison of minimum and maximum volume-averaged temperatures between coupled and segregated solvers.

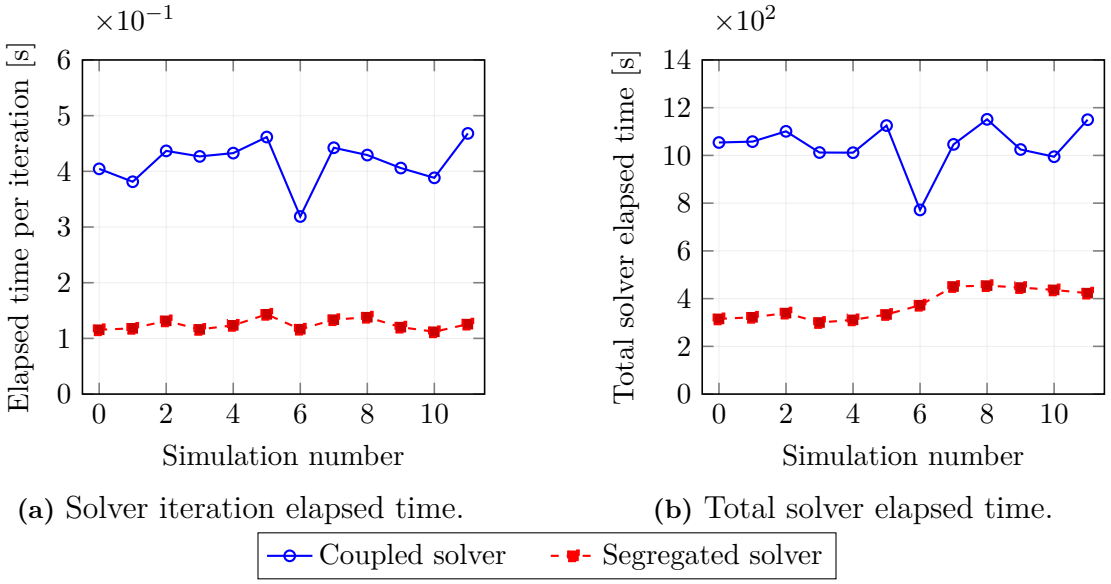
Statistic	Coupled (°C)	Segregated (°C)
min	20.03	20.33
max	23.27	23.38

In addition to solution accuracy, computational speed is measured for both the solvers in terms of total solver elapsed time and elapsed time per iteration. As shown in Fig.3.6a and Fig.3.6b, the segregated solver outperformed in all simulation cases. The segregated solver is on average 68% faster than the coupled solver, highlighting the computational overhead in performing simulations with the coupled solver. This time difference is crucial when numerous CFD simulations are conducted by varying boundary conditions to generate training data for machine learning models.



(a) Volume-averaged velocity. **(b)** Volume-averaged temperature.

Figure 3.5: Coupled vs. segregated solver comparison over 12 simulations: Comparison of temperature and velocity fields.



(a) Solver iteration elapsed time. **(b)** Total solver elapsed time.

Figure 3.6: Coupled vs. segregated solver comparison over 12 simulations. Comparison of solver iteration time and total solver time.

Table 3.4: Comparison of minimum and maximum volume-averaged velocities between coupled and segregated solvers.

Statistic	Coupled (m/s)	Segregated (m/s)
Min	0.0552	0.1008
Max	0.1100	0.1018

3.2.4 Transient Simulation

In addition to steady-state simulations, transient simulations are also performed to study time-dependent evolution of temperature and velocity in the room for fixed boundary conditions. The primary objective is to verify how long the HVAC system must take to reach steady-state condition. This is required to define the control time horizon for the RL-based controller. Without this reference time, the controller may be trained on intervals that are either too short to reflect meaningful changes or too long to overlook dynamic variations.

Table 3.5: Solver and CFL control parameters applied in the transient simulation.

Parameter	Value / Setting
Time Step Size	0.05 s (fixed)
Initial CFL Number	1
Minimum CFL Number	0.1
Maximum CFL Number	500
Target AMG Cycles	4
Max Iterations per Multigrid Level	50
CFL for Multigrid Acceleration	5

The physics and solver settings used for the transient case are considered to be identical to the steady-state configurations. The simulation is conducted in implicit unsteady time-integration scheme with a fixed time step of 0.05s. The key constraint with the time step is to maintain the CFL number below 0.5. A lower CFL number is critical in transient simulations for buoyancy-driven flows, where even small errors can distort the representation of natural convection. By enforcing lower CFL values, the simulation can capture gradual changes in flow and temperature fields, providing a robust understanding of the controller's update. A summary of the transient simulation setup is summarized in Table 3.5

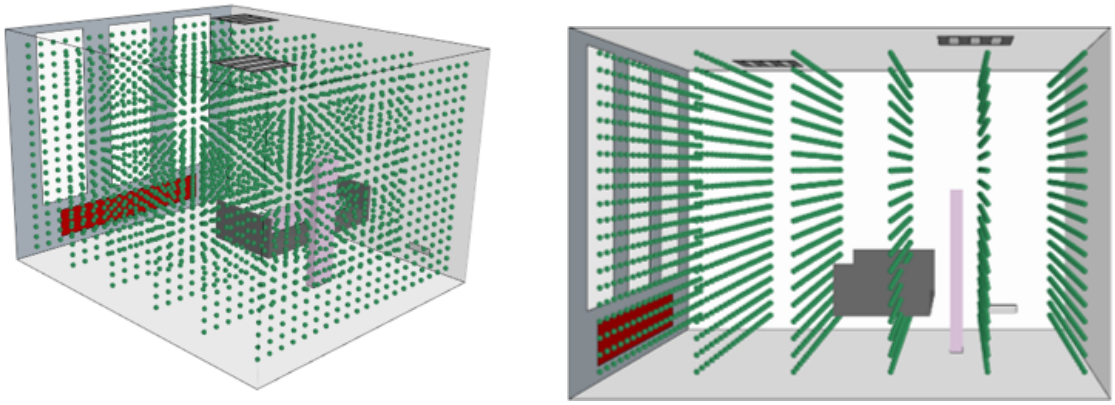
3.3 Data Collection for Machine Learning

To generate a comprehensive dataset to train the machine learning model, the CFD simulation is executed iteratively under different boundary conditions. The boundary condition ranges were chosen to reflect realistic seasonal operation in occupied rooms. A 21°C target setpoint reflects standard human comfort temperature [26]. The supply air temperature intervals cover both cooling and heating operation for a mixing HVAC system. This is also coupled with space heating from low-temperature to legacy high-temperature regime, providing realistic modes of operation [27] [28]. The outdoor temperature ranges are chosen to reflect Swedish climatic conditions. The air change rates cover a range from typical office conditions to high ventilation rates used in clinical control room, thus enabling the analysis across various operating scenarios [29] [30]. The mass-flow rate is calculated using Eq.2.15 from the chosen range of air-change cycle. The variation for each of boundary condition is presented in Table 3.6.

Table 3.6: Boundary condition ranges across summer, autumn, and winter, including common operating parameters.

Variable	Summer	Autumn	Winter
Target temperature (°C)	21	21	21
Initial temperature (°C)	23–27	18–24.5	15–30.5
HVAC inlet temperature (°C)	15–21.5	16–24.5	20–30.5
Heating element temperature (°C)	–	40–70	50–95
Window temperature (°C)	15–32.5	6–14.5	-15–6.5
Common to All Seasons			
Air-change cycle (/h)	6, 10, 15		
Mass-flow Rate (kg/s)	0.1224, 0.204, 0.306		
Mass Flow Rate per Inlet (kg/s)	0.0612, 0.102, 0.153		

In order to automate the process, a Java-based macro is set up to run the simulations in a loop within STAR-CCM+, with boundary conditions are being randomly sampled from the predefined range. To ensure that boundary condition combinations are physically meaningful, the boundary condition space is divided into three categories representing three seasons, summer, autumn, and winter. This seasonal segmentation of boundary conditions ensures contextual relevance to the indoor comfort simulations. Separate iterative loops were configured for each season to systematically cover the entire space of valid boundary condition combinations.



(a) 3D view of point distribution inside the domain

(b) Side view of point distribution inside the domain

Figure 3.7: Point representation within the room domain used to extract data from CFD simulations.

For each simulation case, the flow-field data were extracted using the presentation grid in STAR-CCM+. The distribution of the points is visually represented in Fig.3.7. Specifically, six presentation grids were defined within the domain. The first and last sampling planes were placed 200mm away from the walls. The remaining four planes are placed between these two sampling grids, spaced uniformly to have uniform coverage across the domain. Each plane contains a structured grid of 20 x 20 points, resulting in 2,400 sampling points in the domain. Each of these points extracts the state variables such as pressure,

velocity and temperature along with the spatial location of the point. The output from each simulation is stored in a .csv file.
Simultaneously, the corresponding set of input boundary conditions used in that simulation are stored in a separate CSV file. These two files were given matching identifiers to prevent data mismatch during training the machine-learning model. This loop is executed for all 3 seasons, resulting in 973 unique simulations that collectively form a training dataset for the machine learning model.
The collected CSV files are then processed using Python to combine the boundary condition file (input) and its corresponding flow field data (output) into a compressed binary format, .npz (NumPy Zip) files. This format significantly reduces the data loading and processing times compared to .csv files. These .npz files are later unpacked into .npy arrays, which are highly efficient for batch loading and help with parallel processing of data during training.

3.4 Energy Consumption Calculation

To calculate the energy consumed by the HVAC system, three primary energy sources were considered: the HVAC heating/cooling coil, the HVAC unit fan, and the heating element inside the room. These components contribute to the total thermal and electrical energy usage within each control step, which corresponds to a twenty-minute simulation interval. The total energy consumed by the HVAC system is calculated as:

$$E = E_{coil} + P_{fan} + Q_{rad} \tag{3.1}$$

HVAC heating/cooling coil (E_{coil})**:** The HVAC coil is responsible for heating/cooling the temperature of the supply air before it reaches the room. The energy consumed by the coil to reach the room to the target temperature is calculated based on enthalpy increase of air as it is heated or cooled down to the specified inlet temperature. This is given by the fundamental thermodynamic equation for heat transfer.

$$Q_{\text{coil}} = \dot{m} \cdot C_p \cdot (T_{\text{inlet}} - T_{\text{room}}) \tag{3.2}$$

where,

- Q_{coil} is the instantaneous heating power required by the coil $[W]$,
- $\dot{m}$ is the air mass flow rate $[kg/s]$,
- C_p is the specific heat capacity of air, taken as 1005 J/kg·K,
- T_{inlet} is the target supply air temperature $[°C]$,
- T_{room} is the current mean room air temperature $[°C]$.

Since the electric power consumed by these systems is expressed using the Coefficient of Performance (COP), the COP reflects how much of the thermal energy provided by the system per unit of electric energy consumed. Typical air-to-air heat pumps for building applications exhibit COP values ranging from 2.5 to 4.5, depending on the outdoor temperature and system configuration [21]. In this project, the value of COP is chosen to be 3.5 which provides balanced and realistic energy consumption calculations. The electric power is calculated as:

$$E_{\text{coil}} = \frac{|Q_{\text{coil}}| \cdot \Delta t}{\text{COP}_{\text{coil}} \cdot 3.6 \times 10^6} \tag{3.3}$$

where,

- E_{coil} is the energy consumed by the coil over the time step [kWh], [s],
- Δt is the control interval duration $[s]$,

Fan Power Consumption (P_{fan})**:** The energy consumed by the HVAC fan unit is calculated based on the relationship between static pressure and the inlet airflow rate, as mentioned in the fan performance curve. For this project, the fan curve for Mitsubishi PLFY-P100/125VBM-E series HVAC unit was chosen from its official service manual [22], which provides static-pressure values corresponding to airflow rates for two inlet configurations. Fig.3.8 shows the curve chosen for this project. This provides a realistic non-linear relationship to the fan power and inlet airflow rate.

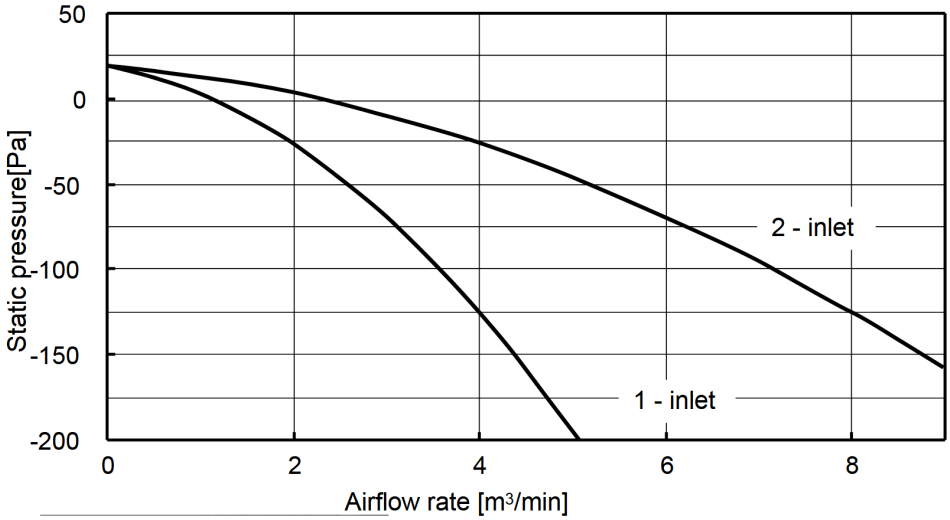


Figure 3.8: Fan Curve HVAC unit [22]

The mechanical power required by the fan is given by:

$$P_{\text{fan}} = \frac{\Delta P \cdot \dot{V}}{\eta} \tag{3.4}$$

where,

- ΔP is the pressure rise across the fan [Pa],
- $\dot{V}$ is the volumetric flow rate [m^3/s],
- η is the fan efficiency (assumed absorbed in the polynomial fit).

The fan curve provides the ΔP as a function of $\dot{V}$ which is then digitized to data points, the volumetric flow rate is converted to mass-flow rate (kg/s), and a suitable cubic polynomial fit is given by Eq.3.5

$$P_{\text{fan}} = | -2.89 \cdot \dot{m}^3 - 13.89 \cdot \dot{m}^2 + 50.0 \cdot \dot{m}| \tag{3.5}$$

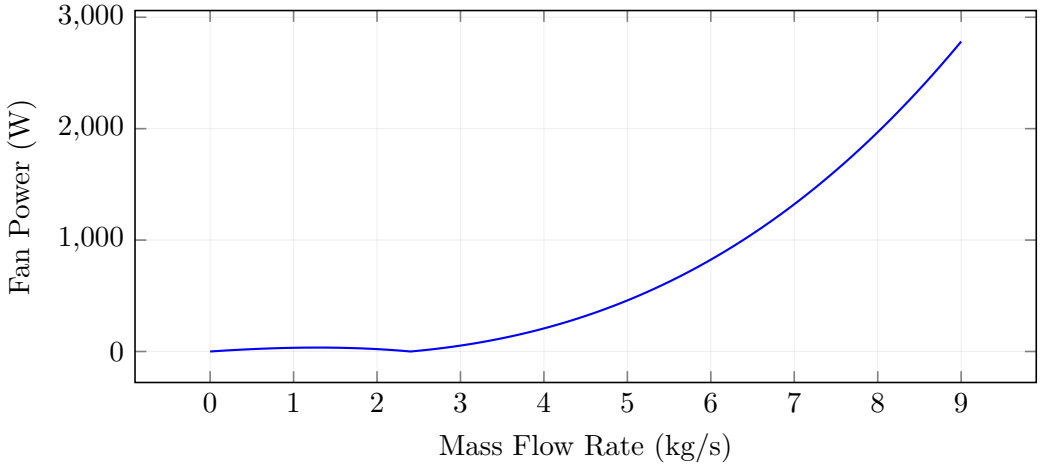


Figure 3.9: Fan power consumption as a function of mass flow rate.

