Politecnico di Torino

ICT in Building Design: Energy Optimization, Prediction, and Control

Professors:

Giacomo Chiesa, Lorenzo Bottaccioli

Student Group 4:

Elnaz Zare - s327086

Fatemeh Sangin - s329999

Elahe Fallahi - s329577

Ehsan Soltanmohammaloo - s327039

Table of Contents

1. Introduction 2
1.1. Input Files Used in This Project 3
1.2. Building Energy Model and Simulation Setup 4
1.3. Structure of This Report and Project 6
.2 Design Optimization 8
2.1. Preparation for Use of Surrogate Models 9
2.1.1. Design Parameters and Sampling Strategy 9
2.1.2. Simulation Execution and Output Collection 11
.2.1.3 Simulation Outputs and Data Preparation 11
2.2. Surrogate Model and Multi-Objective Optimization with NSGA-II 14
.2.2.1 Overview and Motivation 14
.2.2.2 Mathematical Formulation of the Problem 15
2.2.3. Surrogate Model Setup 17
2.2.4. Multi-Objective Optimization with NSGA-II 18
2.3. Pareto Front Quality Assessment Using Hypervolume 20
2.4. Final Design Selection for Simulation and Validation 25
.2.5 Modifying the IDF for Final Simulation 26
2.6. Conclusion 26
.3 Stepped Simulation with Input/Output 27
3.1. Preparing and Running Basic Simulations 29
3.2. Offline Stepped Simulation Using Gym 31
3.3. Online Stepped Simulation with Real-Time Control 33
3.3.1. Detailed Process Description 33
3.3.2. Sensors and Actuators in Online Stepped Simulation 34
3.3.3. Importance and Challenges 36
3.4. Analysis of Adaptive Setpoint Control Strategies 37
3.4.1. January – Adaptive Policy 37
3.4.2. January – Adaptive Indoor Policy 38

3.4.3. July – Adaptive Indoor Policy ... 39
3.4.4. Comparative Analysis ... 39
Chapter 4: Building Management System (BMS) ... 40
4.1 Chapter Introduction ... 40
4.2 System Architecture ... 40
4.3 System Design and Configuration ... 41
4.3.1 Publisher ... 41
4.3.2 Mosquitto MQTT Broker ... 41
4.3.3 OpenHAB ... 41
4.3.4 InfluxDB and Grafana ... 42
4.4 Challenges and Solutions ... 42
4.5 System Performance and Outcomes ... 42
4.6 Energy Signature ... 45
4.7 Summary ... 46

Table of Figure

Figure 1: Course Project Overview........ 2
Figure 2: 3-D model from the building. 4
Figure 3: Inside look into the building and its six zones........ 4
Figure 4: Heatmap of LHS input parameter ranges showing distribution of sampled values across parameters before and after normalization of parameters. 10
Figure 5: Pairplot showing correlation between input parameters and energy outputs. 12
Figure 6: Correlation matrix heatmap showing the linear relationships between input parameters and energy performance outputs, highlighting key positive and negative correlations 13
Figure 7: Pair plot showing the Pareto front (blue dots) and dominated solutions (red crosses) among the 184 simulation-generated design configurations, before applying surrogate modeling and NSGA-II optimization. 22
Figure 8: Pairwise objective comparison plots after applying surrogate-assisted NSGA-II optimization, where all solutions (red crosses) are positioned on the Pareto front, indicating optimal trade-offs across objectives. 23
Figure 9: Overview of the stepped simulation workflow combining offline and online EnergyPlus executions with control policy evaluation........ 28
Figure 10: January – Adaptive Policy 38
Figure 11: January – Adaptive Indoor Policy 38
Figure 12 July – Adaptive Indoor Policy 39
Figure 13: *Real-time data flows from the Python publisher through MQTT and OpenHAB into InfluxDB, with Grafana visualizing the live system loop.*........ 41
Figure 14: embedded Grafana dashboard views in OpenHAB displaying real-time visualization of indoor zone metrics, global temperature and humidity, and system-level energy and environmental variables streamed through the integrated BMS. 43
Figure 15: Grafana dashboard visualizations comparing global and zone-level temperature trends over time, showcasing real-time data integration and cross-panel analysis within the BMS........ 44
Figure 16: Seasonal energy signatures for July and January........ 45

Table Of Tables:

Table 1: Key design parameters and their descriptions used in the optimization process, detailing their role in influencing the building's thermal and energy performance 9
Table 2: Summary of NSGA-II optimization progress, showing the number of generations, total evaluations, non-dominated solutions, epsilon values, and indicator types across the optimization process. 19

Abstract

This report presents the comprehensive development and implementation of a data-driven Building Management System (BMS) integrating detailed simulation, optimization, and real-time monitoring. The project begins with the creation of building energy models using EnergyPlus, capturing thermal, lighting, and HVAC performance under various operational scenarios. Advanced surrogate modeling techniques and multi-objective optimization, including NSGA-II, are applied to identify Pareto-optimal solutions that balance energy consumption and electricity consumption. Once the optimized designs are established, stepped simulations are conducted to generate high-resolution time-series data reflecting zone-level environmental conditions and system responses. These simulation outputs are then integrated into a fully containerized BMS infrastructure. A Python-based publisher streams the time-stamped data to a Mosquitto MQTT broker, with OpenHAB orchestrating data routing through configured `.things` and `.items` layers. Data persistence in InfluxDB ensures efficient time-series storage, while Grafana dashboards provide dynamic visualization and analysis. The system addresses key challenges, including data redundancy, service readiness, and performance optimization, resulting in a robust framework for intelligent building management and continuous feedback between virtual simulations and real-time analytical environments.

1. Introduction

Building energy efficiency has become a critical concern in addressing global energy challenges and reducing carbon emissions. Information and Communication Technologies (ICT) offer powerful tools for optimizing building design, predicting energy consumption, and implementing intelligent control strategies. This project focuses on applying these technologies to a residential apartment in Turin, Italy, with the goal of minimizing energy consumption while maintaining electricity consumption balance.

As referred to the course project overview brought in Figure 1, The project follows a comprehensive approach that includes:

1. **Design parameter optimization using surrogate modeling and Pareto analysis:** This phase involved the use of DesignBuilder and EnergyPlus, integrated with Python-based surrogate models, to efficiently explore the design space and identify optimal solutions that balance energy performance.

2. **Virtual Building Management System (BMS):** In this phase, a stepped simulation environment was developed using OpenAI Gym framework. Real-time data from EPW weather files and virtual sensors/actuators were integrated using a device connector based on RESTful APIs and MQTT protocols. The BMS includes an energy signature module for analyzing the building's thermal behavior and a prediction algorithm for forecasting energy demand. The system is further equipped with control strategies, such as adaptive setpoint adjustment and dynamic ventilation control, to optimize operational performance.

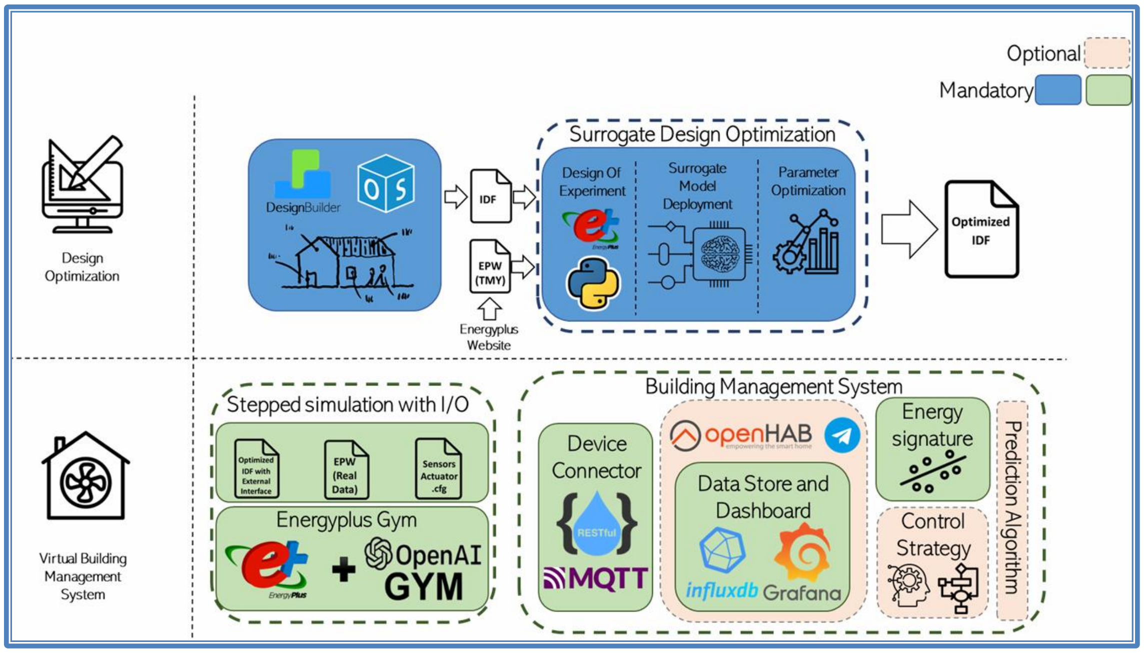


Figure 1: Course Project Overview

1.1. Input Files Used in This Project

The project setup relies on several key input files and paths:

idd_path: */usr/local/EnergyPlus-9-6-0/Energy+.idd*: the IDD file defining EnergyPlus data structures.

epw_path: *Torino_IT-hour.epw*: the local weather file used to provide climatic conditions for simulations.

idf_path:

- **Res_flat1.idf:** This is the original EnergyPlus input file provided by the instructor, representing the baseline residential apartment model before any optimization.
- **Res_flat1_optimized.idf:** This file was generated as the result of the design optimization phase, reflecting the improved configuration after applying surrogate modeling and multi-objective optimization methods. It incorporates the best-performing design parameters identified during the optimization process.
- **Res_flat1_optimized_hourly.idf:** This version is a slightly modified form of the optimized IDF, specifically adjusted to change the simulation timestep from the default 10-minute intervals to hourly outputs. This adjustment was made to facilitate more efficient post-processing and integration with the virtual building management system developed in the subsequent project phase.

energyplus_executer_path: /usr/local/EnergyPlus-9-6-0/energyplus: the EnergyPlus executable path used to run simulations.