This polynomial captures the non-linear increase in fan power with mass flow rate, which is shown in Fig.3.9

Radiator Heating Energy Consumption (Q_{rad})**:** In practice, the radiator is required only in colder conditions to save energy. Thus, in this implementation, the radiator in the HVAC system is activated when the outdoor temperature falls below 10°C. This condition simplifies the implementation of the control system, as the radiator does not need to operate in warmer weather. The energy consumption of the radiator is modelled using a simplified steady-state heat transfer approach that relates the heat output to the temperature difference between the radiator and room temperature. The heat transfer is modelled as:

$$Q_{\text{rad}} = U A_{\text{rad}} \cdot \max(0, T_{\text{rad}} - T_{\text{room}}) \tag{3.6}$$

where,

- U is the overall heat transfer coefficient $[\text{W m}^{-2}\,\text{K}^{-1}]$,
- $A_{\text{rad}} = 1.04, \text{m}^2$ is the measured surface area of the radiator,
- T_{rad} is the radiator surface or setpoint temperature $[°C]$,

The heat transfer coefficient considers both natural convection and thermal radiation from the radiator surface. The radiator is assumed to deliver heat output of, 1500W at a temperature difference of 40°C between the radiator surface and the ambient room temperature. Using these assumptions in Eq.3.6 and a radiator area of 1.04m^2, the overall heat transfer coefficient, U, is calculated to be 36W m^{-2} K^{-1}. This is simplified to calculate the relative performance comparison between two controllers. However, this heat transfer is not constant, and it is not linear with temperature difference, as it is influenced by the convective and radiative effects under different thermal conditions.

3.5 Surrogate Model

This section covers the methodology adopted for implementing a Fourier Neural Operator (FNO) to predict three-dimensional steady-state indoor flow fields, such as temperature and velocity for the given input boundary conditions. This methodology part is split into

three sections: data loading and processing, setting up the FNO architecture, the training loop, and model evaluation and visualization. All the computations are performed using GPU. Tuning and selection of hyperparameters are performed on an NVIDIA V100 GPU and the final training process is performed in NVIDIA H100 GPU. All of these steps are explained in detail in the following subsections.

3.5.1 Data Loading

The FNO model requires the input and target data to be structured, which enables the model to build a correlation between input and target data across spatial dimensions. The data are loaded from the .npz files, which contain both input and its corresponding flow-field data. The input and output files are extracted from the .npz files and appended to the respective list. The arrays have shape of [N, P, C], where N is the number of samples, P is the number of spatial points in the domain, and C is the number of features. For input, the features are HVAC inlet temperature, inlet mass-flow rate, radiator temperature, outdoor (weather) temperature, initial temperature and three-dimensional spatial coordinates for each point. The output features consist of temperature and velocity magnitude.

3.5.2 Feature Scaling and Normalization

The input features vary drastically in terms of scales. For example, the temperatures is in kelvin, and the inlet mass-flow rate is in kg/s, and velocity in m/s. Thus, it is crucial to normalize the data before training the model. The normalization ensures uniformity in the data distribution across features.

Z-score normalization is used to normalize the data. This subtracts each feature elements with mean and then divides by standard deviation of each feature across the dataset. StandardScaler from scikit-learn library is used in this implementation. Before normalizing, the data are flattened to a two-dimensional array. Once normalized, the data are reshaped to their original structure. These scaler objects, used for input and output scaling, are saved using the pickle module. They are used to normalize new sets of data during inference and to convert model outputs back to physical units for evaluation and visualization.

3.5.3 Tensor Preparation and Dataset Splitting

Once the data were normalized, the input and output arrays are converted to PyTorch tensors and reshaped from a three-dimensional tensor ([N, P, C]) to five-dimensional tensor ([N, X, Y, Z, C]) to reflect the three-dimensional spatial structure of the input tensor and be suitable for the FNO model. The FNO applies FFT across the spatial dimensions and the channels are treated as an independent features, like batch dimension (N). So, the data is then permuted to [N, C, X, Y, Z] as the FNO model expects the channel dimension to be positioned after the batch dimension in the input tensor.

This tensor dataset is then split into training and validation subsets. This helps analyze the generalization capability of the trained model. The total dataset split ratio is set to 85:15 where 85% of the data is for training and 15% for validation, and this split is randomized by using the random split method from PyTorch. The input and output tensors are then a Tensor Dataset object from PyTorch, forming a complete dataset for training with a batch size of 8, enabling data shuffling to improve training convergence.

3.5.4 FNO Architecture

The FNO model implementation is aimed at predicting the temperature and velocity magnitude fields from spatial and boundary condition data. The model architecture is adopted by using neuralop library, with the network components configured for three-dimensional structured data. Different hyperparameters are tested and choice of the hyperparameters mentioned below are further analysed in Section 4.2.1.2.
The model configuration uses 256 hidden channels and 4 spectral layers. Each layer includes a spectral convolution module that operates on the 3D spatial grid with 20 low-frequency Fourier modes along each dimension. This enables the model to efficiently process the spatial data with boundary conditions and long-range spatial correlations.
Once the data are passed through spectral transformation and linear transformation, the output data is then passed through a single MLP layer to enhance the output representation. This MLP performs pointwise transformations to refine the final predictions at each spatial location.

3.5.5 Training

The weights of the FNO model are optimized using Adam optimizer, with a learning rate of 0.0005. MSE is used as the loss function to calculate the loss over each training epoch. The training is carried out for 200 epochs. In each training epoch, the model is set to training mode and a forward pass is performed for each batch of training dataset. After computing the losses, backpropagation is applied to update the model weights using the optimizers. This step ensures that the model progressively improves its predictions by minimizing the loss value. Apart from MSE, MAE and relative error are also calculated for monitoring the performance of the neural operator. Individual MSE losses for temperature and velocity predictions are tracked to check for any imbalance in training. After training, the model is saved in .pth format so that it can be used in the controller to predict the flow field.

3.5.6 Model Evaluation and Visualization

After the FNO model is trained, this model is used to generate predictions of temperature and velocity fields for the given input boundary conditions. This follows a series of steps such as input data generation, normalization, and post-processing. The given input boundary conditions are uniformly distributed over the entire spatial grid by repeating for every spatial point. The spatial grid is the same as in the input files that were used for training to ensure consistency with the training and prediction resolution. The input array is two-dimensional, and it consists of eight columns (three spatial coordinates and five input boundary condition). The input is normalized using the previously saved StandardScaler parameter file, and it is reshaped to a five-dimensional array ([1, 8, X, Y, Z]). This input array is now converted to PyTorch tensor. The trained model is loaded and set to evaluation mode. This step ensures that all the nodes in the architecture are active and contribute to the output. Then a forward pass is performed with the normalized input tensor. The output of the model is now permuted and reshaped back to [X, Y, Z, 2]. This output array contains the temperature and velocity field at the corresponding locations, and it is denormalized using StandardScaler with the same parameter file. These unscaled values are now flattened to a two-dimensional array and saved in a .csv file to be used by the controller. The unscaled values are also saved in .vtk format using the PyVista library for post-processing and visualization.

3.6 Soft Actor-Critic RL for HVAC Control

This section elaborates on the implementation of the Soft Actor Critic (SAC) controller, targeting energy efficiency and indoor comfort.

3.6.1 Actor network

The actor network is a simple feed-forward neural network that is responsible for learning stochastic policy, which maps the current state of the system to a probability distribution over possible control actions. In this implementation, the actor network is constructed using a two-layer neural network with 512 nodes at each layer. Fig.3.10 illustrates the process flow of the actor network. The input to the network is the normalized state vector which consists of current room temperature, outdoor temperature, the room temperature at the previous time step, the temperature difference between current room temperature and target room temperature and finally action vector that contains inlet temperature and inlet mass flow rate of the HVAC unit and radiator surface temperature. This state vector is passed through the neural network and ReLU activation function is utilized to introduce nonlinearity to the output of each node.
The output of the actor network contains two separate linear layers, one of them predicts the mean (μ) and another predicts the standard deviation ($\log \sigma$) of the Gaussian distribution. This distribution is used to define the probability distribution of possible actions. The standard deviation from the neural network is limited to the range -20 to 2. This restriction avoids extremely large or small standard deviations, which can lead to unstable learning and ineffective exploration.

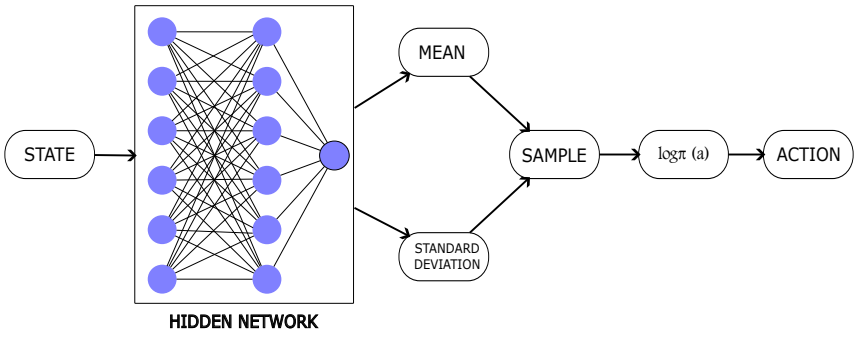


Figure 3.10: Actor Network Workflow

The action vector is then sampled from the Gaussian distribution. However, random sampling is non-differentiable and this restricts the utilization of chain rule during back-propagation of actor neural network. This affects the model's ability to learn. So the action vector is then sampled by using reparametrization. This involves in sampling the action vector from the generated Gaussian distribution and performing a deterministic transformation of the samples using learned mean and standard deviation. This allows the gradients to propagate through the sampling step and, this is essential for training the actor network.
The output mean value from the actor neural network is unbounded, as the reparametrization trick to sample the action vector. To ensure the sampled values are bounded to avoid gradient explosion, the sampled action vector is passed through tanh function, which

forces the sampled action vector to be bounded within $[-1, 1]$. This helps in stabilizing the learning of the actor network.
After sampling the action vector, the log probability of the action term is calculated to update the actor policy, which balances exploration and exploitation using Eq.2.33. The action vector from the actor network is then denormalized using inverse min-max normalization. This conversion provides the action values that are directly applied as the control parameter.
In the implementation by Haarnoja et al. (2018) [23], the entropy temperature term, α is learned dynamically by using a separate optimization step to balance the trade-off between reward maximization and entropy. In the current implementation, this term is kept constant for simplification.

3.6.2 Critic Network

The critic network calculates cumulative reward, which is also known as Q-values, for a given state-action pair. This provides a score on how good or bad the action is for that specific state. In this project, two independent critic networks are formed following[23] to eliminate the issue of overestimation bias which is commonly observed in single Q-network function approximation. Fig.3.11 shows the workflow of the critic network.

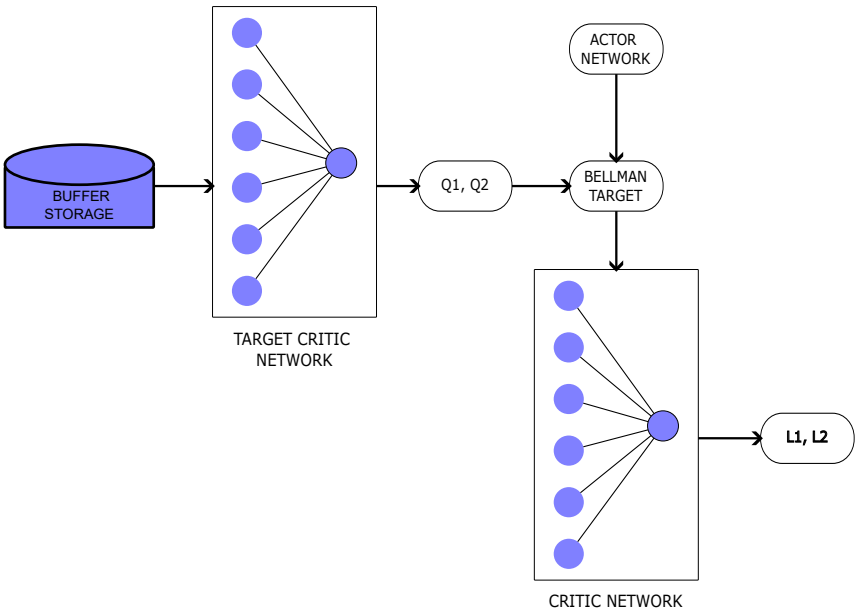


Figure 3.11: Critic Network Workflow

Each critic network receives two inputs, the current state (s_t) and its corresponding action (a_t). The state vector consists of current ambient room temperature, and it's associated boundary conditions, which is flattened in a one-dimensional vector. The action vector consists of control parameters provided by the actor network at that time step. These state vectors are normalized using the min-max scaling function, to be bounded within [-1, 1]. The action vectors are scaled with tanh activation to be bounded within [-1, 1] to match with the actor's output range. These two vectors are concatenated and given as input to a fully connected feed-forward neural network. This critic neural network contains two hidden layers, each with 512 nodes, and each node is activated using ReLU activation function. The output layer returns a scalar Q-value which scores the state-action input.

Mean Squared Error (MSE) loss function is used to calculate the error between predicted Q-values and the target value. The target value is calculated using the Bellman backup equation to train the critic network. For the predicted state values (s_{t+1}) from the flow prediction model, the actor network samples the next action (a_{t+1}) from its current policy and the log probability for the corresponding action $\log \pi(a_{t+1}|s_{t+1})$ is calculated. These values are fed into the target critic networks, which estimate the corresponding Q - values. The Bellman target, y is calculated using the Eq.2.31. The minimum of Q-values from both target function is used to prevent overestimation, and this is penalized with the entropy term. The resulting Bellman target y is calculated using Eq.2.31.
The discount factor, γ, is set to 0.998 and entropy temperature coefficient, α, is fixed to 0.1, which balances between exploration and exploitation. The Q-values from the two critic networks for the current state-action pair are compared with the target value using MSE loss function, which is represented in Eq.2.21. This loss is minimized using individual optimizer function using an Adam optimizer for each critic network with specified initial learning rate of 1×10^{-4}. The target value, y, is detached during backpropagation to avoid the gradient flowing through other variables used to calculate the y. This prevents incorrect update of weights of critic network during backpropagation. Gradient clipping is applied to the critic networks by limiting them to a maximum of 1.0 to avoid gradient explosion during learning.
The target critic networks are updated gradually using Polyak averaging, given in Eq.2.35. Here, the weights of the target critic network are slowly moved towards the current critic network with an update rate of 0.997, which contributes to stable learning.
The training data for the critic network is taken from buffer memory that stores the experiences in tuple format (s_t, a_t, r_t, s_{t+1}). The buffer memory is defined to store 10,000 tuples and in each training step, a batch of 128 randomly selected samples is used to train both critic networks.

3.6.3 Reward and Penalty

The reward function defines and guides the learning behaviour of the SAC by quantifying desirability of a particular control action based on the weighted combination of thermal comfort metrics, and energy consumption. The reward function penalizes excessive energy consumption, control oscillations, and thermal discomfort.
For each time step, the surrogate prediction model updates the room temperature based on the actions given by the controller. The reward function then evaluates the outcome of this transition based on a set of defined scalar penalty terms. The final reward is calculated as

$$r_t = 1.0 - \text{cost} \tag{3.7}$$

The cost is a weighted sum of six penalty components. The first component accounts for total energy consumed by the system.

3.6.3.1 Energy-Based Penalty Component

The energy consumption is calculated as in Eq.3.1. The energy component is multiplied by the time-step duration of 1,200 seconds, and it is then penalized by a weighting factor of 0.595. To simplify the controller, the radiator component is conditionally disabled when the outdoor temperature exceeds 10°C.

The second component penalizes the deviations from the target temperature of 21°C. There is no penalty applied when the temperature deviation is within ±0.5°C. The deviation is then normalized by a scaling factor of 10°C, resulting in the penalty term below:

$$\varepsilon = \frac{|T_{room} - T_{target}|}{10}, \qquad \text{if }, |T_{room} - T_{target}| > 0.5 \tag{3.8}$$

This term is multiplied by a weighting factor of 1.33.
The third penalty term penalizes abrupt changes in the control action. The difference between the current and previous control action is computed and normalized by the difference between the action bounds of the inlet temperature. This normalized delta is penalized with a weighting factor of 0.26.

3.6.3.2 Comfort-Based Penalty Component

The occupied zone is the part of the room where the thermal comfort is critical. This includes several phenomena such as ambient temperature, relative humidity, radiant temperature. For simplification, in this project thermal comfort refers to the room temperature being as close to the target temperature (21°C). The occupied zone for is defined to be in the range of 1.05-1.5 m of room height throughout the room. Fig.3.12b provides a visual representation of the defined zone. Three additional penalties were given based on the temperature distribution in the indoor environment.

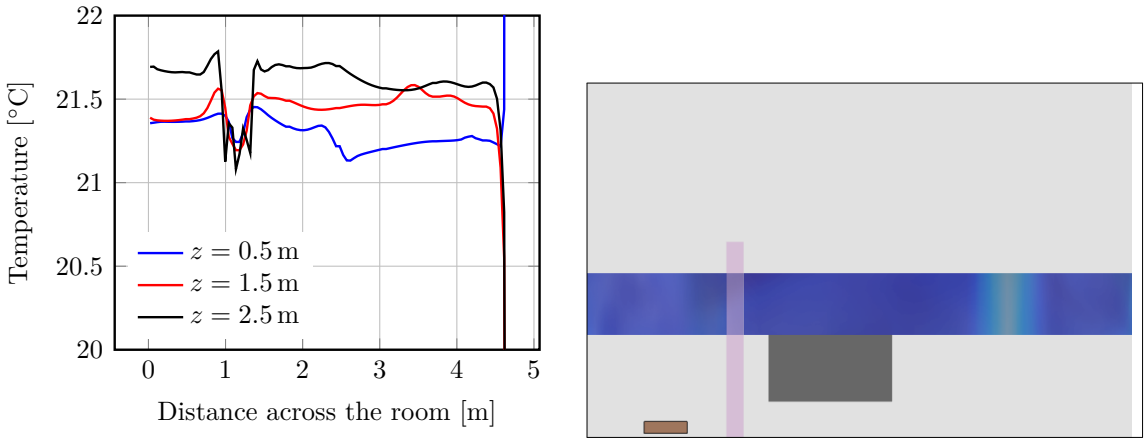
- **Stratification Penalty:** It is calculated as the difference between maximum and minimum temperatures within the occupied zone. Penalties are imposed when the deviations are greater than 1°C. Fig.3.12a illustrates the stratification effect, where the temperature differences are observed within the comfort zone. Stronger stratification in these zones produces a larger penalty, and the controller provides actions to reduce these penalties.
- **Zone Mean Penalty:** This is computed to be the difference between mean occupied-zone temperature and overall room-mean temperature. The penalties are given when the deviations are more than 0.5°C. From Fig.3.12a, the overall room mean temperature lies closer to the mean zonal temperature as the temperature trends across three heights are parallel and closer to each other.
- **Zone-Variance Penalty:** The penalties are applied when the standard deviation of the temperature in the occupied zone exceeds 0.5°C

All the weighted penalties are summed to compute the total cost:

$$\begin{aligned} \text{cost} = {} & W_{\text{TEMP}} \cdot \varepsilon + W_{\text{ENERGY}} \cdot E_n + W_{\text{OSC}} \cdot \Delta\alpha \\ & + W_{\text{COMFORT}} \cdot \text{spread} + W_{\text{ZONE\_DEV}} \cdot \text{zone\_dev} + W_{\text{ZONE\_STD}} \cdot \text{zone\_std} \end{aligned} \tag{3.9}$$

3.6.4 Training Loop

Unlike conventional RL training loops, which rely on episodic training, the SAC implementation adopts a continuous on-the go learning approach, where the control decisions are made every 20 minutes. This specific interval is chosen because the CFD simulations, used for dataset preparation for FNO, are modelled under steady-state conditions and



(a) Temperature along the room at three heights.

(b) Occupied zone (1.05–1.5 m) used for comfort-penalty evaluation.

Figure 3.12: Visual representation of vertical temperature stratification and the occupied-zone band used to compute comfort penalties.

transient data are not available. As a result, the FNO model predicts steady-state temperature fields corresponding to the controller action. Through the results of transient simulation, the room's thermal conditions stabilize within 20 minutes on average. So the controller must operate every 20 minutes. The external weather conditions are updated every hour, the weather conditions are updated once in every fourth time step.
For each step within the training loop, the agent collects the current state of the room (temperature field), selects an action through the action network and applies it. The FNO model then predicts a new room temperature field based on the new action and the updated weather conditions. These results are then used to calculate the reward using the reward function as stated in Eq.3.9.
This transition-current state s_t, chosen action a_t, the computed reward r_t and next state, s_{t+1} is stored in a replay buffer. This buffer memory is stored as a list with maximum capacity of 10,000 samples. A warm-up phase is defined where the update of the actor and critic network is frozen until the buffer memory has at least 128 samples. For each training iteration, a batch of 128 samples is drawn randomly from the buffer. The learnable critic updates first by minimizing the Mean Squared Error between the predicted Q-values and the Bellman target. The target Q values are calculated using the minimum Q-value among two target critic networks and the entropy term. The target critic network is then updated using Polyak averaging with a factor, $\tau = 0.997$. This ensures the stability in training by evolving the target network gradually over time.
The actor network is trained once the critic networks are updated to improve the policy. The policy update targets to minimize the loss function by maximizing the log probability of the action. This makes the actor to maximize the expected Q-value while maintaining sufficient entropy in action distribution. The overall workflow of the implementation of the SAC algorithm is illustrated in Fig.3.13.

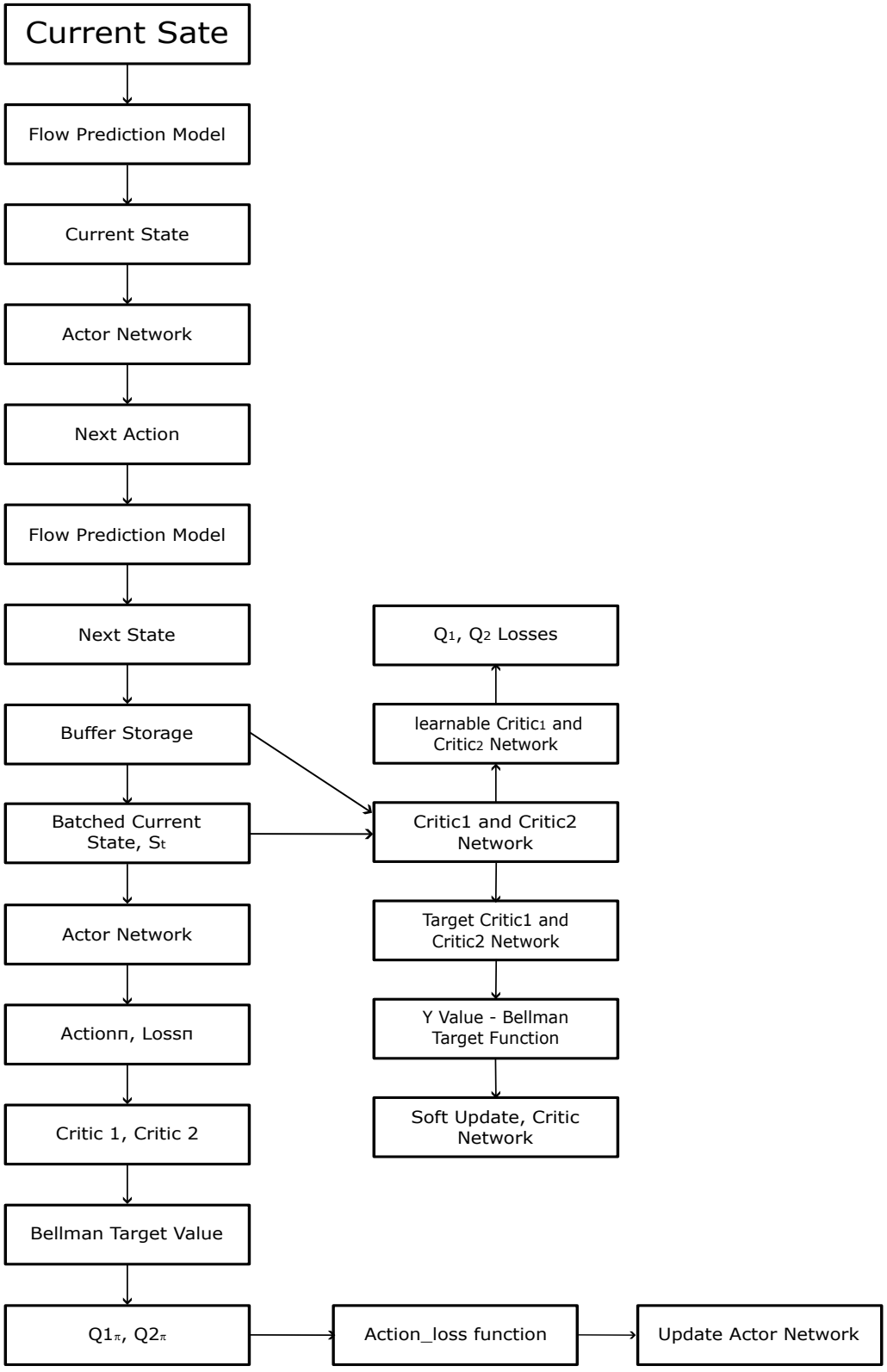


Figure 3.13: Workflow of the Soft Actor-Critic (SAC) controller

3.7 PID Controller

A PID controller is implemented to control and regulate the indoor thermal environment. This controller serves as a reference for the RL controller. This controller controls three action parameters in parallel: the inlet-air temperature, inlet air mass-flow rate, and radiator surface temperature analogous to the RL controller. The derivative part of the PID controller is not considered in this implementation in order to reduce the complexity of the controller. Therefore, a Proportional-Integral (PI) structure is adopted, and three different control actions contribute toward reaching the horizon.
At each control step, the controller calculates the error as the difference between current room temperature and the setpoint (21°C). This error governs three independent PI controllers which work in parallel, controlling each action signal. Each controller calculates the proportional term $K_p e$ and an integral term, which integrates the error over the sampling period. Upper and lower bounds are set to prevent the integral error from becoming excessively large. This eliminates large persistent error of the integral term and makes the controller output stable [24]. Further, the output signals for each control action are bounded by upper and lower limits to limit the action within the data distribution used for modelling, as given in Table 3.7

Table 3.7: Control parameter operating ranges defined for the PID controller.

Parameter	**Range**
Inlet Temperature (°C)	15 – 25
Mass Flow Rate (kg/s)	0.001 – 0.15
Radiator Temperature (°C)	21 – 60

Since the weather data is updated every hour and the controller sampling time is twenty minutes, the same weather data is applied for three consecutive controller steps. At each controller step, the PI controller outputs T_{inlet}, $\dot{m}$ and T_{rad}. These, together with the previous room temperature and the current outdoor temperature, are passed into the FNO model. This model returns the steady-state temperature field and the controller calculates the volume averaged room temperature and this is the response for the controller input for that time step. The controller gains are tuned iteratively to obtain the stable and non-oscillatory values. the gain values are mentioned in Table 3.8. The calculation of energy consumption is performed in the same way as for the reinforcement learning controller for comparable results.

Table 3.8: Proportional (K_p) and integral (K_i) gains assigned to each control parameter loop in the PID controller.

Parameter	K_p	K_i
Inlet temperature	0.15	0.0019
Mass-flow rate	0.0018	0.006
Radiator temperature	0.25	0.0015

4
Results

4.1 Computational Fluid Dynamics

This section presents the results obtained from the CFD simulation. A mesh independence study is performed to ensure the balance between numerical accuracy and computational expense. Thereafter, the flow field and temperature distribution from the steady-state simulation are analysed for a defined boundary condition. This helps in setting up the reward parameters in the reinforcement learning controller. Finally, the results from the transient simulations are evaluated, providing insights into the dynamic behaviour of the indoor flow field, which is used to establish the reference time required for determining the actuation time interval of the controller.

4.1.1 Mesh Independence Study

This study involves systematically varying the base cell size, prism layer count, and local refinement near critical regions such as windows, heaters, and the wall that contains both, where the thermal gradients are expected to be strongest. The base mesh resolution is varied between coarse (0.04m) and fine grids (0.02m), while the prism layer count is adjusted from 4 to 8 while having the same near-wall thickness and total boundary layer height. A consistent surface and volume growth rate of 1.2 is maintained across all mesh configurations to ensure smooth cell transitions and reduce numerical instability. The base cell sizes are based on these variations, ranging from 1.6 million to 4.7 million cells across different mesh cases. Steady-state simulations are conducted for each mesh configuration, and the mesh size is evaluated based on two key parameters

Fig.4.1 shows the variation of average steady-state indoor temperature for different element counts and solver time. The average temperature stabilizes after 3.5 million cells, within a narrow band around 20.5°C. This indicates that further refinement of the mesh does not produce any considerable change in the simulation result. However, the solver time shows a significant increase as the cell count increases. For example, for the coarsest mesh, the solver took 15 minutes for the solution to converge, but for higher cell counts greater than 3.5 million, it required over 45 minutes to converge. This signifies the trade-off between computation time and solution accuracy, especially the meshes with more than 3.5 million elements showed no improvement in volume-averaged temperatures but had a higher computational cost.

The initially chosen cell size was 2 million. To enhance the mesh to capture gradients in the domain better, Adaptive Mesh Refinement was introduced. Therefore, the cell count was reduced from 2 to 1.8 million to adopt the mesh refinement cell count within 2 million and after simulating with AMR, the final count was 2.05 million cells.

The selected mesh for the final simulation contains 1.8 million cells, which was found to provide an optimal balance between accuracy and efficiency. Further mesh settings are

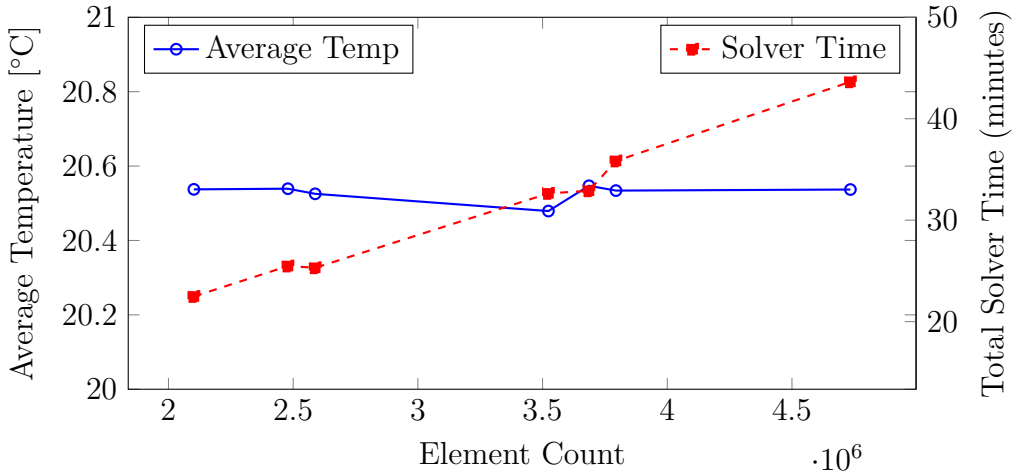


Figure 4.1: Mesh independence study showing variation of average temperature (left axis) and total solver time (right axis).

illustrated in Table 3.1. The visual representation of the mesh is observed in Fig.3.3 and Fig.3.4

4.1.2 Steady-State Simulation

This section focuses on the results from one operating condition, where the boundary conditions are mentioned in Table 4.1. The details about the solver settings and physics chosen are explained briefly in Section 3.2. The solution is analysed on a plane defined 1 m away from the wall, normal to the X-direction, such that it captures the inlet jet from the inlet vent as well as the convection of the air near the radiator and windows.

Table 4.1: Boundary conditions applied in the steady-state CFD simulation.

Parameter	Value
Radiator surface temperature, T_{radiator}	60°C
Supply air temperature, T_{inlet}	21°C
Air mass flow rate, $\dot{m}_{\text{inlet}}$	0.0612 kg s^{-1}
Initial internal air temperature, T_{initial}	15°C
Window surface temperature, T_{window}	12°C

The flow can be determined whether it is buoyant or turbulent, dominated by the calculation of the Richardson number. The ceiling-to-floor height is chosen as the characteristic length, and the temperature difference between the inlet jet and the bulk room air temperature is considered. The simulation yields the Richardson number as 0.14. This lies just above 0.1, which is the threshold that separates the buoyancy-driven flow and inertia-dominated flow [31]. This signifies that the flow is shear dominated in the regions near the jet, while buoyancy forces also contribute to the motion of the air. The colder jet loses its momentum downstream as the significance of the buoyancy forces becomes stronger. In the region near the radiator, the buoyancy forces are dominated as the radiator surface generates a thin hot boundary layer which rises to the ceiling as a thermal plume. At the same time, the colder buoyant stream from the window just above the radiator has a colder boundary layer that creates a shear interface with the hotter buoyant stream from

the radiator surface. This interface redirects the hotter plume into the room. This thermal plume then captures part of the cold fluid on the floor, causing one large recirculation.