The optimization process started from the base file provided by the instructor, *Res_flat1.idf*, representing the original residential apartment model.

After applying design optimization techniques, the improved file *Res_flat1_optimized.idf* was generated, reflecting the optimal design configuration.

Finally, *Res_flat1_optimized_hourly.idf* was derived from this optimized file, adjusting the timestep settings to provide hourly simulation outputs for use in further analyses and integration with the Virtual Building Management System.

1.2. Building Energy Model and Simulation Setup

The building under study, referred to as '*RES_Flat1'*, is a residential apartment modeled in IDF file. The model includes 7 thermal zones. Figure 2 and 3 give an insight of the entire building schema.

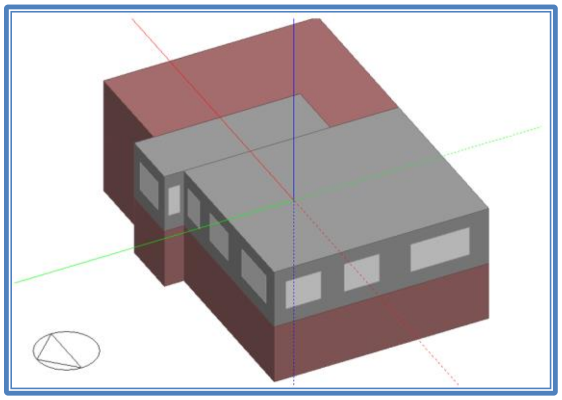


Figure 2: 3-D model from the building.

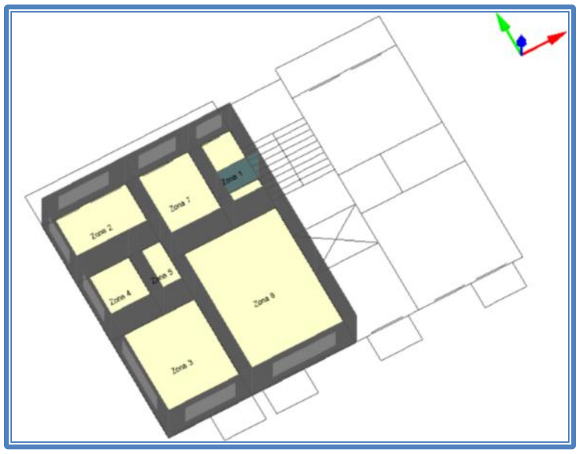


Figure 3: Inside look into the building and its six zones

.

The simulations were performed using EnergyPlus with the climate data for Turin, Italy.The weather file represents realistic hourly temperature, humidity, solar radiation, and wind data for an average year (epw file).

Key climate parameters include:

- Latitude: 45.22° N
- Longitude: 7.65° E

- Elevation: 287 meters
- Time Zone: UTC+1

The simulation setup includes:

- A full-year run period from January 1st to December 31st
- Timestep: 6 per hour (i.e., simulations run at 10-minute intervals. As mentioned earlier, we obtain another idf file that changes this setting to 1 per hour).

Simulations were executed using Python-based automation with the '*BESOS*' library to control parameter updates, simulation launches, and result parsing.

The model integrates key design parameters that influence the building's energy performance, focusing on both envelope characteristics and internal conditions. Specifically, the optimization included:

- **Insulation thickness:** Adjustable layers for materials such as *glass wool* ('MW Glass Wool (rolls)_.0001') for roof and and generic insulation ('_Insulation_.0001') for walls, with thickness ranges between 0.1–0.9 m for glass wool and 0.01–0.09 m for wall insulation layers.
- **Lighting power density:** Defined as watts per zone floor area, optimized within a range of 2–5 W/m^2 to balance lighting quality and energy use.
- **Ventilation rates:** Both *Air Changes per Hour (ACH)* and mechanical ventilation *outdoor air flow per person* were considered, with ACH ranging from 0–6 and outdoor air flow per-person from 0.01–0.15 m^3/s-person, allowing the model to explore the impact of natural and mechanical ventilation strategies.

This selection of parameters provided a balanced exploration of envelope performance, internal loads, and ventilation efficiency, ensuring the surrogate models capturing the most influential variables for the optimization process. It also ensures realistic and data-driven assessment of building thermal behavior in response to climate conditions and design changes.

1.3. Structure of This Report and Project

This report is organized into four main chapters, each addressing a key phase of the project workflow. Chapter 1 provides an introduction to the project goals, background context, and the relevance of integrating intelligent data-driven systems into building energy management. It outlines the motivation behind the study and sets the foundation for the technical components that follow.

Chapter 2 focuses on the optimization phase, describing the use of surrogate models and multi-objective optimization algorithms to refine building design choices for improved energy efficiency. Chapter 3 presents the stepped simulation process, detailing how adaptive policies and control strategies are applied to generate high-resolution time-series data. Finally, Chapter 4 describes the implementation of the Building Management System (BMS), including the configuration of MQTT communication, OpenHAB integration, InfluxDB storage, and Grafana visualization, culminating in a fully connected and automated data monitoring loop.

The files and folder are all zipped into smart-home.zip. Once extracted, you can find each chapter's python and necessary files as follows:

Chapter 2:

- ./data/Step1_Design_Optimization_v1.ipynb
- ./data/surrogate_feed_simulation_outputs.csv (we use it in the above notebook file to avoid executing energyplus for generated samples and save time for further plots and analyses).

Chapter 3:

- ./data/Step2_1_Create_Energyplus_Simulation_v1.ipynb
- ./data/Step2_2_Stepped_Simulation_offline_v1.ipynb
- ./ Step2_3_Online_Stepped_Simulation_v1.ipynb

Chapter 4:

- ./data/Step3_1_Seperate_Simulation_Output_Files_v1.ipynb
- ./data/Step3_2-Energy_Signature_v1.ipynb
- ./docker-compose.yml
- ./grafana
- ./influxdb
- ./mosquito
- ./openhab
- ./publisher (The service is commented on the ./docker-compose file to avoid redundant running. There is already a set of data in influxdb influenced by one time running the publisher).

- (The influxdb resides into ./influxdb folder whose mapping can be found inside its respective service definition in the ./docker-compose.yml file)

2. Design Optimization

This chapter presents the design optimization phase of the project, which aims to improve the building's energy performance while balancing electricity consumption and operational efficiency. Given the high computational cost of detailed EnergyPlus simulations, the approach integrates advanced surrogate modeling techniques with multi-objective optimization algorithms to efficiently explore the design space and identify optimal solutions.

The process involves several key steps:

Preparation for Use of Surrogate Models

To avoid the need for running thousands of heavy EnergyPlus simulations, surrogate models were developed. These simplified, data-driven models approximate the building's energy performance based on a limited set of simulation results, drastically reducing computation time.

Surrogate Model and Multi-Objective Optimization with NSGA-II

A multi-objective optimization approach was applied using the NSGA-II (Non-dominated Sorting Genetic Algorithm II) algorithm to identify Pareto-optimal solutions that balance competing objectives, such as minimizing energy consumption.

Pareto Front Quality Assessment Using Hypervolume

The quality of the Pareto front was evaluated using quantitative indicators like the hypervolume metric, which measures the spread and convergence of the solution set, ensuring robust optimization results.

Final Design Selection for Simulation and Validation

From the optimized set of Pareto solutions, a final design was selected for detailed simulation and validation, ensuring a balanced compromise among objectives and preparing it for implementation.

Modifying the IDF for Final Simulation

The selected design parameters were applied to the EnergyPlus IDF file, updating it to reflect the optimized configuration in preparation for the final detailed simulation phase.

Conclusion

The chapter concludes by summarizing the effectiveness of the surrogate-based optimization approach, highlighting the improvements achieved, and outlining the readiness for integration into the virtual building management system development phase.

2.1. Preparation for Use of Surrogate Models

In the context of building energy modeling, running detailed simulations using tools like EnergyPlus can be extremely time-consuming, especially when performing parametric or optimization studies that require hundreds or thousands of simulations. To overcome this challenge, surrogate models were developed and employed in this project.

A surrogate model is a simplified, data-driven approximation of a complex simulation model. Instead of running full EnergyPlus simulations for every design iteration, we first generated a representative dataset by running a limited set of carefully selected simulations. These results were then used to train the surrogate models, which can predict building performance (e.g., heating and cooling energy needs, electricity consumption indicators) for new design configurations at a fraction of the computational cost.

In this project, we explored machine learning-based surrogate modeling approaches to capture the nonlinear relationships between design parameters (e.g., insulation thickness, ventilation rates, etc.) and building performance outputs.

Overall, the use of surrogate models significantly accelerated the optimization process while maintaining acceptable prediction accuracy, enabling an efficient exploration of the design space.

2.1.1. Design Parameters and Sampling Strategy

Five design parameters were chosen for optimization, based on their direct impact on thermal performance and energy use are brought in **Error! Reference source not found.**.