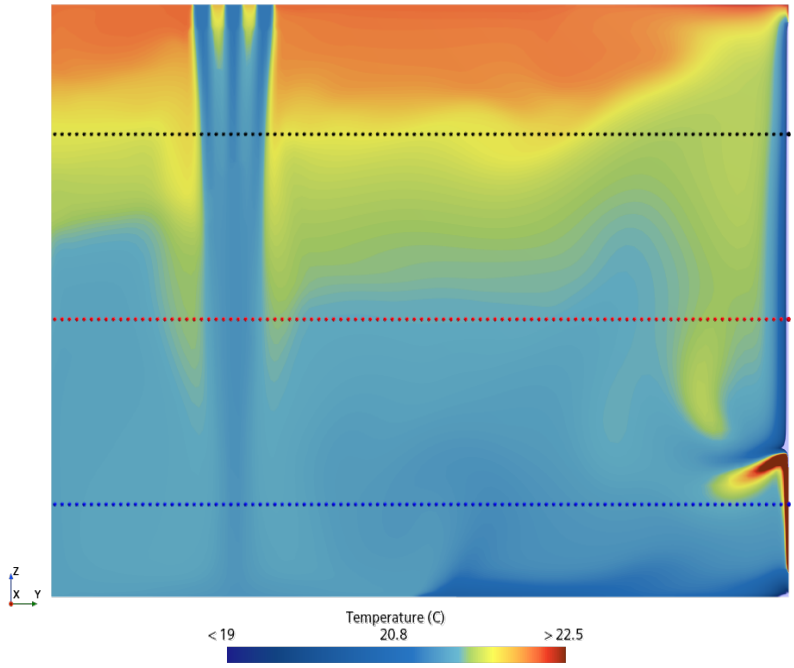


Figure 4.2: Temperature contour of the computational domain showing spatial distribution of the temperature field

This interaction also gives necessary reasons for the three-layer stratification zone observed in Fig.3.12a. In the lowest zone (Z $\leq$ 0.5 m), the temperature lies a few tenths below the average room temperature because of denser air from the inlet jet, and in the intermediate layer (0.5 $\leq$ Z $\leq$ 1.5) is closer to the average room temperature and has the presence of stronger entrainment of the inlet jet and counteracting buoyant separation. The region above this acts like a thermal reservoir, showing that the hotter plumes accumulate beneath the ceiling and spread along the ceiling surface. This spatial pattern is used to define the comfort penalties in the RL controller. The region between Z = 1.1m and Z = 1.5m is designated to be the occupied zone in the RL controller. At every control step, three statistical properties are calculated in this zone, such as the difference between the hottest and coldest points, the difference between the average room temperature and the local mean temperature in the occupant zone, and the standard deviation in this zone. The penalties for these metrics are defined with a dead zone where there is no penalty applied as long as the metric remains within the acceptable range. The limits for each variable are set to 0.5°C for the temperature spread, 0.2°C for the mean deviation, and 0.2°C for the standard deviation. The corresponding penalties are applied once the values increase beyond this limit.
Fig.4.3 illustrates the convergence of the volume average temperature for the room over iterations. The volume-averaged temperature oscillates within ±0.02°C, which signifies that the room has reached the thermal equilibrium with average temperature of 21.52°C. This indicates that the realistic thermal equilibrium of the room may not coincide with the nominal target temperature. This is used to define the comfort criterion in the reinforcement learning controller, which is the difference between the current room temperature and the setpoint. A tolerance of ±0.5°C is introduced to the calculation of the comfort

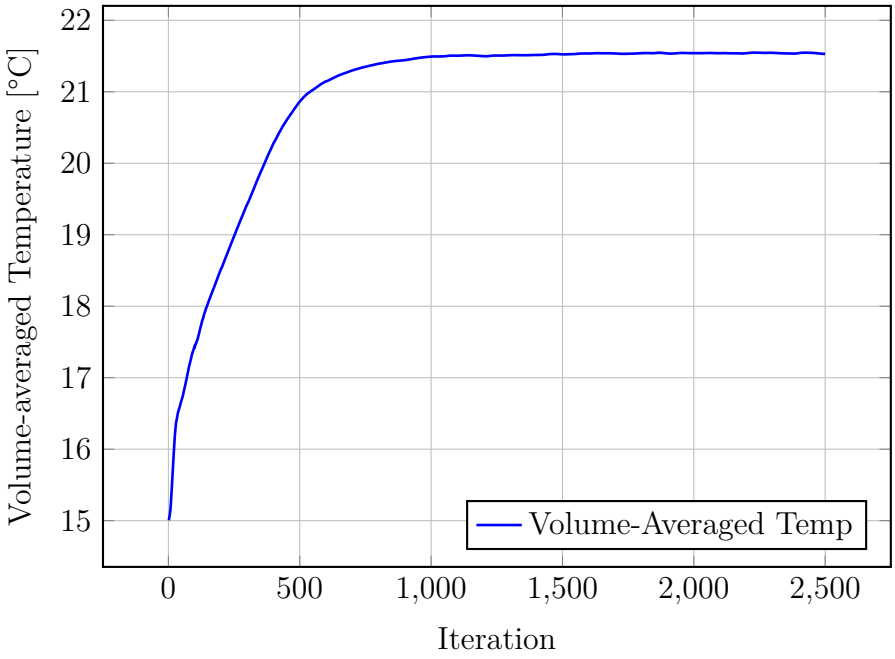


Figure 4.3: Convergence of volume-averaged temperature vs iteration count.

criterion reward component to prevent penalizing for insignificant changes. This ensures that the RL controller does not overcompensate for these insignificant deviations, and provides stable control actions.

4.1.3 Transient Simulation

This subsection provides insights into the results from transient simulation tests to determine the time taken to reach steady-state conditions for four different cases. The boundary conditions for the cases are mentioned in Table 4.2. For each case, the volume-averaged temperature is monitored over time and the time taken to reach the steady-state condition is recorded. Fig.4.4 shows the steady-state, volume-averaged temperature and the time taken to reach the steady-state condition.

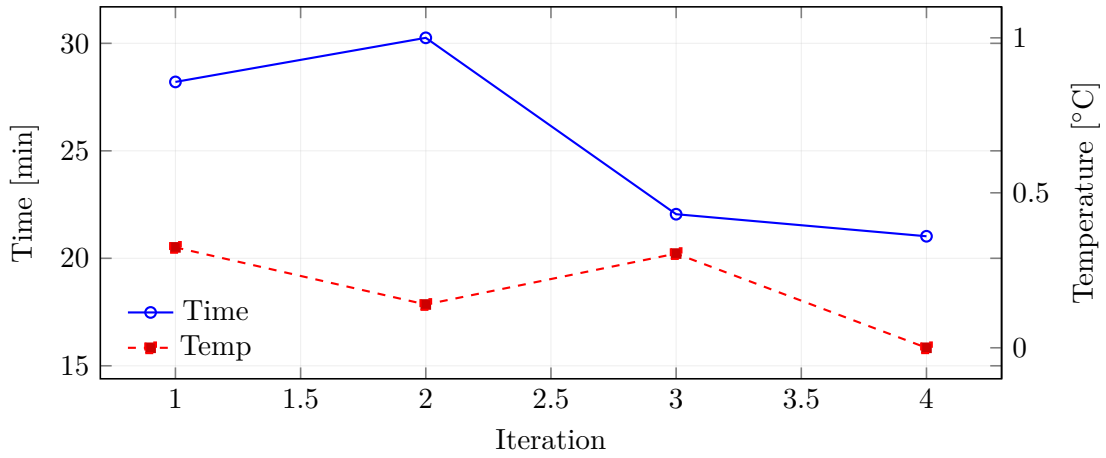


Figure 4.4: Convergence of time and volume-averaged temperature vs. iteration.

For different cases, the time taken to reach steady-state shows modest variations. The

cases with higher mass flow rates converge faster than lower mass-flow rates under the same boundary conditions, yet their differences are small. By contrast, the steady-state averaged temperature varies significantly, reflecting different heat gain and heat losses set by the simulation boundary conditions. This implies that the time taken to reach a state is governed by air exchange time, being proportional to room volume and inversely proportional to the inlet mass flow rate. This curve justifies setting the controller time step to 20 minutes for both PID and RL controllers, since the temperature difference between one action to the subsequent action is smaller, thereby closer to the new equilibrium.

Table 4.2: Boundary-condition cases defined for inlet air, heating element, and outdoor temperatures.

Parameter	**Case 1**	**Case 2**	**Case 3**	**Case 4**
Inlet temperature, T_{inlet} (°C)	21	15	21	15
Initial room temperature, T_{initial} (°C)	15	27	15	27
Heating element temperature, T_{heater} (°C)	90	21	90	21
Inlet mass-flow rate, $\dot{m}_{\text{inlet}}$ (kg/s)	0.0612	0.0612	0.306	0.306
Outdoor temperature, T_{out} (°C)	−15	32	−15	32

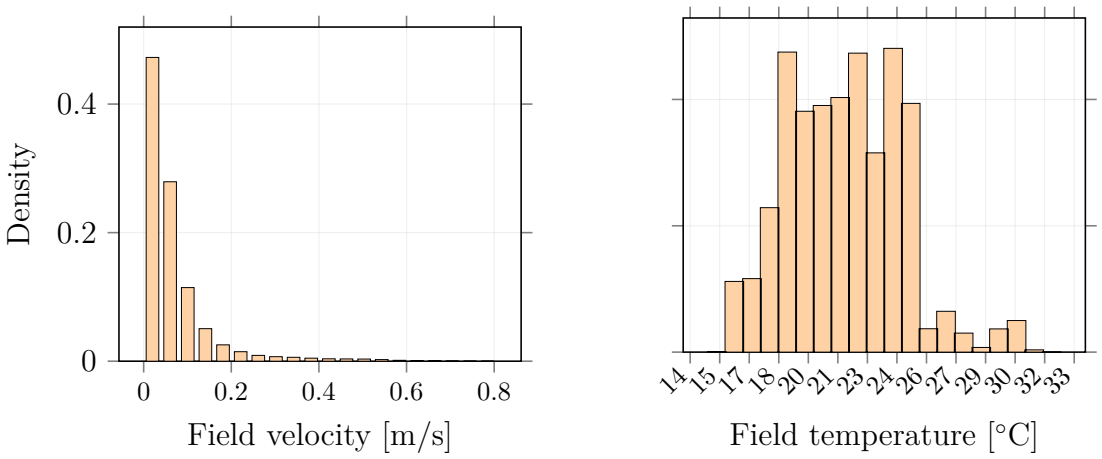
4.2 Machine Learning

In this section, the results obtained from the machine learning model are discussed. The performance of the Fourier Neural Operator (FNO) is evaluated across different metrics and also compared with CFD data to demonstrate the ability to reproduce temperature and velocity distributions under varying boundary conditions. Thereafter, the performance of the reinforcement learning controller based on Soft-Actor-Critic (SAC) algorithm is analysed and benchmarked against a PID controller. This comparison highlights the ability of the SAC controller to balance the energy consumption and thermal comfort effectively. These analyses represent both the capabilities and shortcomings of the data-driven approach for HVAC control optimization.

4.2.1 FNO Model

This subsection presents the results of the Fourier Neural Operator (FNO) trained on the CFD dataset. The analysis begins with evaluating the distribution of the dataset to indicate the range and distribution of the data that the model has been trained on. This is followed by the evaluation of the effect of hyperparameter choices and finally the predictive performance of the model compared with the ground truth data. This helps to understand the accuracy with which the model can reproduce the physical behaviour of the indoor environment under varying boundary conditions.

4.2.1.1 Dataset Distribution



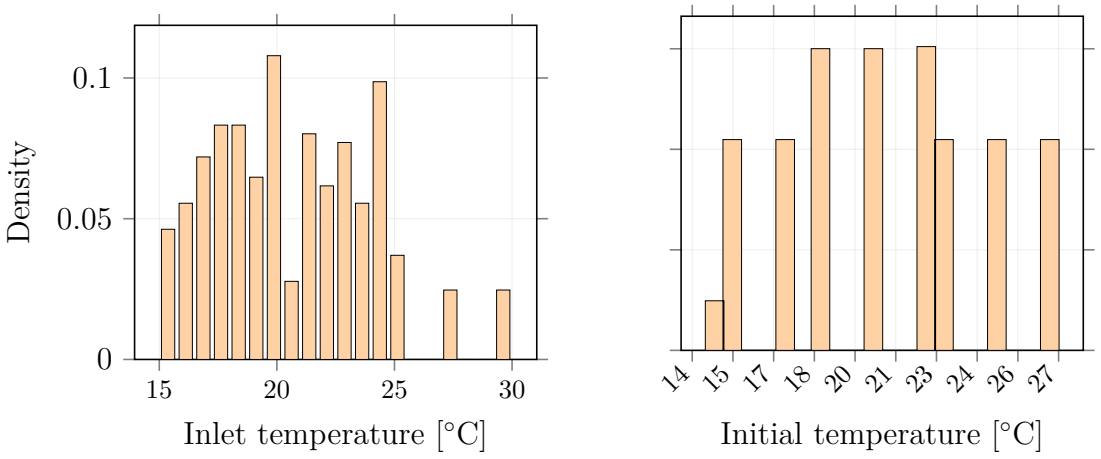
(a) Distribution of field velocity.

(b) Distribution of field temperature.

Figure 4.5: Distributions of field velocity and field temperature in the dataset.

The statistical distribution of the field variables and boundary conditions used for training the Fourier Neural Operator (FNO) is represented in Fig.4.5a to 4.7b. Fig.4.5a illustrates the distribution of the velocity field, and it can be noted that most values are below 0.2 m/s, indicating that the airflow in the room is generally weak. A small proportion is at a higher velocity due to the presence of localized jets by the inlet vents. Fig.4.5b represents the temperature field distribution in the room. The temperature field distribution spans 14.4°C-32.3°C, while the spread of data is concentrated more between 18°C-28°C, reflecting

the typical indoor conditions while still including adverse scenarios, providing a wide range of data. This ensures that the dataset is not biased towards a narrow operating band.

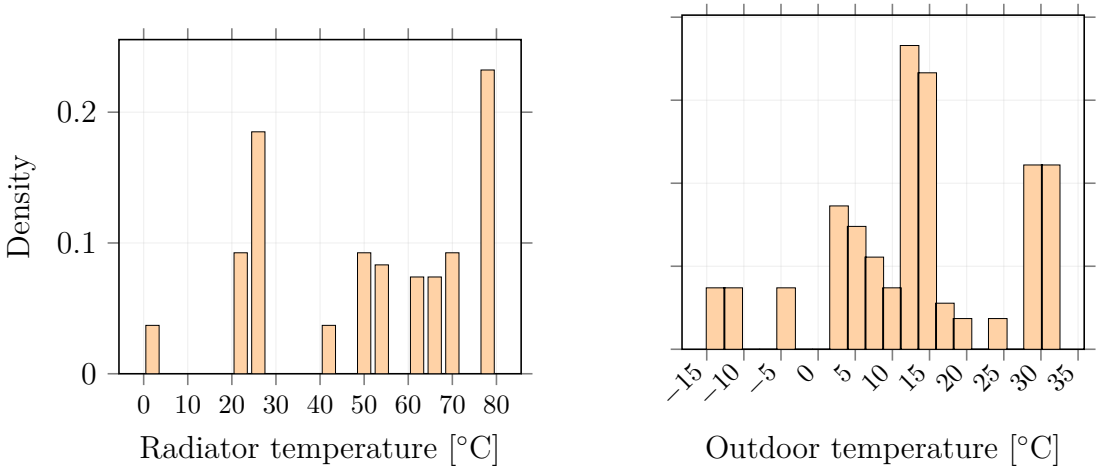


(a) Distribution of inlet temperature.

(b) Distribution of initial temperature.

Figure 4.6: Distributions of inlet temperature and initial temperature in the dataset.

The distribution of the boundary conditions highlights the diversity of the dataset. Fig.4.6a represents the distribution of inlet air temperature varying from 15°C to 30°C, while the initial room temperature, which is represented in Fig.4.6b, captures both cool-start and warm-start conditions. The distribution of the radiator surface temperature is illustrated in Fig.4.7a, providing three distinct common operational setpoints: low (50°C), medium (65°C), and high (80°C) heating modes. Fig.4.7b represents the distribution of outdoor temperatures ranging from -15°C to 35°C, reflecting both extreme winter and summer conditions.



(a) Distribution of radiator temperature.

(b) Distribution of outdoor temperature.

Figure 4.7: Distributions of radiator temperature and outdoor temperature in the dataset.

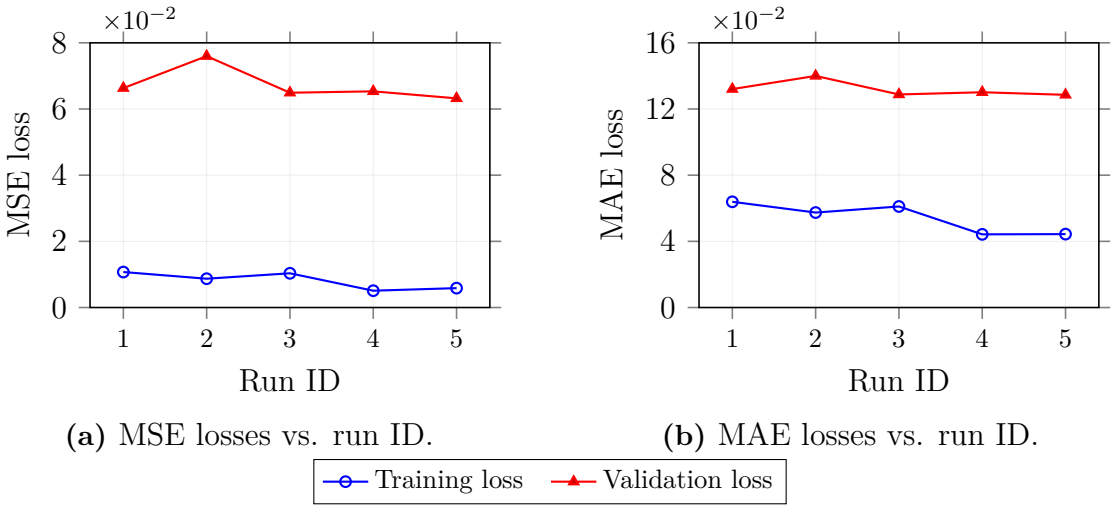
The dataset provides a good representation of both typical and extreme operating conditions. This enables the surrogate model to generalize across different HVAC operation scenarios. This is necessary for the dynamic controller to have a range of operating conditions to explore.

4.2.1.2 Hyperparameter Selection

Suitable hyperparameters are chosen from testing several configurations and evaluated based on their training losses and the time taken to train each model. The primary objective is to choose a set of hyperparameters which provide better generalization and are capable of capturing spatial correlations in the flow fields efficiently, while maintaining the computational cost to a minimum. Table 4.3 shows the choice of hyperparameters for each run.

Table 4.3: Hyperparameter configurations of the FNO model across different runs.

Run ID	Batch size	MLP nodes	No. of layers	Modes			In channel	Out channel
				1	2	3		
1	8	128	4	16	16	9	8	2
2	8	256	4	20	6	11	8	2
3	8	256	4	20	18	11	8	2
4	8	256	4	20	20	11	8	2
5	4	256	4	20	20	11	8	2



(a) MSE losses vs. run ID. **(b)** MAE losses vs. run ID.

Figure 4.8: Comparison of training and validation losses for MSE and MAE across five model runs.

Fig .4.8a and Fig.4.8b represent the losses in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE), for each run. It is observed that the validation losses are higher than the training losses, as a result of overfitting. The magnitude of the training losses remain closer to each configuration, whereas a characteristic variation across validation losses can be observed. Despite increasing the number of MLP nodes, the validation loss is observed to be higher in run 2 than in run 1 because the number of Fourier modes in second dimension was decreased from 16 to 6. This lowers the spectral

resolution and therefore leads to underfitting and higher validation losses. A reduction in training losses between run 3 and run 4, with an increased number of Fourier layers, yet no significant change in validation losses both in terms of MSE and MAE.

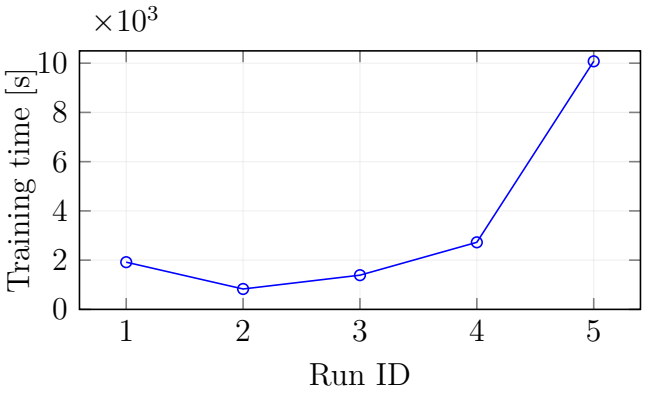
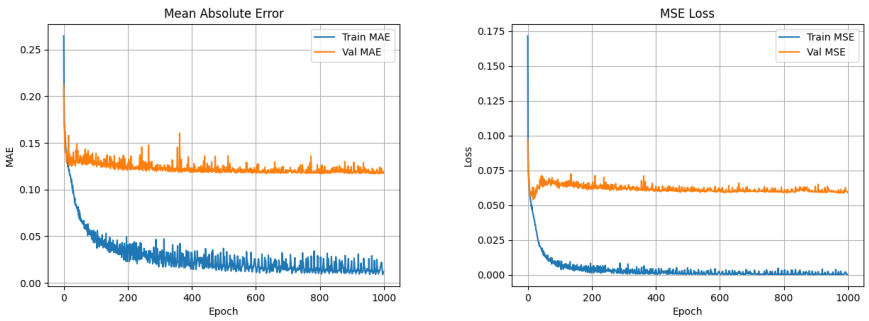


Figure 4.9: Variation of training time with run ID, illustrating differences in computational cost across runs.

Fig.4.9 shows the computation time for each configuration. Computation time increases as the MLP nodes and Fourier modes are increased. By contrast, the training time increased from run 4 to run 5, mainly by reducing the batch size from 8 to 4, yet no significant improvement in the performance of the model. This validates that run 4 shows a good balance between the validation accuracy and the training time. Therefore, this configuration is used to train the final FNO model, which is then coupled with the SAC controller. The performance of this model is discussed further in the next section.

4.2.1.3 Model Performance

The performance of the FNO model is evaluated by comparing the model's predictions with the CFD data of the same boundary conditions. The model's training performance is evaluated using loss curves. The spatial fidelity of the prediction model is analysed by comparing the temperature and velocity contours of ground-truth data and predicted data. The overall prediction performance is then analysed using histograms of temperature and velocity error in the spatial domain.



(a) Mean Absolute Error **(b)** MSE Loss

Figure 4.10: Convergence of training and validation losses for the FNO model, shown using MAE and MSE loss functions.

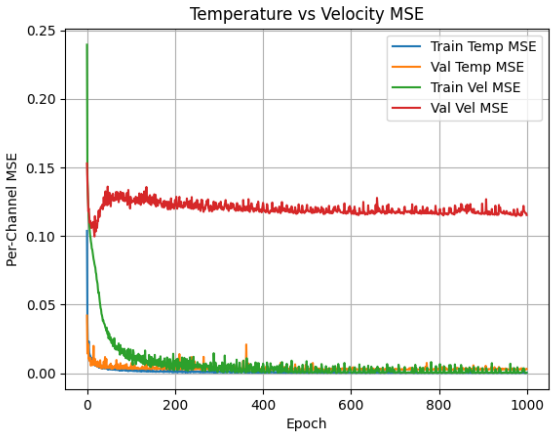


Figure 4.11: Per-channel MSE loss comparison between temperature and velocity.

Fig.4.11 and Fig.4.10 represent the training performance of the FNO model. They are evaluated using MSE loss, MAE loss, and per-channel MSE loss for temperature and velocity. The MSE losses penalize large deviations more from the ground truth, making the gradients steeper during backpropagation. This helps the model provide larger corrective actions, making the training faster. Meanwhile, the MAE captures the deviation, providing the prediction performance of the model. In Fig.4.10a and Fig.4.10b, it can be noticed that the validation losses is noticeably higher than the training losses in both MAE and MSE, indicating overfitting of the model. Fig.4.11 represent the MSE loss curve per output channel. The training loss and validation loss overlap each other for the temperature. This indicates that this model generalizes well for the temperature field predictions. However, the validation loss for the velocity converges at a higher value and can cause deviations in the predictions under unseen boundary condition. This signifies the need for further exploration of hyperparameters, such as increasing the Fourier modes or increasing the number of data points within the domain, providing more information to effectively capture higher-frequency variations.

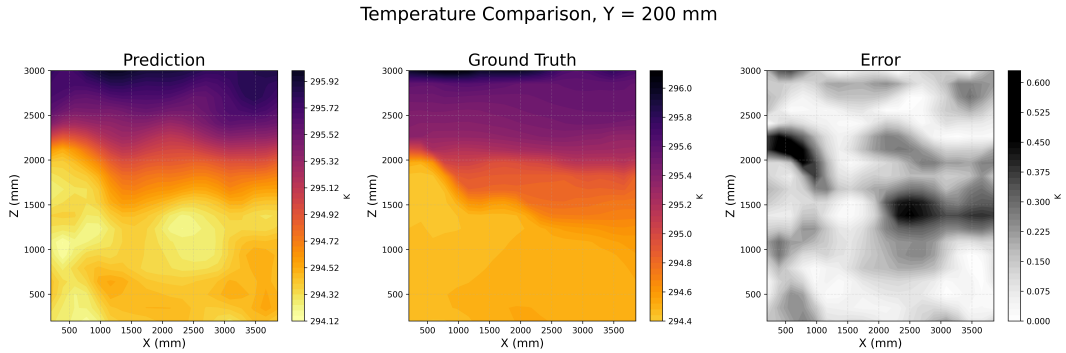


Figure 4.12: Temperature field comparison at $Y = 200$ mm between prediction, ground truth, and error-map.

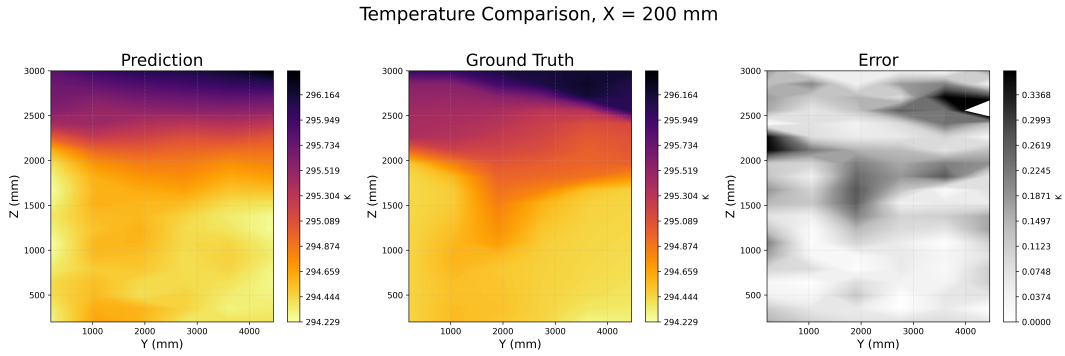


Figure 4.13: Temperature field comparison at $X = 200$ mm between prediction, ground truth, and error-map.

Fig.4.12 and Fig.4.13 represent the comparison of prediction and ground truth for the temperature field and their respective error map shown in greyscale contour. The prediction model captures the primary thermal gradients and stratification patterns effectively, with the magnitude of error lying within 0.6 °C. However, some discrepancies can be found at Z = 2000 where there is a stronger thermal gradient. The uniformity of the error distribution signifies that the model captures the flow field without apparent spatial bias.

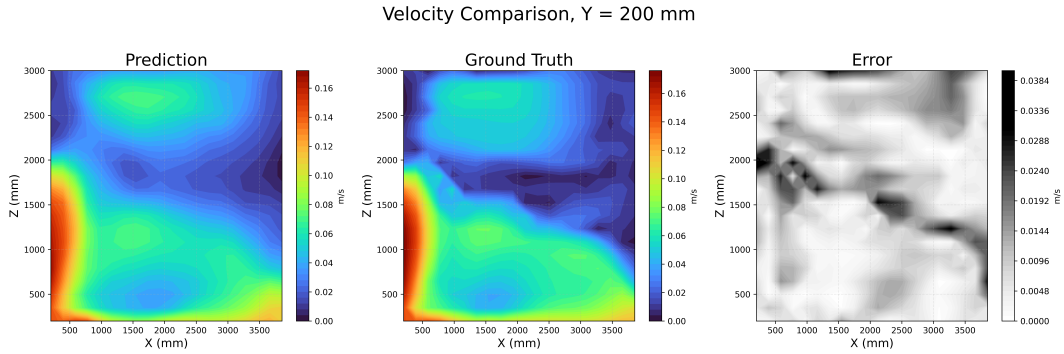


Figure 4.14: Velocity field comparison at $Y = 200$ mm between prediction, ground truth, and error-map.

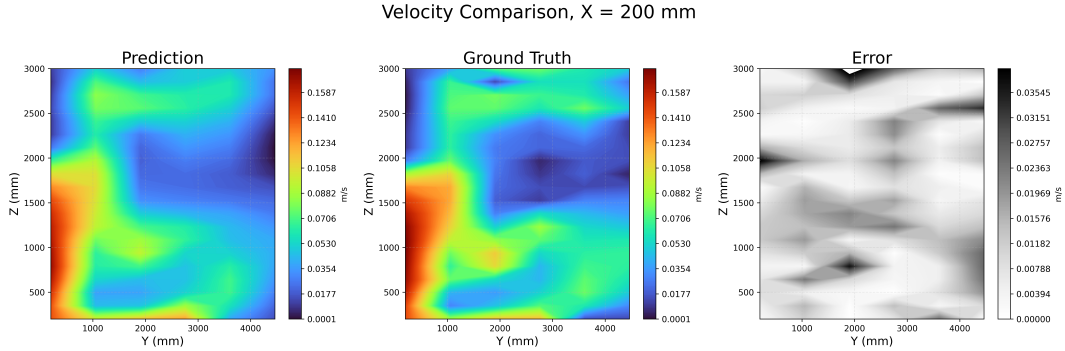


Figure 4.15: Velocity field comparison at $X = 200$ mm between prediction, ground truth, and error-map.

Fig.4.14 and Fig.4.15 show the comparison of the velocity field in the domain between prediction and ground truth. The prediction model produces major flow field characteristics such as the large recirculation pattern and convective flows due to the buoyancy of the fluid. However, the model under-resolves the smaller flow structures in regions of low velocity recirculation. This can be observed in the error-field map, where prominent deviations are present in the zones corresponding due to buoyancy-driven mixing regions.

The comparison of the contour plot shows that the FNO model produces good resolution on temperature fields, reproduces fairly accurate stratification with minimal error. The velocity fields also capture essential flow patterns, but under-resolve the smaller length scales.