Table 1: Key design parameters and their descriptions used in the optimization process, detailing their role in influencing the building's thermal and energy performance

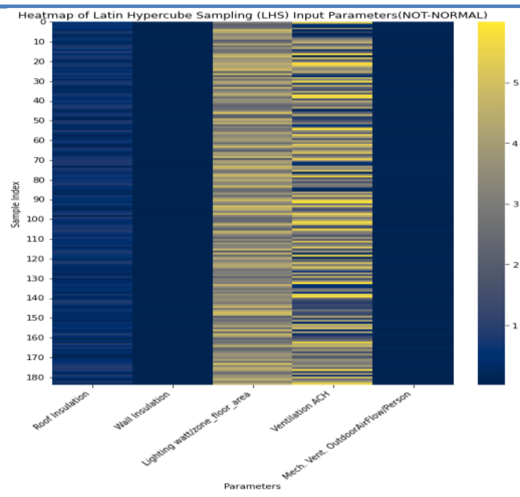
Parameter	Description
Roof insulation thickness	Affects heat exchange through the roof
Wall insulation thickness	Controls thermal transmission through walls
Lighting power per zone floor area (W/m²)	Influences electrical consumption for lighting
Natural ventilation (ACH)	Air change rate due to window-driven ventilation
Mechanical ventilation (L/s/person)	Outdoor air supplied by mechanical ventilation

Latin Hypercube Sampling (LHS): This sampling technique [1] was used to generate combinations of the above parameters, ensuring a well-distributed design space with minimal computational expense. This method allowed over **184** simulations covering a broad and statistically significant range of parameter values.

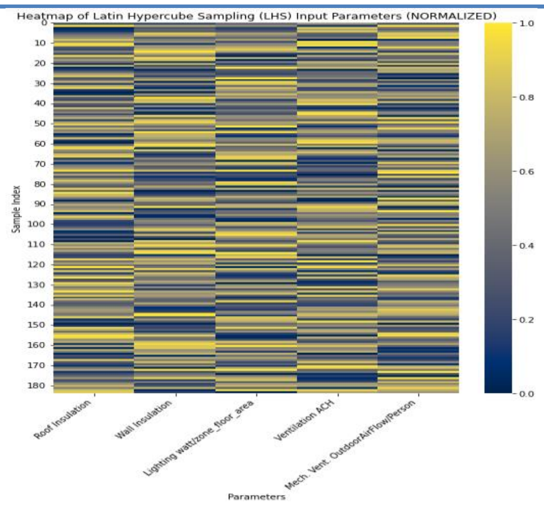
The heatmap presented in Figure 4 provides a detailed visualization of the LHS approach applied to the input parameters of this study. On the vertical axis, each row represents one of the generated LHS samples, while on the horizontal axis, each column corresponds to a specific input parameter (e.g., Roof Insulation, Wall Insulation, Lighting watt/zone_floor_area, Ventilation ACH, and Mechanical Ventilation Outdoor Air Flow per Person). The color gradient, ranging from dark tones (low values) to bright yellow (high values), indicates the numerical magnitude of the parameter values, as shown by the color bar on the right.

The importance of this analysis lies in confirming that the LHS method was correctly implemented, ensuring that the simulation results account for diverse scenarios without bias.

In conclusion, the heatmap serves as a crucial validation step, demonstrating the robustness of the experimental design and providing confidence in the representativeness of the obtained results



a: *raw (non-normalized) parameter values, directly reflecting their original scales.*



b: *presents the same dataset but after normalization (rescaling values between 0 and 1) to allow a direct comparison across parameters of different units and magnitudes.*

Figure 4: Heatmap of LHS input parameter ranges showing distribution of sampled values across parameters before and after normalization of parameters.

2.1.2. Simulation Execution and Output Collection

Following the Latin Hypercube Sampling (LHS) procedure to generate a diverse set of design parameter combinations, the next step involved evaluating each sampled design configuration using EnergyPlus simulations. This was achieved by creating an **EvaluatorEP** object, which encapsulates the optimization problem, the building model, the simulation directories, and the weather file (EPW).

The `evaluator.df_apply(samples, keep_input=True)` function was then used to apply the simulation runs to all sampled configurations, systematically executing each design case and collecting the resulting outputs.

To ensure proper documentation and later use in surrogate modeling, the generated outputs were stored in structured CSV files using the `save_outputs_to_csv(outputs)` function.

This step provided a complete dataset of energy performance results across the sampled design space, ready for further preparation, normalization, and integration into the surrogate modeling and optimization workflow.

2.1.3. Simulation Outputs and Data Preparation

After executing the EnergyPlus simulations across all sampled design configurations, the next critical step involved extracting and organizing the simulation outputs for further analysis. Specifically, three key performance metrics were collected from each simulation run:

- **Electricity: Facility (Joules)**
- **DistrictHeating: Facility (Joules)**
- **DistrictCooling: Facility (Joules)**

These outputs represent the building's total electrical consumption, heating demand, and cooling demand, respectively—essential indicators for assessing the energy performance of each design configuration.

The prepared dataset was then analyzed using pairwise correlation plots in Figure 5, providing initial insights into the relationships between input parameters (such as insulation thickness, lighting power density, ventilation rates) and key energy performance outputs. These scatterplots allowed us to detect broad trends, nonlinear relationships, and potential clusters or outliers across the dataset.

In addition to the pairplot, a correlation matrix heatmap (Figure 6) was generated to quantitatively assess the strength and direction of linear relationships between all input and output variables. For example, the matrix revealed that mechanical ventilation outdoor airflow per person showed a strong positive correlation (above 0.9) with district heating demand, indicating that increased ventilation rates significantly raise heating needs due to the additional load of conditioning outdoor air. Conversely, roof insulation thickness showed a moderate negative correlation (around -0.4) with district cooling demand, suggesting that better-insulated roofs help reduce cooling loads by limiting solar heat gains.

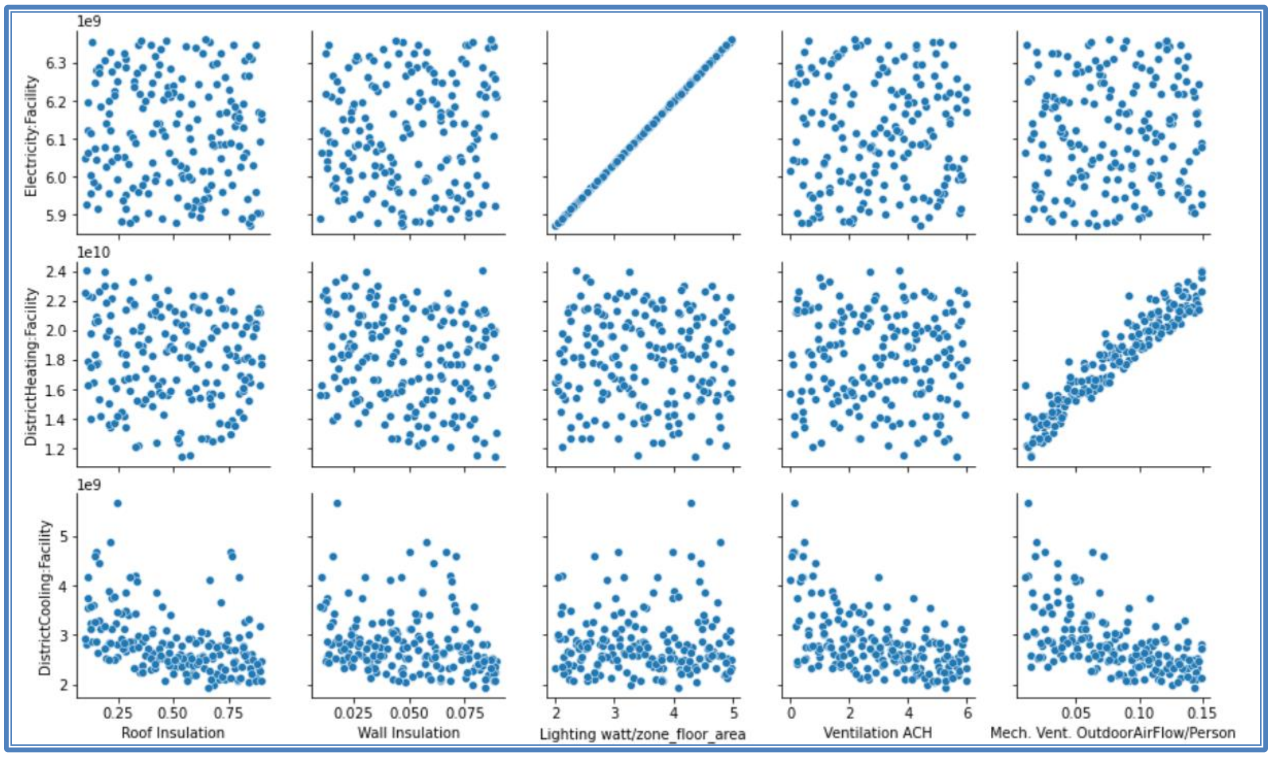


Figure 5: Pairplot showing correlation between input parameters and energy outputs.

Interestingly, some parameters, such as lighting power density, showed minimal correlation with heating or cooling demand, highlighting their relatively isolated impact on electricity consumption. This combination of visual and numerical analysis helped prioritize which variables were most impactful, ensuring that the surrogate models would focus on the most meaningful interactions and avoid overfitting to spurious or weakly correlated features.

Together, these exploratory analyses laid the groundwork for the subsequent surrogate modeling and multi-objective optimization phases, ensuring that the models would effectively capture the most influential variables, interactions, and trade-offs required for robust design decision-making.

Additionally, the dataset was normalized using **Min-Max scaling** [2], a preprocessing technique that transforms all input and output variables to a common range, typically between 0 and 1. This method preserves the relationships between data points while eliminating differences in scale that could bias the surrogate models during training. By ensuring that all features contribute proportionally, Min-Max normalization improves the numerical stability of machine learning algorithms, accelerates the convergence of optimization procedures, and enhances the interpretability of the results. This step was essential for preparing the dataset for reliable surrogate modeling and robust multi-objective optimization.