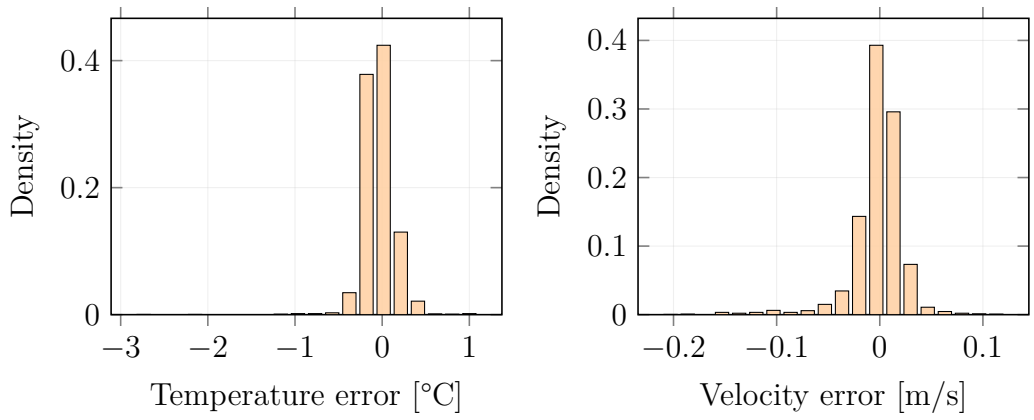


Figure 4.16: Error histograms for prediction versus ground truth. (a) Temperature-error distribution. (b) Velocity-error distribution.

The performance of the FNO model is further quantified by visualizing the distribution of error values in a histogram, which is shown in Fig.4.16. The temperature error distribution shows that the model reproduces the temperature field with minimal deviations, lying within the narrow range of ±0.5°C. In contrast, the velocity error distribution is wider, ranging from ±0.1m/s. The error values are largely centered towards zero, which indicates that there is no bias in the model. However, larger deviations are observed in a few locations, reflecting the model's struggles to produce smaller length-scale features. Despite this, most of the errors are relatively smaller than the absolute predicted magnitude of temperature and velocity. This concludes that the FNO model captures the overall flow field adequately.

4.2.2 SAC Controller

The performance of the SAC controller is evaluated with respect to controlling the indoor temperature while balancing energy cost. The evaluation begins with choosing reward function weights and hyperparameters for the controller by performing multiple runs with randomized hyperparameters and reward weights. The best performing configurations are set, and the long-term behaviour of this model is analysed by simulating for one year of weather data. These results are then compared against the PID controller to quantify the performance of the SAC controller.

4.2.2.1 Hyperparameter Selection

The output of the controller for reinforcement learning is significantly influenced by the choice of weights for each component of the reward function. These weights govern the trade-off between energy consumption and achieving necessary thermal comfort. Therefore, to analyse the sensitivity of the reward weights, the controller is set to run multiple times with a unique combination of randomized weight values using the same weather data over five days. The reward weights are randomly assigned within the interval [0, 1]. This ensures broader exploration of the combination of weights without any bias.

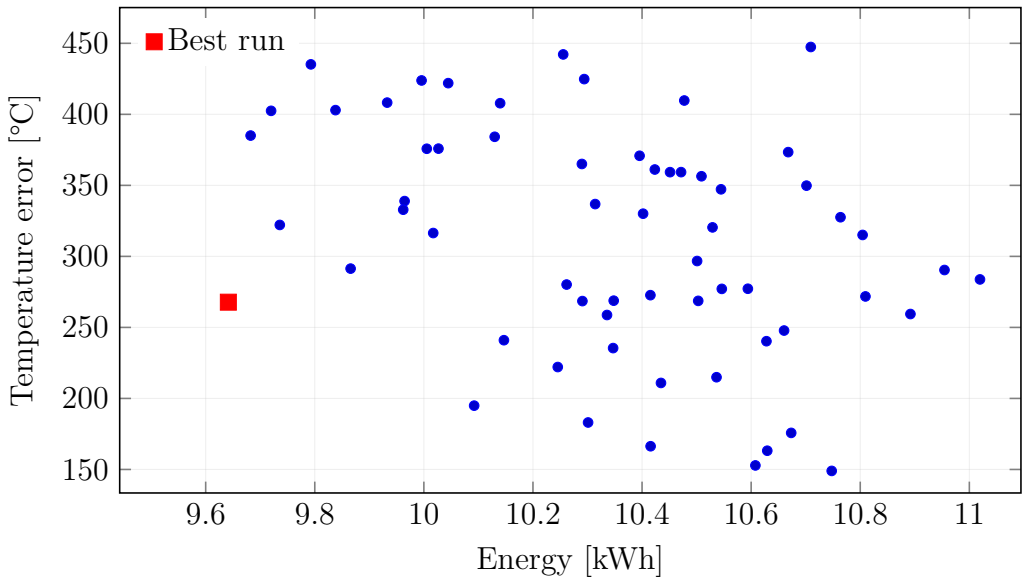


Figure 4.17: Energy–temperature-error trade-off across runs, showing the best-performing case in red.

Fig.4.17 illustrates the cumulative energy consumption against cumulative temperature error, which is the difference between setpoint temperature and volume average room temperature at that control step. Each point corresponds to one set of randomly assigned reward weights evaluated over five days of weather data. This plot highlights the sensitivity of the controller to the choice of reward weights. The controller increases the energy consumption when the weights are placed more on reducing the comfort error. Similarly, the controller reduces the energy consumption at the expense of thermal comfort by placing more emphasis with the weights on energy consumption. In other cases, this weighting method led to unstable control, leading to results where both energy consumption and comfort error are relatively high. The chosen best-performing weight combination achieved the lowest energy consumption while maintaining narrow deviations in thermal comfort. This analysis highlights that different strategies can be adopted in reinforcement learning to balance indoor climate comfort and energy efficiency. Table 4.4 lists the chosen weight values.

Table 4.4: Reward weight values assigned to different terms in the reinforcement learning objective.

Weight Term	Value
Temperature error, W_{TEMP}	0.4297
Energy, W_{ENERGY}	0.8974
Oscillation, W_{OSC}	0.0068
Comfort penalty, W_{COMFORT}	0.6475
Zone deviation, $W_{\text{ZONE\_DEV}}$	0.4265
Zone std. deviation, $W_{\text{ZONE\_STD}}$	0.8342

After identifying the reward weights, the hyperparameters of the SAC algorithm are optimized by performing additional sweeps like before. Parameters such as discount factor (γ), polyak averaging coefficient, entropy (α), and learning rates for both actor and critic networks. Table 4.5 presents the chosen hyperparameters, and Table 4.6 presents the performance for the respective hyperparameters.

Table 4.5: Combination of hyperparameters used for multiple runs of SAC.

Run	Discount γ	Polyak τ	LR Actor	LR Critic	Entropy α
1	0.9918	0.9020	2.21×10^{-4}	2.45×10^{-4}	0.0775
2	0.9814	0.9447	4.00×10^{-5}	2.59×10^{-5}	0.1296
3	0.9641	0.9017	2.25×10^{-4}	7.45×10^{-5}	0.1831
4	0.9754	0.9170	8.05×10^{-5}	4.95×10^{-5}	0.0512
5	0.9783	0.9019	6.89×10^{-5}	2.95×10^{-4}	0.0363

In contrast, all the runs achieved similar reward values, yet notable differences were observed in the oscillations of the control step. These oscillations are quantified by calculating the standard deviation of the reward over time. It is observed that Run 5 produced the most unstable results and a higher critic-1 loss by two orders of magnitude, even though mean reward values are among the highest. This behaviour makes the combination unsuitable for practical use. Runs 1-3 provided better control actions with fewer oscillations, but produced greater actor loss and lower mean reward values compared to Run 4.

Table 4.6: Performance metrics from SAC hyperparameter sweep.

Run	Mean Reward	Reward Std.	Critic-1 Loss	Critic-2 Loss	Actor Loss
1	0.9100	0.0790	0.0071	0.0055	-0.0449
2	0.9263	0.0656	0.0020	0.0020	-0.2688
3	0.9409	0.0776	0.0058	0.0064	-0.3437
4	0.9297	0.0631	0.0060	0.0039	-0.0835
5	0.9279	0.0717	0.3814	0.1165	0.0325

Run 4 was identified as the most favourable combination. This has achieved a comparable mean reward value, with the standard deviation being the lowest, indicating stable control

output. Furthermore, both critic and actor losses are among the lowest of all combinations, making this combination the ideal choice. Fig.4.18 shows the training performance for the chosen hyperparameters.

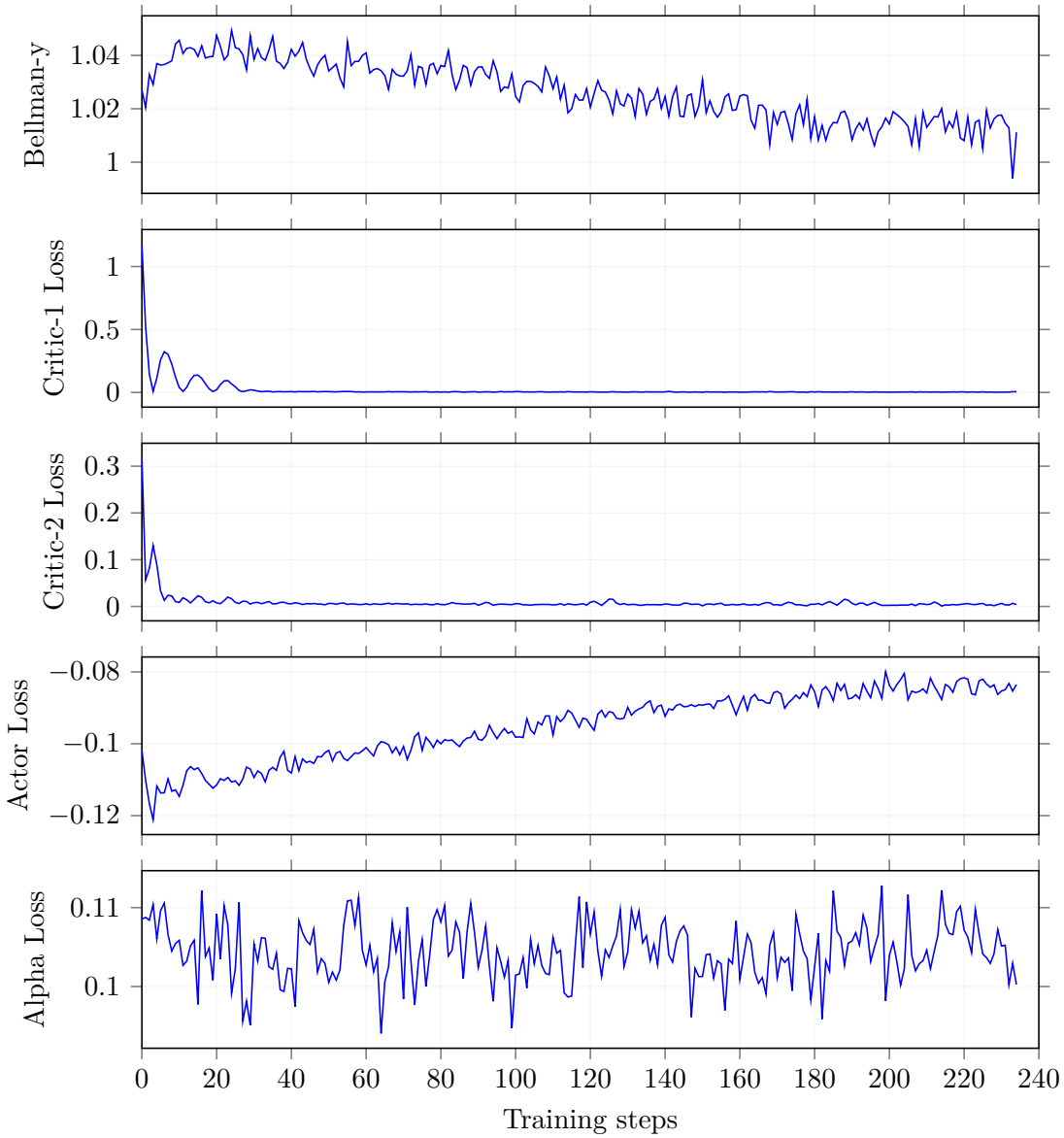


Figure 4.18: SAC training performance for the selected hyperparameters (Run-4).

4.2.2.2 Training Performance

The selected hyperparameters and reward weights are used to run the controller on one year of dynamic weather data to evaluate the long-term performance. Fig.4.19 illustrates the performance of the critic networks, actor, and the entropy coefficient (α) losses over the entire process.

Initial fluctuations are observed in the critic losses in the earlier control steps, and it reaches convergence around 10,000 steps. This indicates that the value function is optimized as the controller has improved its decision-making.

The actor loss curve indicates the adaptation of the controller to the varying operating

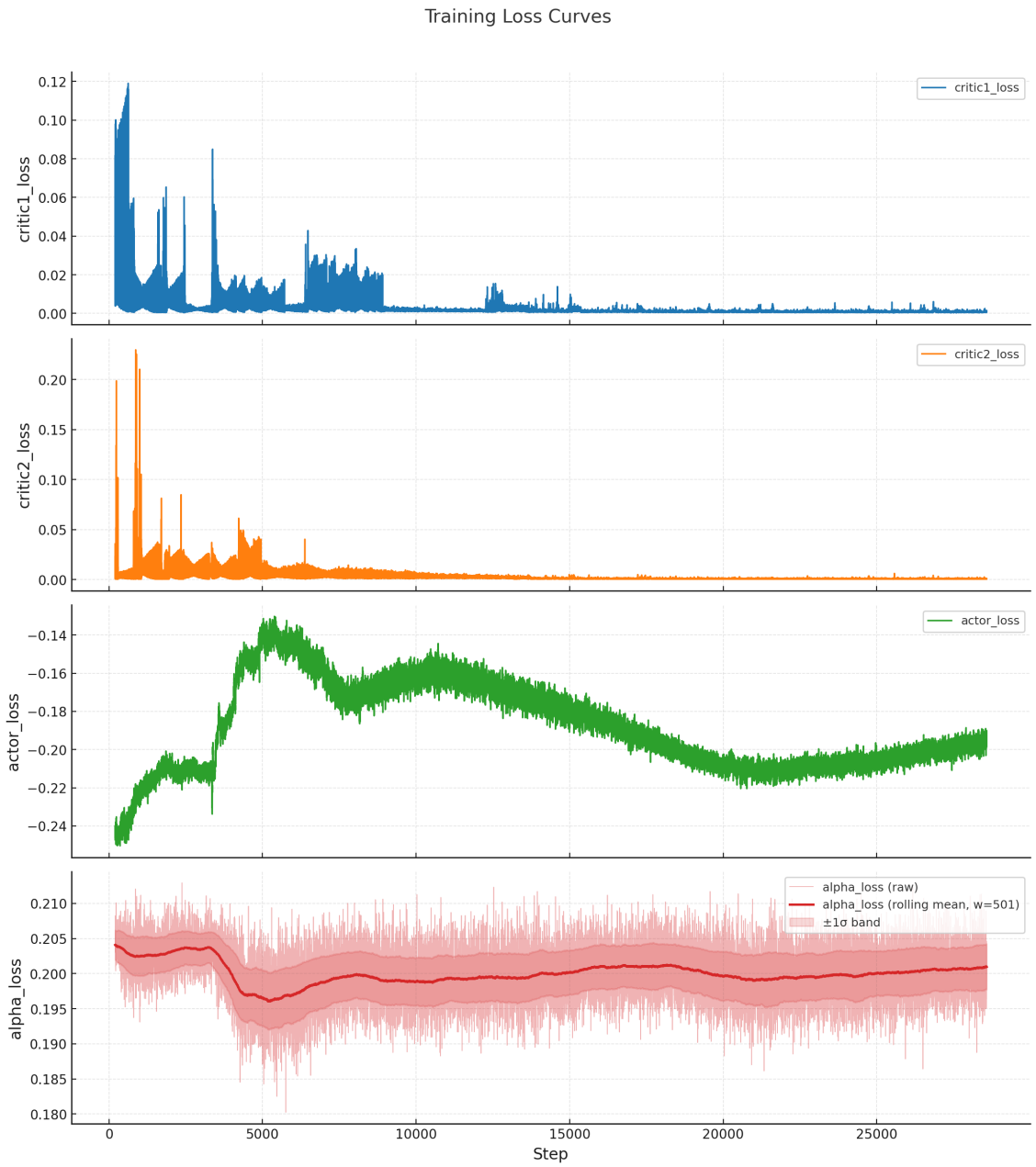


Figure 4.19: Training loss curves for the SAC controller with the chosen hyperparameter.

conditions. In the initial phase, a rapid change in actions can be observed as the controller explores more than exploiting in previously learned strategies. Over time, it provides stable control actions with fewer oscillations as the model has more previously learned control strategies to exploit rather than to explore. The observed oscillations occur over time as the distribution of the selected actions evolves, which modifies the log-probabilities used in the entropy calculation. This makes the actor network modify its control actions to varying weather conditions and adjust the inlet temperature, airflow rate, and heating element temperature to achieve better reward values.
Although the entropy coefficient is defined as a constant and not optimized through the neural network, bounded oscillations are observed. The plotted values do not represent the true 'α-loss', but rather than a scaled entropy term from the actor. The use of a stochastic policy $-\alpha \cdot \mathbb{E}\left[\log \pi(a \mid s)\right]$, Tanh-squashing introduces variability when actions approach their bounds, which leads to sharp variations in the overall entropy term.
The loss curves demonstrate that the chosen SAC configuration can adapt to ever-changing dynamics while being able to provide stable control actions under dynamically varying weather conditions

4.2.2.3 Controller Performance

This section presents the control actions provided by the SAC controller over one year of dynamic weather conditions to evaluate the ability of the controller to balance thermal comfort and energy consumption. Fig.4.20 shows the room temperature for the respective control actions for a dynamically varying weather condition and its corresponding energy consumption over a 12-hour rolling average.
The room temperature lies close to the set point temperature (21°C) despite dynamic seasonal variations in outdoor conditions. The controller consistently maintained the room temperature within the comfort band. Noticeable deviations are observed, particularly during the rapid transition in spring and autumn. The inlet temperature values follow a similar pattern to the room temperature, adapting to the varying weather conditions, providing smoother control inputs that avoid excessive oscillations. It can also be noted that the inlet mass-flow rate increases during warmer conditions and reduced during colder outdoor environments, indicating the actor network producing meaningful control actions in response to varying thermal loads.
It can be observed that the radiator is enabled when the outdoor temperature falls below 10 °C. The controller increases the radiator temperature during the winter months to supply the necessary heat to achieve the target temperature. During warmer outdoor conditions, the radiator is disabled for most of the time, thereby reducing the energy consumption whenever possible.

4.2.2.4 Reward Components

The contributions of each reward component are presented in Fig.4.21. The energy consumption penalty term shows noticeable influence from seasonal pattern, with a higher penalty during late autumn and winter, when the heating demand is highest. The oscillation penalty remains close to zero for most of the year, with a few spikes can be observed; these are due to steeper control actions from the actor network in response to the rapid fluctuations in the weather data. These spikes decrease in subsequent control steps, indicating that the controller is effectively learning from past actions and providing smoother control actions.

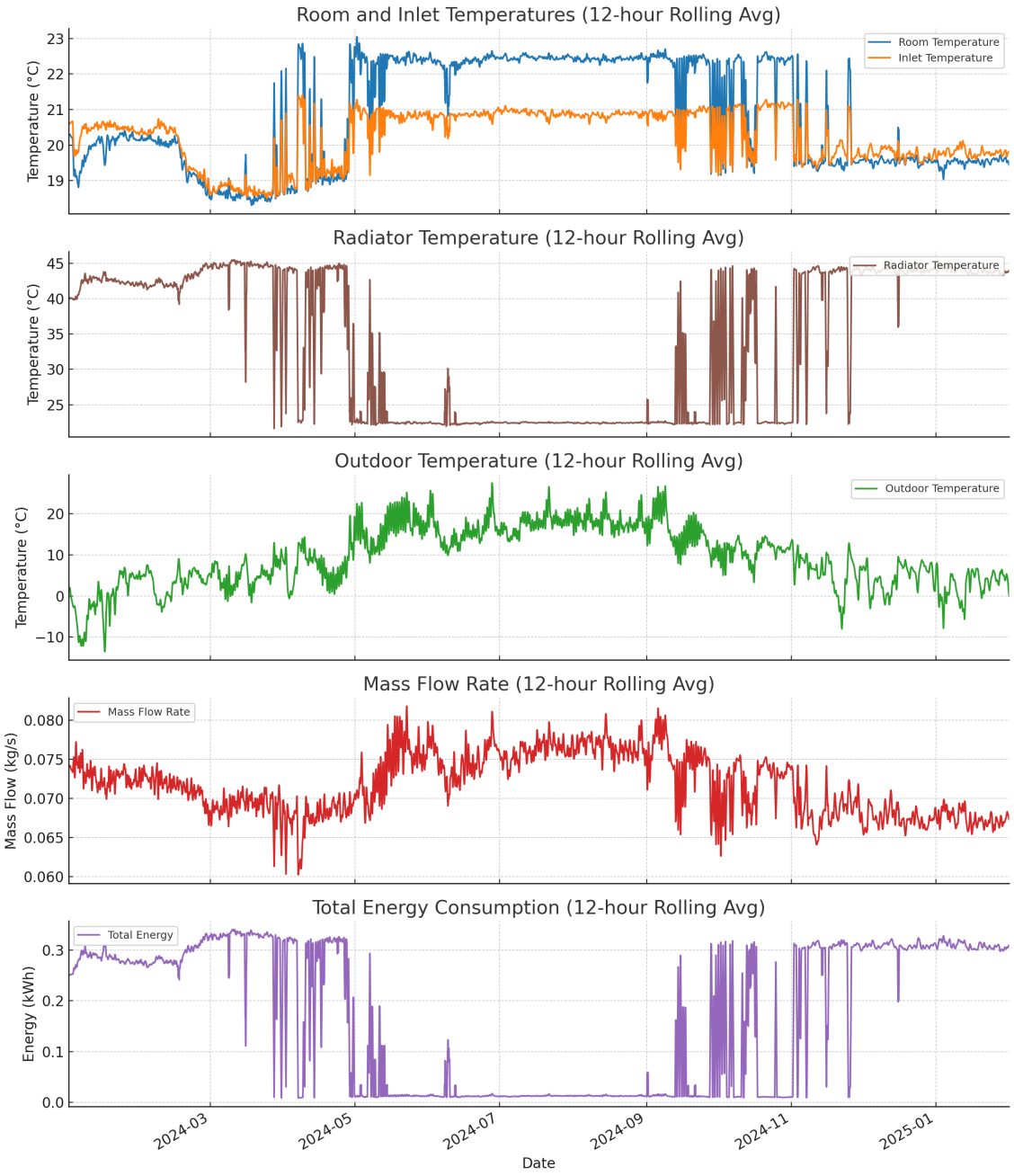


Figure 4.20: Room/inlet temperature, outdoor temperature, mass flow rate, and total energy consumption (12-hour rolling averages).

The temperature penalty term (ε), which represents the deviation of the volume-average room temperature from the target, shows significant fluctuations in the initial control steps and then shows stable performance within the defined comfort zone (± 0.5°C). The performance of the model can be further improved by reducing the comfort-zone tolerance and increasing the entropy to enable the model to explore better control actions.
The overall reward curve stabilized to a constant reward value, indicating the controller maintained a favourable balance between energy consumption, thermal comfort, and control smoothness. Additional reward terms, such as zonal stratification and zonal comfort variations, are also included in the reward function, but they provided negligible results.
The SAC controller produced stable control outputs, maintaining the room temperature while adjusting the energy consumption based on the seasonal demands. The following section presents the PID controller implementation which serves as the benchmark for the comparison of the SAC controller.

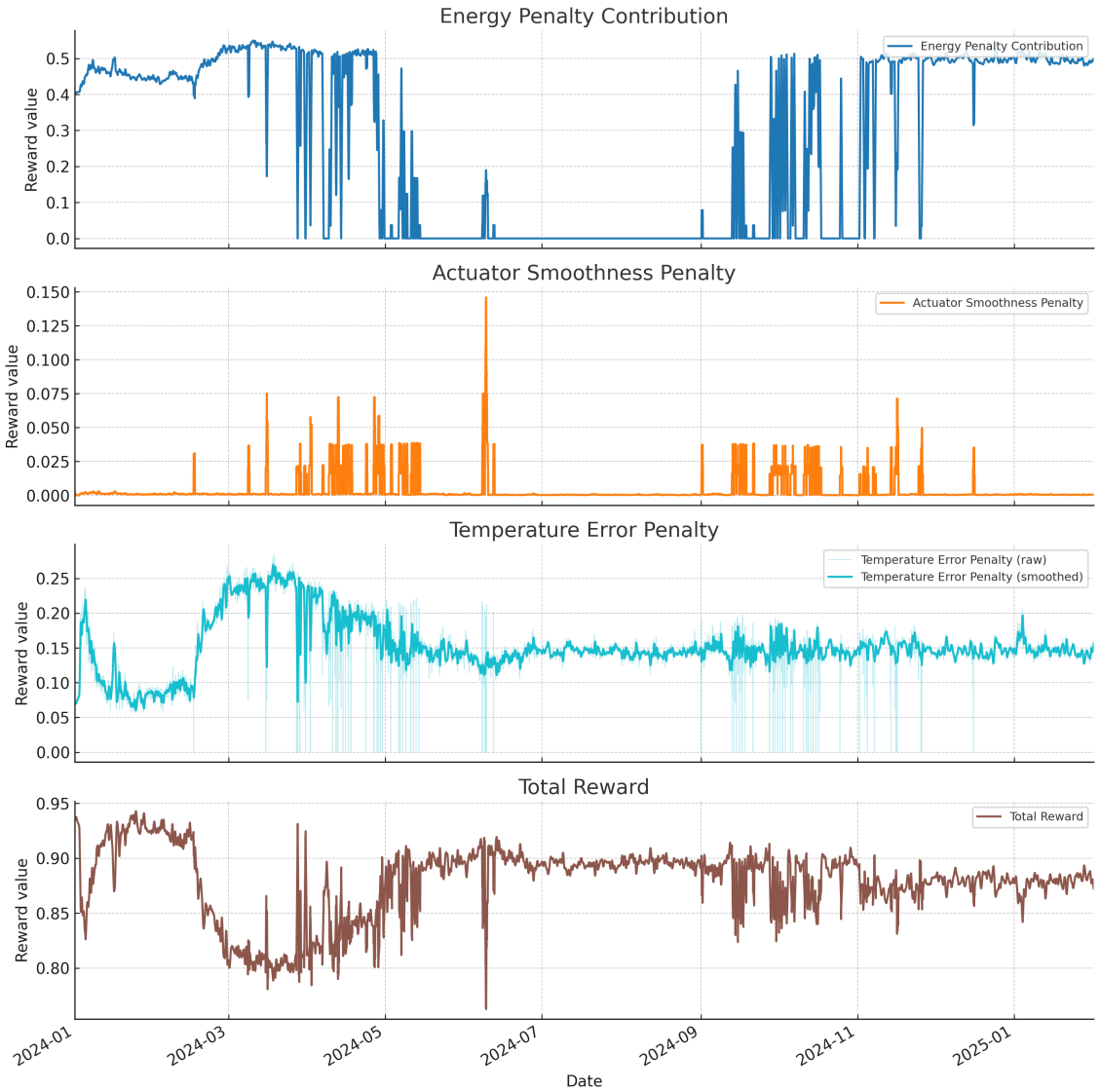


Figure 4.21: Year-long controller outputs (12-hour rolling averages): normalized energy E_n, actuator duty d_{act}, exploration rate ε, and total reward.

4.2.3 PID Controller

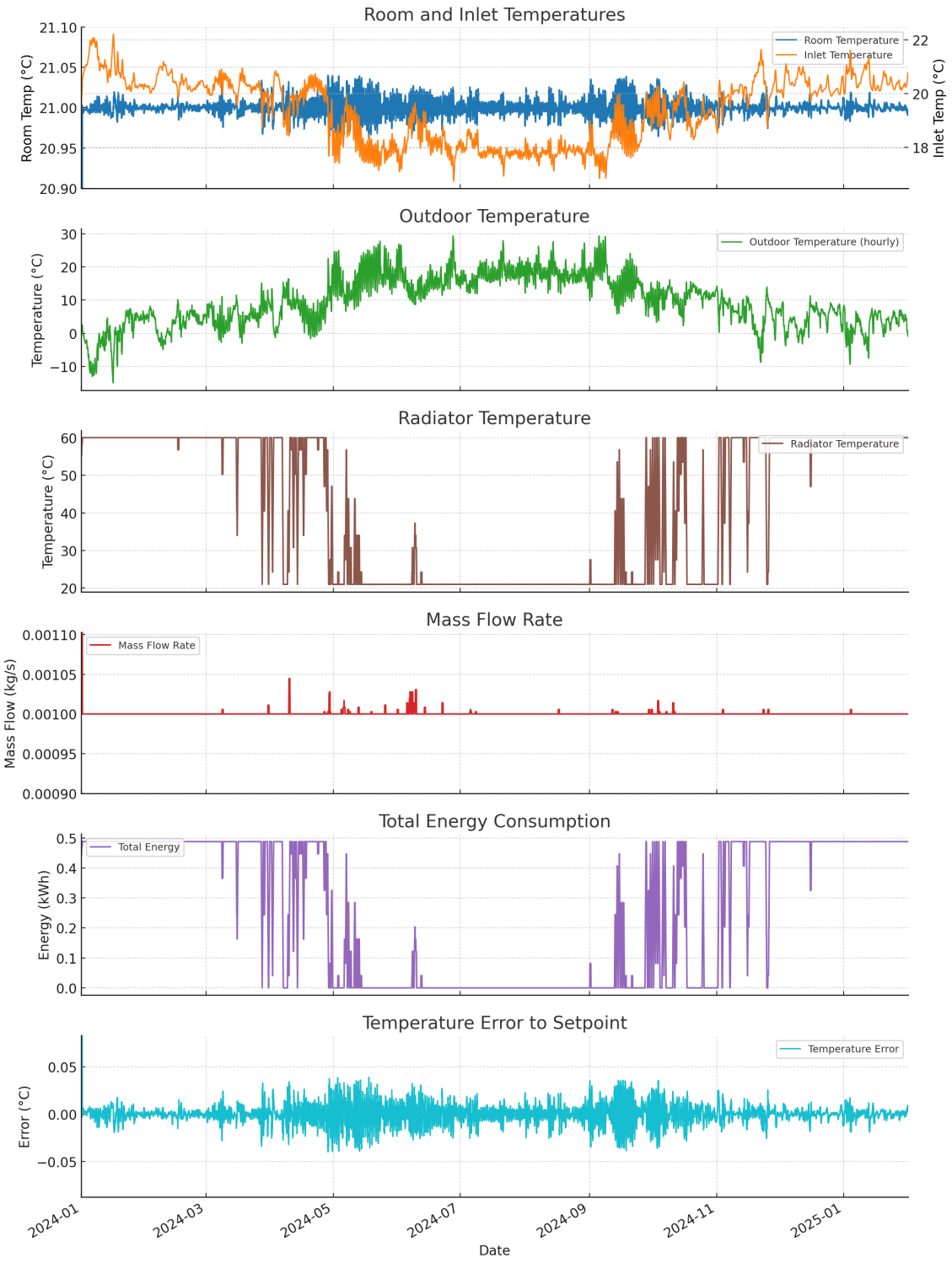


Figure 4.22: Year-long PID controller outputs: room/inlet temperatures, outdoor temperature, radiator temperature, mass flow rate, total energy consumption, and temperature error to set point.

Fig.4.22 presents the response of the PID controller over one year of weather data. The weather data are the same as the one used to simulate the SAC controller. The room temperature was maintained close to 21°C setpoint with minimal deviations. The primary control actions used to achieve these are the inlet-airflow temperature and the radiator temperature
The influence of mass-flow rate on the control actions is minimal, with only a short-term increase to accommodate the weather fluctuations. The control in the radiators demonstrated seasonal dynamics similar to those of the SAC controller, with higher values in late autumn and winter conditions and near-zero contributions during warmer weather. Since the radiator has a greater influence on the energy consumption, the energy consumption curve follows a similar trend to the radiator temperature, together with the energy consumption to achieve the necessary inlet airflow temperature.
The PID controller achieves the desired setpoint temperature with minimal possible error. However, due to strong dependence on the radiator and the inlet-airflow temperature, this controller provides a less adaptive strategy, and energy consumption is not considered by this controller. The following section provides a direct comparison between the SAC controller and the PID controller.