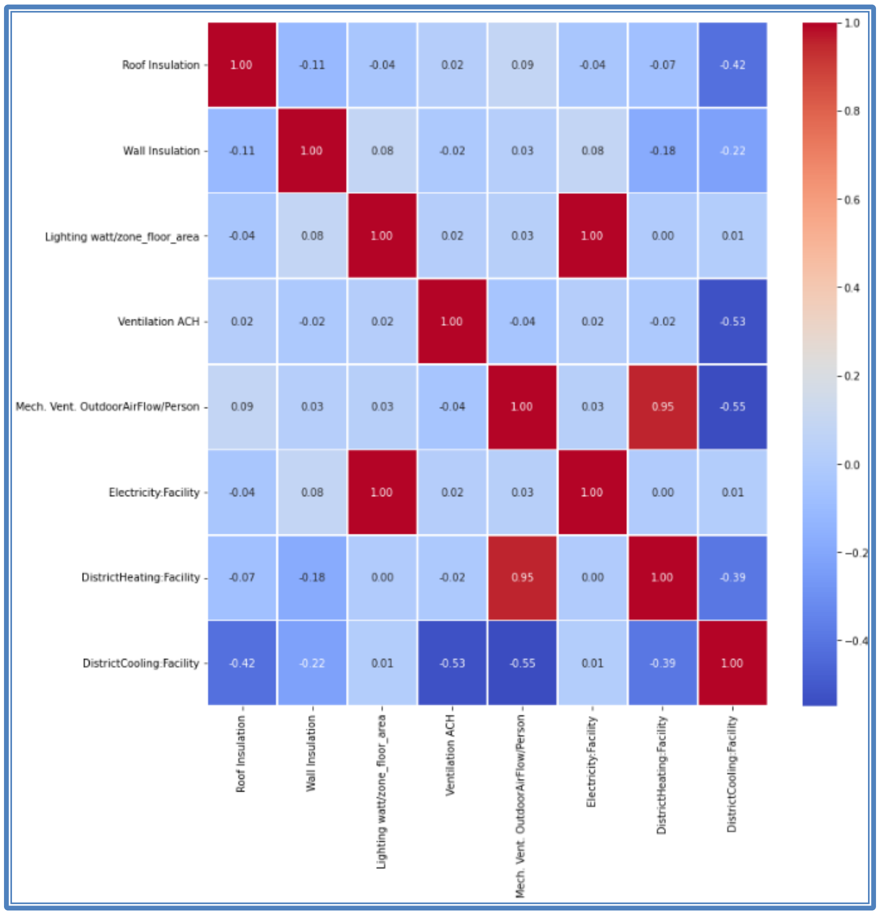


Figure 6: Correlation matrix heatmap showing the linear relationships between input parameters and energy performance outputs, highlighting key positive and negative correlations

In summary, Section 2.1 detailed the preparation phase required for implementing surrogate models in the design optimization process. By carefully selecting key input parameters, generating representative design samples using Latin Hypercube Sampling (LHS), and executing EnergyPlus simulations, we established a robust dataset linking design configurations to energy performance outcomes. This dataset was preprocessed through unit conversion, normalization (using Min-Max scaling), and exploratory analysis, ensuring that the surrogate models would be both reliable and computationally efficient.

With this foundation in place, we now proceed to Section 2.2, where the surrogate models are integrated into a multi-objective optimization framework. Here, the NSGA-II algorithm is employed to systematically explore trade-offs between competing performance objectives and to identify Pareto-optimal design solutions.

2.2. Surrogate Model and Multi-Objective Optimization with NSGA-II

Building upon the robust dataset and the carefully constructed surrogate models developed in the previous phase, Section 2.2 embarks on the application of advanced multi-objective optimization techniques, specifically leveraging the NSGA-II algorithm, to navigate the complex design space, uncover trade-offs between energy performance and electricity consumption, and systematically identify a diverse set of Pareto-optimal solutions that guide informed and balanced decision-making in the building design process.

2.2.1. Overview and Motivation

Building design inherently involves balancing multiple, often conflicting objectives — most notably minimizing energy consumption while ensuring acceptable levels for electricity consumption. For example, increasing insulation thickness might reduce heating and cooling loads but can simultaneously affect natural ventilation potential, indoor air quality, or even construction costs. As a result, **single-objective optimization** approaches are insufficient.

To address these challenges, this project adopts a **multi-objective optimization** framework, where the aim is not to find a single “best” solution but rather to identify a diverse set of **Pareto-optimal** solutions. These are solutions where no objective can be improved without degrading at least one other. Such *Pareto fronts* offer valuable insights to designers and decision-makers, allowing them to explore a range of balanced trade-offs.

To perform this multi-objective optimization, the **Non-dominated Sorting Genetic Algorithm II (NSGA-II)** [3] was selected as the primary algorithm. NSGA-II is widely recognized for its efficiency, robustness, and ability to maintain a well-distributed Pareto front even in high-dimensional and non-convex problems.

Compared to other methods like Particle Swarm Optimization (PSO) [4] or Grid Search [5], NSGA-II offers distinct advantages:

- It naturally handles multiple objectives without requiring artificial weightings or scalarization.
- It employs genetic operators (selection, crossover, mutation) to explore the solution space effectively.
- Its non-dominated sorting and crowding distance mechanisms ensure both convergence and diversity in the solution set.

However, running NSGA-II directly on a complex building energy model, such as an EnergyPlus simulation, would demand thousands of computationally expensive runs, making the process impractical for most projects.

To overcome this limitation, the project integrates **surrogate models**, specifically **Gaussian Process Regressors (GPR)** [6], as computationally efficient approximations of the full simulation models.

Gaussian Process Regression (GPR) stands out among surrogate modeling techniques for several key reasons, making it particularly suitable for complex multi-objective optimization problems like building energy design. First, GPR is a **non-parametric, probabilistic model**, meaning it does not assume a fixed functional form for the relationship between inputs and outputs but instead learns this relationship directly from data. This flexibility allows GPR to capture highly nonlinear and intricate interactions between design variables (such as insulation thickness, ventilation rates, and lighting power density) and performance outputs, which are common in building energy systems.

Another advantage of GPR is its ability to provide **uncertainty quantification** alongside predictions. Unlike simpler surrogate models (e.g., linear regression or polynomial models), GPR not only predicts the expected value of an output but also estimates the confidence interval around that prediction. This feature is particularly valuable in optimization, as it enables the algorithm to balance exploration and exploitation — prioritizing regions of the design space where the surrogate model is most uncertain and thus where further improvements might be found. In the context of this project, using GPR ensures that the surrogate-guided NSGA-II optimization can navigate the trade-off space more intelligently, leading to better convergence toward true Pareto-optimal solutions with fewer computational resources.

Surrogate models are trained on a representative set of EnergyPlus simulation results and can predict system responses with high accuracy but at a fraction of the computational cost.

By combining the exploration power of NSGA-II with the speed and efficiency of surrogate models, this approach enables a practical, scalable, and robust optimization workflow, making it feasible to generate high-quality, multi-objective design solutions within realistic timeframes and resource constraints.

2.2.2. Mathematical Formulation of the Problem

Building In this multi-objective optimization problem, the goal is to identify the optimal design configurations x that balance multiple, often conflicting, performance objectives y.

Input Vector (x): Design Parameters

Let $x = [x_1, x_2, \dots, x_n]$ represent the vector of decision variables (design parameters), where each x_i corresponds to a specific controllable aspect of the building design. In this project, the primary input variables include:

- x_1 : thickness of roof's glass wool insulation (m)
- x_2 : thickness of wall insulation layer (m)
- x_3 : lighting power density (W/m²)
- x_4 : air changes per hour (ACH)
- x_5 : outdoor air flow per person (m³/s-person)

Each variable x_i is bounded within a specific range S_i, defined based on physical, regulatory, or practical constraints.

Output Vector (y): Performance Indicators

Let $y = [y_1, y_2, \dots, y_m]$ represent the vector of output objectives, where each y_i quantifies a specific performance indicator. In this project, the primary outputs include:

- y_1 : annual heating energy demand (kWh)
- y_2 : annual cooling energy demand (kWh)
- y_3 : electricity consumption

The relationship between inputs and outputs is captured by an unknown, complex function $f(.)$ approximated using the surrogate model (GPR):

$$f(x) = y, \qquad f: x \rightarrow y$$

Optimization Problem Statement

The multi-objective optimization problem is formulated as:

$$minimize\ F(x) = [f_1(x), f_2(x), \dots, f_m(x)]$$

Subject to:

$$G_k(x) \geq 0, \quad k = 1, 2, \dots, p$$

And

$$x_i \epsilon S_i, \quad i = 1, 2, \dots, n$$

Where:

$f_j(x)$ are the objective functions (heating, cooling, electricity)

$G_k(x)$ represent problem-specific constraints

S_i are the feasible bounds for each decision variable x_i.

This formal structure ensures that the optimization process systematically explores the design space to identify Pareto-optimal solutions, balancing multiple objectives while respecting all technical and regulatory constraints.

2.2.3. Surrogate Model Setup

To efficiently approximate the computationally expensive EnergyPlus simulations, Gaussian Process Regression (GPR) was selected as the surrogate modeling technique for this project. GPR offers several key advantages, including its non-parametric nature, ability to model complex nonlinear relationships, and most importantly, its capacity to provide uncertainty estimates (confidence intervals) alongside predictions. This makes GPR particularly suitable for optimization tasks, where knowing both the predicted performance and the confidence in that prediction helps guide the exploration of the design space.

Given that the optimization problem involves **three distinct performance objectives**, heating energy demand, cooling energy demand, and electricity consumption, three separate GPR models were trained. Each model was specialized to predict one specific output, ensuring that the surrogate models could accurately capture the underlying patterns and sensitivities unique to each objective without interference from unrelated output noise.

Training Process

The training dataset consisted of input-output pairs (x, y_i) where x represents the design parameter combinations and $, y_i$ the simulation-derived output for each objective i. The GPR models were trained using a Matern kernel (with $v = 2.5$), known for its flexibility and smoothness properties, and a small regularization term ($\alpha = 1e - 6$) to ensure numerical stability.

The training process involved fitting the models on the normalized datasets, ensuring that the surrogate models learned not only the average trends but also the variance patterns present in the data. Cross-validation techniques were applied to assess generalization and prevent overfitting.

Summary Pseudocode

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import Matern
# Initialize models list
models = []
# Train separate GPR models for each objective
```

```
for i in range(3):  # 0: heating, 1: cooling, 2: electricity
    gpr = GaussianProcessRegressor(
        kernel=Matern(nu=2.5),
        alpha=1e-6,
        normalize_y=True
    )
    gpr.fit(X_train, Y_train[:, i])  # X_train: input samples, Y_train: target outputs
    models.append(gpr)
```

This modular approach allows the optimization framework (described in the next section) to query each surrogate model individually, making simultaneous predictions for all objectives based on a given design input x. The combination of accuracy, flexibility, and uncertainty awareness provided by GPR forms a critical foundation for the success of the multi-objective optimization process.

2.2.4. Multi-Objective Optimization with NSGA-II

To solve the multi-objective design optimization problem, this project employed the **Non-dominated Sorting Genetic Algorithm II (NSGA-II)**, a widely used evolutionary algorithm well-suited for generating diverse sets of Pareto-optimal solutions. NSGA-II is particularly powerful in problems where objectives conflict, because it does not collapse multiple objectives into a single weighted function but instead seeks a front of trade-off solutions.

NSGA-II Algorithm Steps

The NSGA-II algorithm works through the following main steps:

1) **Initialization:** Generate an initial random population of solutions P_0. Each solution is a vector of design parameters x.
2) **Evaluation:** Calculate the objective values $F(x)$ for each solution using the trained GPR surrogate models.
3) **Non-dominated Sorting:** Rank solutions into different Pareto fronts based on dominance relations — solutions in the first front are non-dominated, those in the second front are dominated only by the first, and so on.
4) **Crowding Distance Assignment:** Within each front, assign a crowding distance to maintain diversity by preferring solutions located in less crowded regions.
5) **Selection:** Use binary tournament selection based on rank and crowding distance to form a mating pool.
6) **Crossover and Mutation:** Generate offspring through crossover (combining parent solutions) and mutation (introducing random variations).
7) **Replacement:** Combine parent and offspring populations, sort them, and select the best individuals to form the next generation P_{t+1}.
8) **Termination:** Repeat the cycle for a predefined number of generations or until convergence criteria are met.

Implementation on Surrogate Models

In this project, the optimization was implemented as:

```
algorithm = NSGA2(pop_size=100)
res = minimize(problem, algorithm, ('n_gen', 100), verbose=True)
```

where `problem` encapsulates the surrogate models, and `n_gen` is the number of generations. The optimization explores the design space by querying the GPR models instead of expensive EnergyPlus simulations, enabling rapid exploration and convergence.

Output Interpretation: F, Nadir, Indicator

- **F (Objective Values)**: The final Pareto front's objective values, representing the achieved trade-offs between heating, cooling, and electricity use.
- **Nadir Point**: The worst (largest) objective values among the Pareto-optimal solutions, providing a reference for scaling and normalization.
- **Indicator**: A convergence metric used to evaluate how much the Pareto front improves between generations, e.g., via epsilon or hypervolume indicators.

Logs and Execution Table

The NSGA-II log records key metrics per generation, including:

Table 2: Summary of NSGA-II optimization progress, showing the number of generations, total evaluations, non-dominated solutions, epsilon values, and indicator types across the optimization process.

Generation(n_{gen})	Evaluation(n_{eva})	Non-dominated Solutions (n_{nds})	Epsilon (ϵ)	Indicator Type
1	100	17	-	-
2	200	19	0.024	Ideal
…	…	…	…	…
100	10000	100	0.011	Final

These logs track how the Pareto front evolves over generations, showing convergence toward optimal solutions and improvements in solution diversity.

Meaning of Each Metric:

- **Non-dominated Solutions (n_{nds}):** Number of solutions in the current Pareto front, reflecting solution diversity.
- **Epsilon (ϵ):** Change in Pareto front between generations, indicating convergence speed.
- **Indicator Type:** Marks whether the front is improving toward the ideal or stagnating (e.g., “ideal,” “nadir,” or “final”).

In summary, NSGA-II combined with GPR surrogate models enabled efficient exploration of the design space, generating a diverse, well-distributed set of Pareto-optimal solutions, which will be further analyzed and refined in the next sections.

2.3. Pareto Front Quality Assessment Using Hypervolume

To evaluate the quality of the Pareto front obtained from the NSGA-II optimization, this project employs the **hypervolume (HV) metric** [7] a widely used quantitative indicator in multi-objective optimization. The hypervolume measures the portion of the objective space that is dominated by the Pareto front relative to a predefined reference point. In simpler terms, it quantifies both the convergence (closeness to the ideal front) and the diversity (spread of solutions) achieved by the optimization process.

Why Use Hypervolume?

The hypervolume indicator is particularly valuable because:

- It captures both the **convergence** and **diversity** aspects of the Pareto front in a single scalar value.

- It is **Pareto-compliant**, meaning that any improvement in the Pareto front (in terms of dominance) will result in an increased hypervolume.

- It allows for **comparative evaluation** between different optimization runs or methods, making it easier to assess improvements or regressions.

Other common metrics, such as spacing or spread, typically measure only diversity, while convergence metrics often focus solely on distance to the ideal point. Hypervolume integrates both dimensions, offering a more holistic assessment.

Calculating HV for the NSGA-II Pareto Front

For this project, the hypervolume was computed over the final non-dominated set produced by the NSGA-II algorithm. The calculation required:

- Selecting a suitable **reference point** in the objective space, typically slightly worse than the worst observed solutions.
- Normalizing the objectives if necessary to ensure comparability across different scales
- Using established hypervolume algorithms (such as the WFG or Lebesgue measure) to calculate the dominated volume.

Analysis of Spread and Convergence

The obtained hypervolume value was analyzed to interpret:

- **Convergence**: How close the solutions are to the ideal (best) known performance.
- **Spread**: How well-distributed the solutions are across the objective space, indicating the range of trade-offs available.

A high hypervolume generally indicates both good convergence and good spread, meaning the optimization process succeeded in finding a well-balanced and diverse Pareto front.

Graphical Results and Interpretation

The project included visual plots of the Pareto front in objective space (e.g., heating vs. cooling vs. electricity) alongside hypervolume trends over generations. These plots provided visual confirmation of:

- The shape and coverage of the Pareto front.
- Whether there were clusters or gaps in certain regions.
- How the hypervolume improved over successive generations, reflecting optimization progress.

In the initial analysis, prior to applying the surrogate modeling and NSGA-II optimization, the dataset consisted of 184 unique design configurations generated through simulation. Among these, only a subset of solutions — as visualized in the

Figure 7 (pair plot) — were identified as Pareto-optimal, meaning they offered non-dominated trade-offs between electricity consumption, heating demand, and cooling demand. The remaining points were classified as dominated solutions, reflecting less optimal performance.

However, after implementing the surrogate model combined with the NSGA-II genetic algorithm, the resulting solution set displayed a markedly different pattern. As illustrated in the Figure 8, nearly all generated solutions were positioned on the Pareto front, indicated by the red crosses marking optimality across all objective comparisons. This shift highlights the strength of the surrogate-assisted optimization process, which effectively guided the search toward high-performing, non-dominated solutions. In essence, the optimization framework successfully navigated the design space to identify configurations that balanced all objectives simultaneously, vastly improving the proportion of optimal solutions compared to the unoptimized baseline.

Together, the hypervolume analysis and accompanying visualizations offered a robust evaluation of the optimization results, setting the stage for comparing alternative methods and selecting final designs.

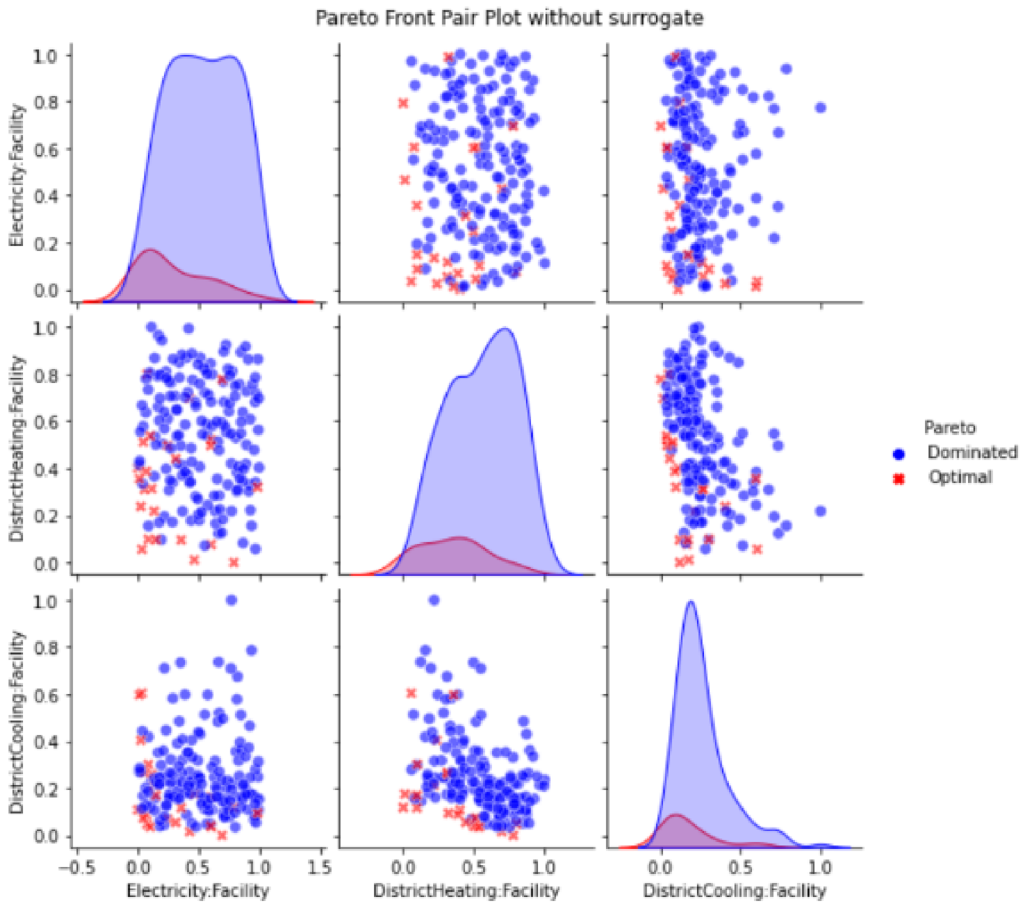


Figure 7: Pair plot showing the Pareto front (blue dots) and dominated solutions (red crosses) among the 184 simulation-generated design configurations, before applying surrogate modeling and NSGA-II optimization.

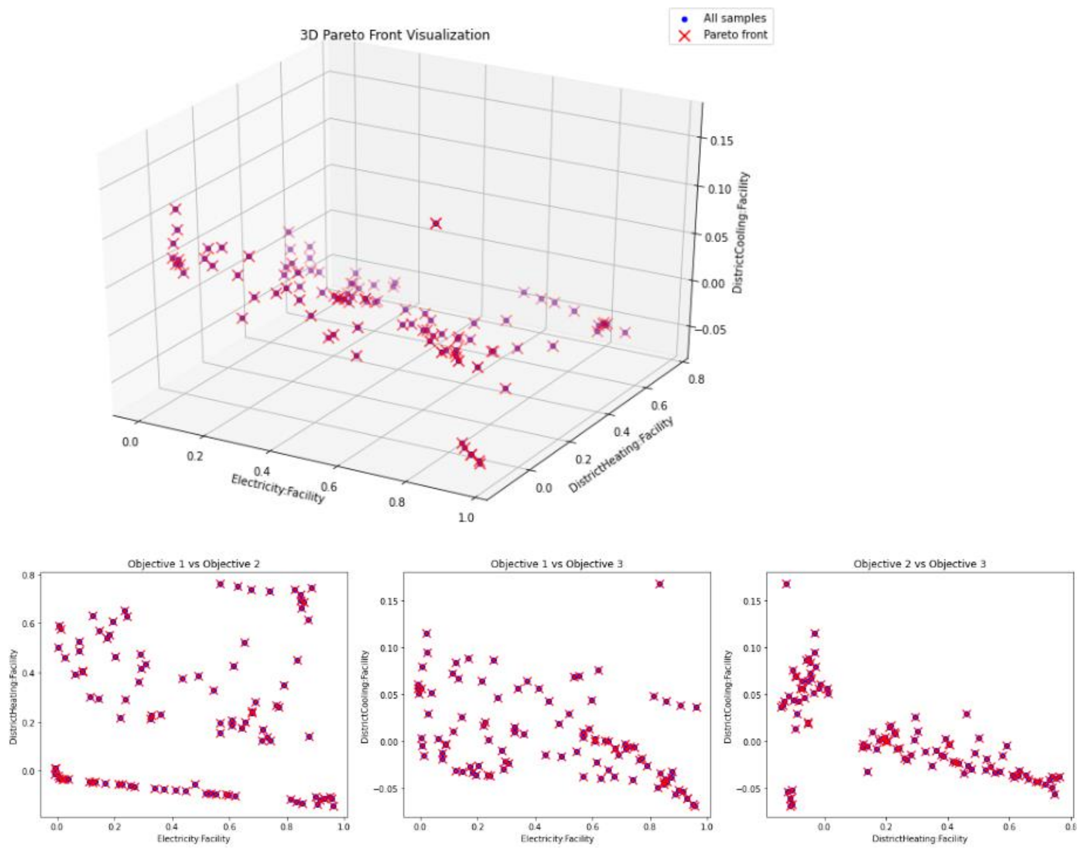


Figure 8: Pairwise objective comparison plots after applying surrogate-assisted NSGA-II optimization, where all solutions (red crosses) are positioned on the Pareto front, indicating optimal trade-offs across objectives.

Hypervolume Comparison and Interpretation

To quantitatively assess the quality of the Pareto fronts generated by the two approaches — direct EnergyPlus simulations versus surrogate-assisted NSGA-II optimization — we computed both the **absolute hypervolume (HV)** and the **hypervolume normalized by the number of points**.

Absolute Hypervolume:

- Surrogate + NSGA-II: 0.7801
- Direct EnergyPlus: 0.5704

The higher absolute hypervolume value for the surrogate-based method indicates that the total dominated space in the objective domain is larger, suggesting that the solutions found by the NSGA-II optimization cover a broader and potentially more optimal region closer to the ideal front compared to the direct simulation approach.

Hypervolume Normalized by Number of Points:

- Surrogate + NSGA-II: 0.007801
- Direct EnergyPlus: 0.023765

When normalizing the hypervolume by the number of Pareto solutions, the results show an interesting shift in interpretation. The EnergyPlus-derived front, although containing fewer points, demonstrates a **higher average contribution per point**, indicating that the individual solutions may carry more value or be better distributed across the Pareto space. In contrast, the surrogate-based front, while larger in total, consists of many points that each contribute relatively less to the overall volume, possibly due to denser clustering or local redundancies.

Summary Interpretation

- The surrogate-assisted optimization successfully identifies a wider set of high-performing solutions, expanding the search space and improving overall coverage.
- However, when focusing on **solution quality per point**, the direct EnergyPlus front reveals that even a smaller, more selective set of solutions can achieve high value in multi-objective trade-offs.

This analysis emphasizes the importance of balancing **quantity** and **quality** in Pareto front generation and suggests that combining both methods could yield the most robust design insights.

2.4. Final Design Selection for Simulation and Validation

After generating a well-distributed Pareto front using the surrogate-assisted NSGA-II optimization, the next critical step was to select a **single representative design** for detailed simulation, validation, and integration into the subsequent virtual building management system (BMS) phase. Selecting one point from the set of non-dominated solutions (Pareto front) is essential when moving from computational exploration to real-world implementation.

Methods for Selecting a Representative Solution

Four commonly used strategies were considered:

1-Minimum Total Cost (Sum-Based Selection)
Selects the point with the lowest total sum across all objectives.
Use case: When all objectives are equally important and on comparable scales.
Limitation: Can bias toward objectives with larger numeric ranges.

2- Closest to Ideal Point (Euclidean Distance) (Chosen Method)
Calculates the ideal point (minimum value in each objective) and selects the solution with the smallest Euclidean distance to this point.
Use case: Provides a balanced compromise across all objectives and avoids being skewed by a single extreme.
This method was chosen in this project for its objectivity and robustness.

3- Most Balanced Solution (Minimum Standard Deviation)
Selects the solution with the least variation among its own objective values.
Use case: When consistency across objectives is prioritized.

4- Weighted Aggregation (Custom Prioritization)
Applies custom weights to objectives and selects the solution with the smallest weighted sum.
Use case: When certain objectives are explicitly prioritized over others.

Pseudocode for Closest to Ideal Point Selection

```
ideal_point = F.min(axis=0)
distances_to_ideal = np.linalg.norm(F - ideal_point, axis=1)
best_idx_ideal = distances_to_ideal.argmin()
best_design_ideal =
df_pareto_with_real_scale.iloc[best_idx_ideal]
```

This approach ensured that the final chosen design scored strongly across all three key objectives (electricity, heating, cooling), offering a robust and balanced candidate for further evaluation.

Visualization of Selected Designs

As shown in the 3D and 2D plots (Figure 8), multiple selection strategies were visualized over the Pareto front, highlighting how different methods identify different points. In the final visualization, the “Closest to Ideal Point” solution is marked alongside alternatives for comparison. This not only confirms the robustness of the selected design but also allows stakeholders to understand trade-offs among top-performing configurations.

Preparing for Simulation and BMS Integration

The selected final design was prepared for detailed EnergyPlus simulation, ensuring all design parameters were properly set in the optimized IDF file. This simulation served as the final validation before transitioning into the virtual building management system (BMS) phase, where real-time control, monitoring, and further optimization strategies would be implemented.

2.5. Modifying the IDF for Final Simulation

After selecting the optimal design from the Pareto front, the next step was to apply the updated design parameters to the EnergyPlus input file (IDF) to prepare it for final detailed simulation. This involved modifying specific fields in the IDF, such as insulation thickness, lighting power density, ventilation rates, and setpoint temperatures, to match the values from the selected optimal solution.

2.6. Conclusion

This project demonstrated an effective workflow for integrating surrogate modeling and multi-objective optimization into the building design process. By combining Gaussian Process Regression (GPR) surrogate models with the NSGA-II genetic algorithm, we efficiently explored the complex trade-offs between energy consumption and electricity consumption, generating a diverse Pareto front of optimal solutions. Using quantitative metrics like hypervolume and well-defined selection strategies, we identified and validated a final design ready for detailed EnergyPlus simulation and subsequent integration into a virtual building management system (BMS). Overall, the approach showcased how advanced computational methods can accelerate sustainable and high-performance building design.

3. Stepped Simulation with Input/Output

Following the design optimization phase presented in Chapter 2, which culminated in the generation of an optimized EnergyPlus input file (**Res_flat1_optimized.idf**), this chapter focuses on the next stage: performing **stepped simulations** where detailed time-based building performance simulations are conducted.

While previous optimization tasks relied on surrogate models to efficiently evaluate performance across many configurations, this chapter centers on **direct simulations** that explicitly generate and process input/output data using EnergyPlus. This phase is essential for validating the optimized design under realistic operational scenarios and for preparing the system for integration with advanced control frameworks such as virtual building management systems (BMS).

The graphical abstract in Figure 9 summarizes the key steps:

- We begin by executing basic EnergyPlus simulations to produce detailed hourly outputs, confirming the proper functioning of the optimized IDF.
- The workflow then splits into two parallel paths:
  (1) an **offline stepped simulation** using precomputed data (eplusout.csv) in an OpenAI Gym environment to validate observation-action-reward mechanisms,
  and (2) an **online stepped simulation**, where real-time adaptive control strategies dynamically update IDF files and rerun EnergyPlus simulations, creating a closed-loop system.

Finally, the results from both approaches are analyzed and compared, providing critical insights into the performance impacts of different control policies.

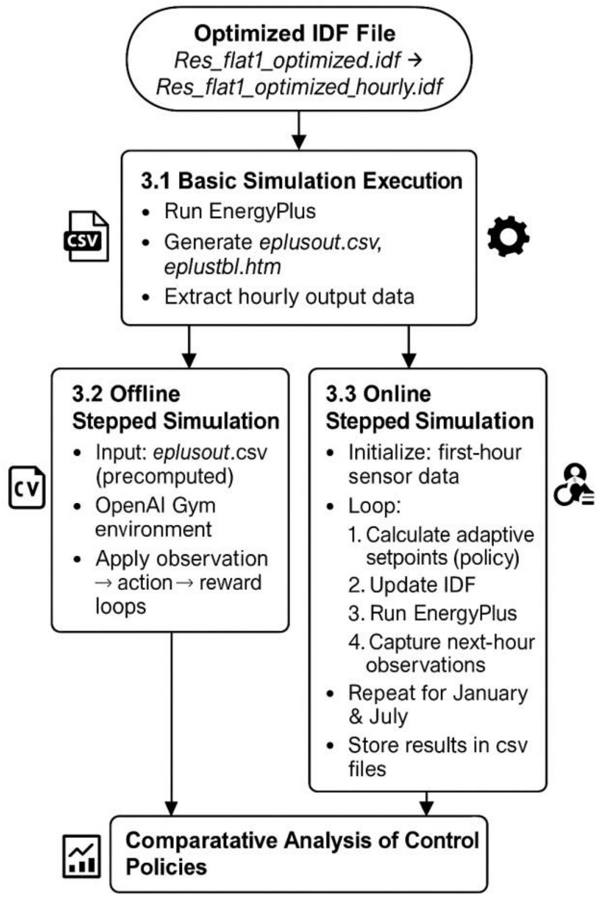

Optimized IDF File
Res_flat1_optimized.idf → Res_flat1_optimized_hourly.idf
3.1 Basic Simulation Execution
• Run EnergyPlus
• Generate eplusout.csv, eplustbl.htm
• Extract hourly output data
3.2 Offline Stepped Simulation
• Input: eplusout.csv (precomputed)
• OpenAI Gym environment
• Apply observation → action → reward loops
3.3 Online Stepped Simulation
• Initialize: first-hour sensor data
• Loop:
1. Calculate adaptive setpoints (policy)
2. Update IDF
3. Run EnergyPlus
4. Capture next-hour observations
• Repeat for January & July
• Store results in csv files
Comparatative Analysis of Control Policies

Figure 9: Overview of the stepped simulation workflow combining offline and online EnergyPlus executions with control policy evaluation.

The chapter is organized into three main parts:

- **Preparing and Running Basic Simulations** introduces the execution of the optimized IDF under both default (10-minute) and hourly timesteps, analyzing the resulting datasets and output structures.
- **Offline Stepped Simulation Using Gym** describes how offline simulations were set up using the OpenAI Gym framework to validate observation-action-reward loops without requiring real-time EnergyPlus execution.
- **Online Stepped Simulation with Real-Time Control** presents the full integration of control strategies into live simulations, where adaptive decisions dynamically modify the IDF and influence subsequent simulation outcomes.

The chapter concludes with a comparative discussion of different control policies and their impact on building performance, highlighting both the methodological innovations and practical challenges encountered during implementation.

3.1. Preparing and Running Basic Simulations

This section describes the initial execution of EnergyPlus simulations using the optimized building model obtained from the previous design optimization phase. Two key input files were prepared:

- **Res_flat1_optimized.idf**: the full-optimization model with detailed timestep settings (10-minute intervals, i.e., six samples per hour).
- **Res_flat1_optimized_hourly.idf**: a modified version of the optimized IDF where the simulation timestep was adjusted to produce hourly output data, significantly reducing computational load.

To run the simulations, the following command-line call was implemented in Python using the `subprocess` module:

```
subprocess.run([
    "/usr/local/EnergyPlus-9-6-0/energyplus",
    "-r",
    "-w", epw_path,
    "-d", output_dir,
    idf_path
])
```

Here,

- `-r` specifies report file generation,
- `-w` provides the EPW weather file path,
- `-d` defines the output directory,
- and `idf_path` points to the input IDF file.

These basic simulations were **non-interactive**, meaning they ran from start to finish without real-time feedback or external interventions, generating raw output files that document the building's year-round energy and thermal performance.

Output Files and Data Structure

The primary output of these runs is the `eplusout.csv` file, which contains time-resolved simulation data. For the hourly version, the file holds approximately 8760 rows (one per hour over a full year) and over 300 columns representing various sensors, system loads, zone temperatures, and environmental metrics. Example columns include:

- Outdoor air drybulb temperature [°C]
- Heating and cooling energy use [J]
- Zone mean air temperatures [°C]
- Zone relative humidity [%]
- Internal heat gains from equipment, lighting, and occupants

Additionally, the `eplustbl.htm` file provides a summary of key metrics, including tabular reports and visual plots:

- Monthly and annual heating/cooling loads
- Peak load occurrences
- Energy consumption breakdown by end use
- Indoor temperature distributions

These outputs serve as a reference point for validating subsequent simulation experiments and as a foundation for the offline and online stepped simulations detailed in the following sections.

3.2. Offline Stepped Simulation Using Gym

To validate the system's ability to simulate observation-action-reward loops without the computational burden of real-time EnergyPlus execution, this section describes the implementation of an **offline stepped simulation** using the OpenAI Gym framework.

OpenAI Gym is a popular toolkit for developing and testing reinforcement learning (RL) algorithms, offering a standardized interface for environments where an agent interacts by taking actions and receiving observations and rewards. In this project, we customized the Gym environment by integrating precomputed data from the `eplusout.csv` file, enabling us to test control strategies and loop mechanics **without rerunning EnergyPlus at every step**.

Code Structure and Workflow

The main components of the implementation include:

Environment Definition (`MultiZoneEnergyPlusEnv`)

- Defines observation space: key sensor readings (e.g., indoor temperatures, humidity, system loads).
- Defines action space: heating and cooling setpoint adjustments.

Data Input

- Reads hourly simulation data from `eplusout.csv`.
- Treats each time step as an environment state.

Reward Calculation

- Custom reward functions based on performance goals

Result Storage

- Outputs from offline runs are stored in `gym_outputs.csv` for later analysis.

Pseudocode Summary

```
done = False
while not done:
    actions = policy(obs)   # Define control actions
    obs, reward, done, info = env.step(actions)  # Apply action
and get next state
    log_results(obs, reward, actions)  # Save outcomes
```

Example Outputs and Plots

To demonstrate the correctness of the setup, the following outputs were generated:

- Distribution of rewards over time.
- Time series plots of selected actions (e.g., heating/cooling setpoints per zone).
- Observation traces showing system response patterns.

These visualizations provide confidence that the environment logic and reward calculations function as expected before moving to full online simulations.

3.3. Online Stepped Simulation with Real-Time Control

This section presents the implementation of the **online stepped simulation**, where the simulation is not merely a passive process but an **active, real-time control loop**. Here, the system interacts dynamically with the building model: decisions are made at each simulation step based on the most recent sensor data, and those decisions directly influence the next simulation run by modifying the building configuration through the IDF file. This approach enables the study of how control strategies affect building performance in realistic operational conditions.

3.3.1. Detailed Process Description

The online simulation operates on the following iterative workflow:

1. **Initialization**

- The system starts with baseline conditions derived from the first-hour outputs of the precomputed simulation (`Res_flat1_optimized_hourly.idf`).
- All sensors (e.g., indoor air temperatures, relative humidity, heating/cooling loads, outdoor air temperature) are initialized and stored.

2. **Observation Collection**

- At each hourly step, the system reads the latest sensor outputs, which are collected from EnergyPlus simulation results.

3. **Adaptive Setpoint Calculation**

- Using a predefined policy function:

```
adaptive_policy_with_indoor(sensors)
```

- the system determines the most appropriate heating and cooling setpoints for each zone, energy efficiency, and environmental conditions.
- This policy considers factors such as:
  - If outdoor temperatures are extreme (cold <10°C or hot >28°C), increase heating or reduce cooling aggressively.
  - If indoor-outdoor temperature differences are very large, moderate the setpoints to avoid sudden shifts.
  - Under normal conditions, maintain baseline setpoints.

4. **IDF File Modification**

- The updated setpoints are written into the corresponding schedule fields within the IDF file using:

```
update_setpoints(self, setpoints)
```

- For simplicity, the implementation overwrites the entire schedule with the calculated setpoint, assuming no complex hourly schedules (a simplification made for faster prototyping).

5. **EnergyPlus Execution**

- The system executes EnergyPlus again using:

```
subprocess.run([...])
```

- This generates a new output dataset (eplusout.csv) containing the next hour's sensor readings.

6. **Data Storage and Logging**

- For each hour, the system logs:
    - Sensor data (inputs)
    - Applied actions (setpoints)
    - Resulting rewards (if calculated)
- All results are saved incrementally into structured CSV files for later analysis.

7. **Iteration**

- The loop repeats for each hour of the selected simulation periods:
    - January (winter)
    - July (summer)
      covering ~744 hours per month and requiring ~4–5 hours of computation per month.

3.3.2. Sensors and Actuators in Online Stepped Simulation

A central component of the online stepped simulation framework is the integration of **sensor data** (observations) and **actuator signals** (control actions). These two elements form the backbone of the system's decision-making loop, enabling adaptive interaction with the building model.

Sensor Inputs

The simulation environment collects a wide range of sensor data at each time step to capture the building's dynamic state. The full list of monitored variables includes:

- **Environmental variables**:
  - Outdoor air drybulb temperature [°C]
  - Diffuse solar radiation rate [W/m^2]
- **System-level variables**:
  - District heating and cooling energy use [J] (hourly and timestep values)
- **Zone-level variables (for each zone 1–6)**:
  - Mean air temperature [°C]
  - Relative humidity [%]
  - Zone air system sensible cooling rate [W]
  - Zone air system sensible heating rate [W]
  - People's sensible heat gains [W]

In total, the simulation monitors over 30 sensor channels, ensuring a rich and detailed view of both environmental conditions and internal thermal dynamics.

Actuators (Control Variables)

The system controls the building via **setpoints**, which act as high-level actuator signals. Specifically, for each of the six zones, two key setpoints are manipulated:

- Heating setpoint (°C)
- Cooling setpoint (°C)

These setpoints are applied through modifications to the **Schedule:Compact** fields in the IDF file, effectively adjusting the target temperature ranges for the HVAC system at each time step. The control implementation simplifies the scheduling by directly overwriting all time slots with the computed setpoints, assuming no complex temporal variations.

Importance of the Sensor-Actuator Loop

The closed-loop interaction between sensors and actuators is essential for:

- **Dynamic adaptation**: Adjusting HVAC operation in response to real-time conditions.
- **Robustness testing**: Exploring how the system responds to varying external and internal loads.

By systematically logging the sensor inputs, calculated actions, and resulting system performance, the framework provides a detailed dataset for analyzing the effectiveness of different control policies and identifying opportunities for improvement.

3.3.3. Importance and Challenges

This real-time simulation loop is crucial because it allows the exploration of **feedback-based control strategies**, where decisions actively shape system performance. Compared to offline or purely predictive simulations, online stepped simulations better capture:

- Dynamic system responses
- Delays and lags in thermal behavior
- The compounding effects of control decisions over time

However, this approach introduces significant **computational challenges**:

- High execution time: each hour requires a full EnergyPlus run.
- File management complexity: hundreds of intermediate files must be generated, read, and updated correctly.
- Risk of error accumulation: small inaccuracies in policy implementation or file updates can propagate over many simulation steps.

Data Outputs and Visualizations

The final dataset produced from the online simulations includes:

- Hourly indoor and outdoor environmental conditions
- Heating and cooling setpoint histories per zone
- System-level energy consumption patterns (heating, cooling, electricity)
- Calculated performance metrics

These outputs are visualized using:

- Time-series plots of setpoints versus zone temperatures
- Cumulative energy consumption charts comparing winter and summer performance
- Heatmaps showing zone-level performance variations over time

Such visual analyses help to:

- Evaluate how effective the adaptive policy is
- Identify periods of underperformance or instability.
- Provide insights for refining the control strategies further.

3.4. Analysis of Adaptive Setpoint Control Strategies

This report presents an analysis of temperature control strategies applied to Zone 1 of a residential building across different simulation scenarios. The goal is to evaluate how various adaptive setpoint policies influence indoor thermal conditions and energy performance during winter (January) and summer (July) periods.

3.4.1. January – Adaptive Policy

In the first scenario, a simple adaptive control policy is used, adjusting heating and cooling setpoints based solely on outdoor temperature conditions. The following plot illustrates the zone's indoor temperature, outdoor temperature, and the applied setpoints over time.

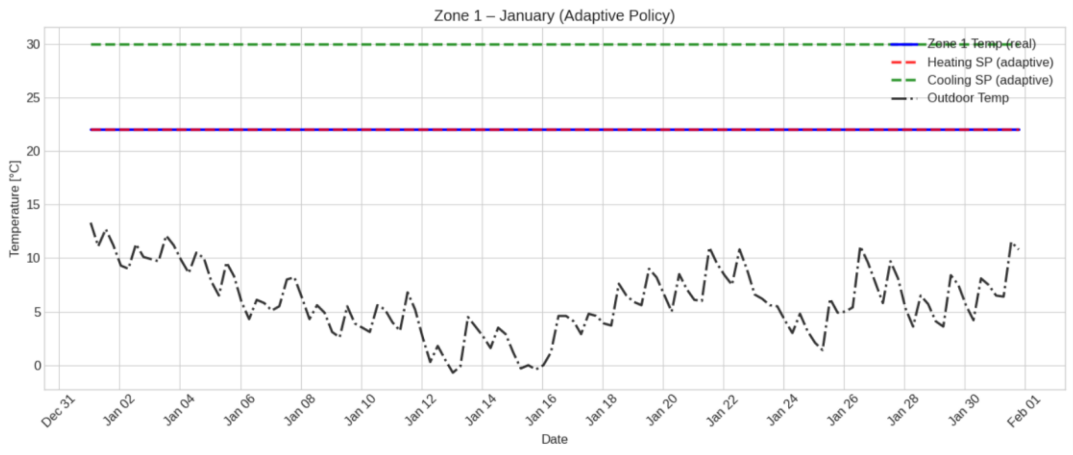


Figure 10: January – Adaptive Policy

The heating setpoint remained at either 18°C, 20°C, or 22°C depending on outdoor conditions, while the cooling setpoint varied between 24°C and 30°C. Indoor temperatures closely followed the setpoints with minimal deviation, demonstrating effective control even without indoor feedback.

3.4.2. January – Adaptive Indoor Policy

This scenario implements a more advanced strategy where both indoor and outdoor temperatures are considered in setpoint calculations. The system dynamically reacts to conditions such as extreme weather or large indoor-outdoor differences. Below is the resulting temperature profile.

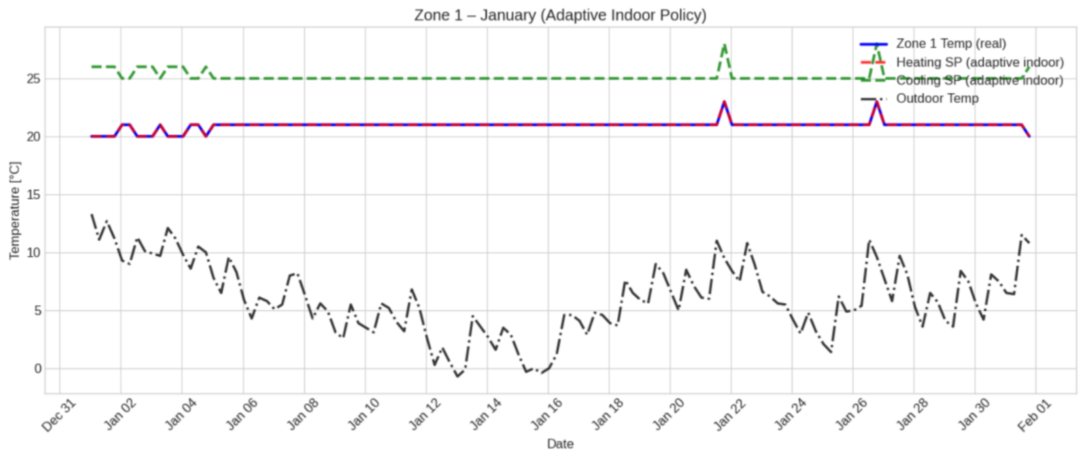


Figure 11: January – Adaptive Indoor Policy

As a result, temperature fluctuations within the zone are smoother, and setpoints are updated more frequently to avoid overheating or overcooling.

3.4.3. July – Adaptive Indoor Policy

The final scenario focuses on the summer season, using the same adaptive indoor policy as in January. This period emphasizes cooling control as outdoor temperatures are significantly higher.

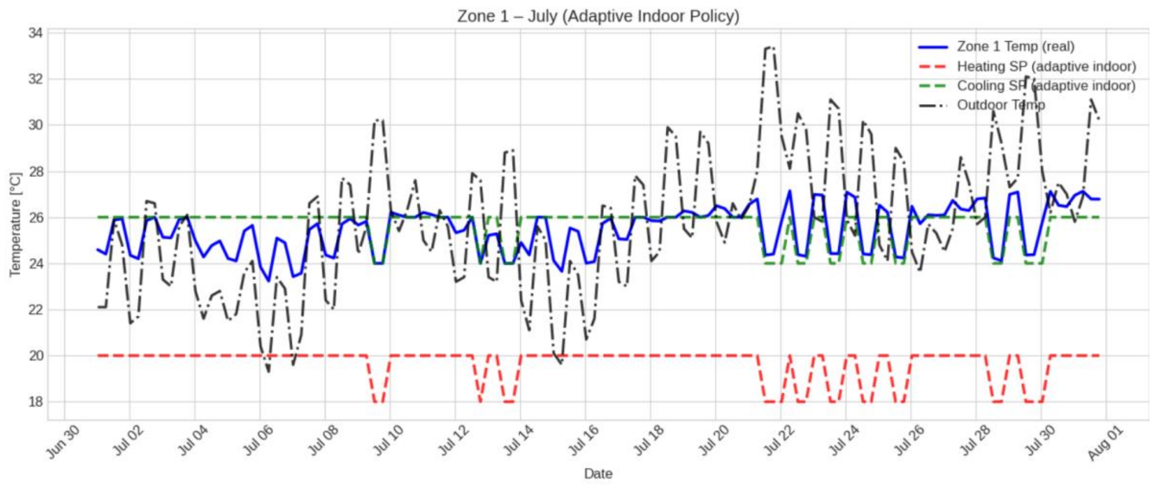


Figure 12 July – Adaptive Indoor Policy

The cooling setpoint remains more actively engaged during July, with several adjustments observed throughout the month. Indoor temperature consistently stays below the setpoint, showing that the control strategy successfully prevents overheating while minimizing energy use.

3.4.4. Comparative Analysis

Across all scenarios, indoor temperatures effectively track the setpoints defined by each control policy. The simpler adaptive policy (based on outdoor temperature only) performs adequately, but the adaptive indoor strategy offers greater precision and comfort. Particularly during summer, real-time indoor feedback proves crucial for maintaining stability under thermal stress