4.2.4 PID vs SAC Controllers

A comparative analysis is conducted on the performance of the SAC controller and the PID controller, which were discussed before. This section provides a detailed understanding of how these two control strategies performed under the same weather conditions to achieve thermal comfort and energy efficiency. Fig.4.23 and Fig.4.24 give a visual representation of the comparison in terms of cumulative temperature error and cumulative energy consumption.

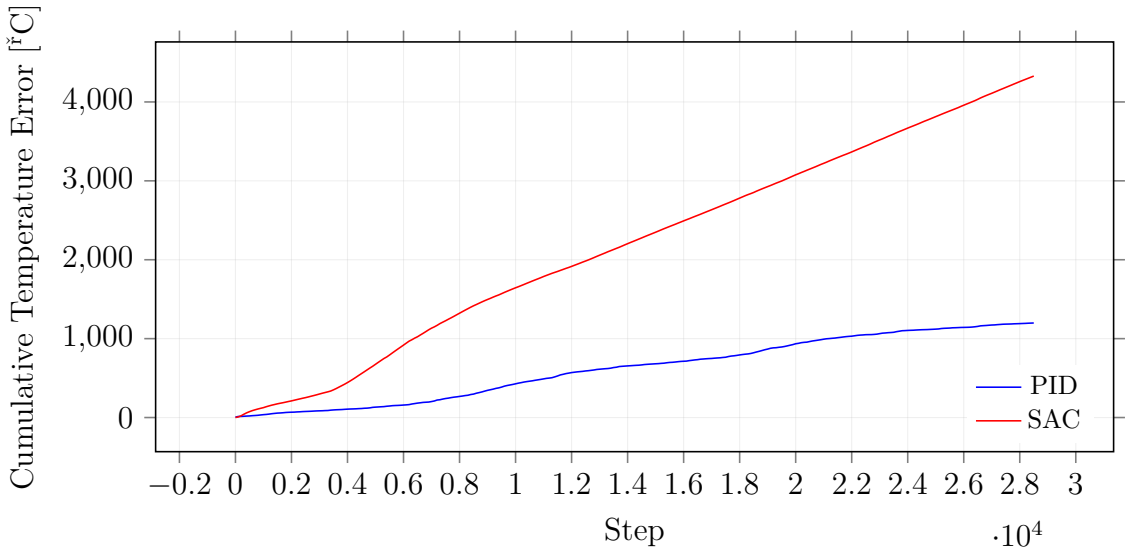


Figure 4.23: Cumulative temperature error over a year for PID and SAC controllers.

The PID controller achieved a cumulative temperature error of 1,200°C and the SAC controller accumulated error of, 4,340°C. This difference shows that the PID maintained tighter control to the 21°C setpoint, with deviations generally within ±0.5°C comfort band.

By contrast, the SAC controller occasionally permitted larger deviations, particularly during erratic weather conditions. Notably, the room temperature deviated by more than ±2.5°C from the setpoint in only 2.3% of the whole control action, making the extreme deviations relatively rare. This results from the controller's policy to balance energy efficiency against thermal comfort.

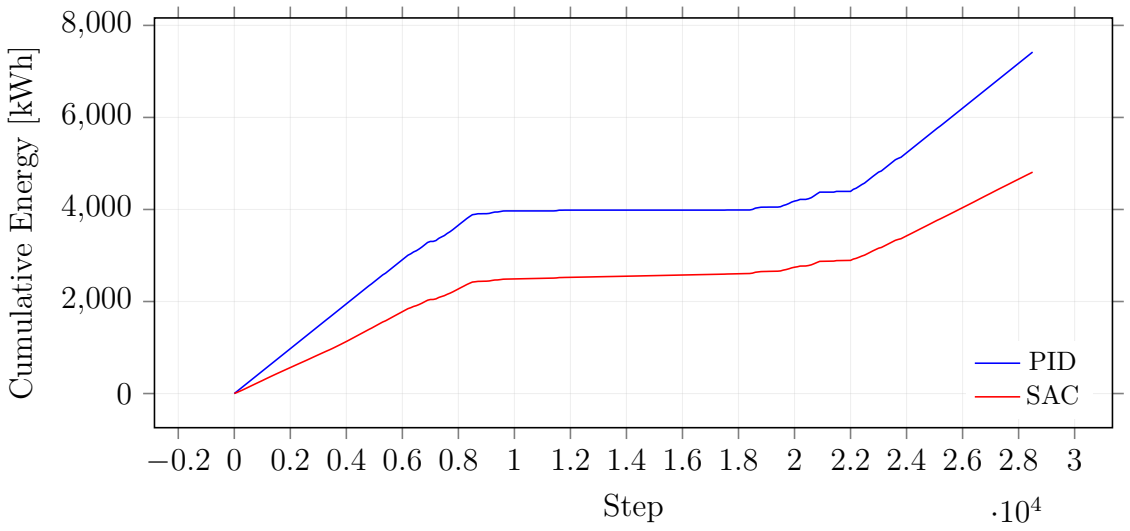


Figure 4.24: Cumulative energy consumption over a year for PID and SAC controllers.

However, the cumulative energy consumption reflects the other way. The PID controller consumed, 7,460 kWh over one year, compared with 4,836 kWh for SAC controller. This corresponds to a 35% reduction of overall energy demand. This is because the PID controller had one primary target to achieve the set point temperature using different input boundary conditions. The SAC controller, on the other hand, optimizes control actions by prioritizing energy efficiency and exploiting outdoor conditions to reduce energy costs. In the first eight months of the year, the PID controller consumed 4,182 kWh compared with the 2,745 kWh for SAC, marking a 34% saving. In the remaining period, PID consumed another 3,277 kWh and the SAC controller consumed only 2,091 kWh of energy. This corresponds to 36% saved energy, compared with 34% before. This shows that the relative energy efficiency is improves with the SAC controller as it learns from the past actions and improves over time. Unlike the PID controller, which provides deterministic control to reduce the error signal, the SAC controller provides stochastic, probabilistic actions based on the past control actions and continuous update in the policy, maximizes the cumulative reward. This enables the controller to provide better control actions, further enhancing energy efficiency,

5

Conclusion

This thesis examined the integration of CFD simulations with data-driven techniques for HVAC optimization, particularly through the development of a Fourier Neural Operator (FNO) surrogate model coupled with a Soft Actor-Critic (SAC) reinforcement learning controller. The framework has been evaluated in comparison to a traditional PID controller to assess its ability to optimize energy efficiency and thermal comfort in a dynamic indoor environment.

Results demonstrated that the FNO surrogate model adequately reproduced the spatial distributions of temperature and velocity fields with high accuracy. The model accurately represented prominent physical flow characteristics across various cross-sections, and the analysis of error distribution indicated that the majority of prediction errors were confined to lower variance, which is 6.72×10^{-4}m/s for velocity and 3.73×10^{-2} for temperature. Although temperature predictions were broadly accurate across the dataset, velocity predictions displayed slight indications of overfitting during training, indicating the challenges of effectively capturing turbulent flow dynamics with reduced-order learning models. However, the surrogate delivered reliable real-time temperature field predictions that made the control-oriented framework computationally achievable.

The SAC controller demonstrated significant advantages over the baseline PID controller, using this predictive framework. The PID achieved closer agreement with the specified temperature and reduced cumulative comfort error, however, but with significantly higher consumption of energy. The SAC controller, in contrast, effectively utilized environmental conditions to minimize unnecessary energy consumption, resulting in a 35% decrease in overall energy consumption over the period of one year. Significantly, extreme temperature fluctuations were uncommon (2.3% of all time steps), indicating that energy efficiency was achieved without drastic loss of comfort. A temporal analysis indicated that the advantages of SAC were more noticeable in the latter portion of the year, as the controller's policy adaptation improved long-term energy efficiency compared to the deterministic PID strategy.

These findings highlight an essential trade-off: conventional PID controllers prioritize short-term thermal stability but result in elevated operational expenses, while reinforcement learning approaches can identify policies that more effectively balance comfort with energy requirements in a historically adaptive manner. The integration of CFD-informed surrogate modelling and data-driven control demonstrates potential for practical HVAC systems, especially in scenarios where long-term sustainability and reduced operating costs are favoured over rigid adherence to comfort set points.

The thesis presents an extensive pipeline—from CFD data generation and surrogate modelling to reinforcement learning control and benchmarking—that demonstrates the potential of machine learning for improving HVAC optimization. Despite current obstacles, particularly in enhancing surrogate fidelity for velocity fields and improving reward weight formulations for reinforcement learning, the findings effectively support the integration of

hybrid physics-informed and data-driven approaches in the future design and operation of cooling and heating systems.

5.1 Future Work

This study illustrates the possibility of integrating a Fourier Neural Operator (FNO) surrogate with a Soft Actor–Critic (SAC) reinforcement learning controller for HVAC optimization. However, numerous possibilities can be explored for expanding the scope and improving the depth of this study. These can be organized into CFD-related extensions, machine learning model development, and controller design.

5.1.1 CFD

The main focus of the CFD framework established during this work was on the simplified domain of air temperature and velocity distributions. With the objective to directly assess perceived comfort, future improvements could couple the model with more comprehensive human thermal comfort metrics, such as predicted mean vote (PMV) or predicted percentage dissatisfied (PPD). To more effectively capture external heat loads, particularly in areas with large glazed surfaces, solar-radiation models could be included. Furthermore, incorporating humidity and CO_2 transport to the CFD setup could allow for the study of both thermal comfort and indoor air quality. Additionally, time-dependent thermal dynamics could be captured by running transient CFD simulations, which would result in richer datasets that capture unsteady flow phenomena.

5.1.2 Flow and Thermal Prediction Model

Even though the FNO surrogate used in this study showed good predictive performance, it is still bound for further development. The model could possibly be able to capture finer-scale physical interactions if the architecture is made more complex, for instance by deepening the network or increasing the number of Fourier modes. Training stability could be further enhanced and generalization error would be decreased with systematic hyperparameter optimization, such as Bayesian optimization. Furthermore, training the surrogate on higher-resolution CFD data would give it access to more intricate flow and thermal characteristics. Recurrent neural networks (RNN) architecture or introducing physics-informed loss function to the existing FNO model could be used to predict flow fields resolving smaller length scales, enabling the surrogate to capture time-dependent variations and memory effects beyond static snapshots. Additionally, Physics-Informed Neural Networks (PINNs) can be considered to capture physical variations in the flow field and thereby improving the surrogate model's ability to generalize better than data-driven surrogate model.

5.1.3 Reinforcement Learning Controller

Though there are still a number of opportunities for improvement, the SAC controller effectively optimized HVAC operations across a variety of weather and comfort scenarios. Instead of being fixed with a constant coefficient, the exploration–exploitation trade-off could dynamically adapt with the introduction of learnable entropy temperature terms. Similarly, more accurate and consistent control outputs may result from focused tuning

of actor network hyperparameters. Weather forecasts could give the controller predictive context, enhancing its capacity to predict changes instead of only responding to the current state. The control strategy would be more in line with actual building usage patterns if the framework were expanded to incorporate occupancy data and time-based occupancy patterns. Finally, the adaptability of reinforcement learning (RL) could be combined with the constraint-handling and stability guarantees of Model Predictive Control (MPC) hybrid framework to create a more reliable real-world implementation.

Bibliography

[1] L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and Buildings*, vol. 40, no. 3, pp. 394–398, 2008. doi: 10.1016/j.enbuild.2007.03.007.

[2] A. Gharbi, M. Ayari, N. Albalawi, Y. E. Touati, and Z. Klai, "Intelligent HVAC Control: Comparative Simulation of Reinforcement Learning and PID Strategies for Energy Efficiency and Comfort Optimization," *Mathematics*, vol. 13, no. 14, p. 2311, 2025. doi: 10.3390/math13142311.

[3] H. Wang, Y. Cao, Z. Huang, Y. Liu, P. Hu, X. Luo, Z. Song, W. Zhao, J. Liu, J. Sun, S. Zhang, L. Wei, Y. Wang, T. Wu, Z.-M. Ma, and Y. Sun, "Recent Advances on Machine Learning for Computational Fluid Dynamics: A Survey," *arXiv preprint* arXiv:2408.12171, 2024. Available: arxiv.org/abs/2408.12171.

[4] L. Davidson, *Fluid Mechanics, Turbulent Flow and Turbulence Modeling*. Chalmers University of Technology, Gothenburg, Sweden, 2018.

[5] Siemens Digital Industries Software, *STAR-CCM+ User Guide*, Version 2023.1, 2023. Accessed via Siemens Support Center (login required), Sep. 9, 2025. Available: docs.sw.siemens.com

[6] J. H. Lienhard IV and J. H. Lienhard V, *A Heat Transfer Textbook*, 5th ed. Mineola, NY: Dover Publications, Dec. 2019. ISBN: 9780486837352, 784 pp. Available: ahtt.mit.edu

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN: 0132733501.

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint* arXiv:1412.6980, 2017. Available: arxiv.org/abs/1412.6980

[9] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier Neural Operator for Parametric Partial Differential Equations," *arXiv preprint* arXiv:2010.08895, 2021. Available: arxiv.org/abs/2010.08895

[10] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, Mar. 2012.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012. doi: 10.1145/3065386.

[12] X. Guo, W. Li, and F. Iorio, "Convolutional Neural Networks for Steady Flow Approximation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pp. 481–490, Aug. 2016. doi: 10.1145/2939672.2939738.

[13] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control," *Journal of Fluid Mechanics*, vol. 865, pp. 281–302, Feb. 2019. doi: 10.1017/jfm.2019.62.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018. Available: incompleteideas.net/book/the-book-2nd.html

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017. doi: 10.1109/MSP.2017.2743240.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint* arXiv:1509.02971, 2015. Available: arxiv.org/abs/1509.02971

[17] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. doi: 10.1007/BF00992698.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. Available: deeplearningbook.org

[19] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. doi: 10.1109/72.279181.

[20] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *arXiv preprint* arXiv:1511.07289, 2016. Available: arxiv.org/abs/1511.07289

[21] Hicham Johra, *Overview of the Coefficient of Performance (COP) for conventional vapour-compression heat pumps in buildings*, 2022.

[22] Mitsubishi Electric Corporation, *PLFY-P VBM-E Service Manual*, 2022. `https://planetaklimata.com.ua/instr/Mitsubishi_Electric/Mitsubishi_Electric_PLFY-P_VBM-E_Service_Manual_Eng.pdf`

[23] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*, arXiv preprint arXiv:1801.01290, 2018. `https://arxiv.org/abs/1801.01290`

[24] K. J. Åström and T. Hägglund, "The future of PID control," *Control Engineering Practice*, vol. 9, no. 11, pp. 1163–1175, 2001. doi: 10.1016/S0967-0661(01)00062-4.

[25] Z. Li, "Review of PID control design and tuning methods," *Journal of Physics: Conference Series*, vol. 2649, no. 1, p. 012009, Nov. 2023. doi: 10.1088/1742-6596/2649/1/012009.

[26] R. de Dear, G. Brager, and D. Cooper, "Developing an Adaptive Model of Thermal Comfort and Preference – Final Report on RP-884," *ASHRAE Transactions*, vol. 104, 1997.

[27] X. Yuan, Q. Chen, and L. R. Glicksman, "A critical review of displacement ventilation," *ASHRAE Transactions*, vol. 107, no. 1, pp. 78–90, 2001.

[28] J. A. Myhren and S. Holmberg, "Design considerations with ventilation-radiators: Comparisons to traditional two-panel radiators," *Energy and Buildings*, vol. 41, no. 1, pp. 92–100, 2009. doi: 10.1016/j.enbuild.2008.07.014.

[29] S. Liu, M. Koupriyanov, D. Paskaruk, G. Fediuk, and Q. Chen, "Investigation of airborne particle exposure in an office with mixing and displacement ventilation," *Sustainable Cities and Society*, vol. 79, p. 103718, Apr. 2022. doi: 10.1016/j.scs.2022.103718.

[30] F. Memarzadeh, "Comparison of operating room ventilation systems in the protection of the surgical site," *ASHRAE Transactions*, vol. 110, no. 2, 2004.

[31] P. V. Nielsen, "Fifty years of CFD for room air distribution," *Building and Environment*, vol. 91, pp. 78–90, 2015. [Fifty Year Anniversary for Building and Environment]. doi: 10.1016/j.buildenv.2015.02.035.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY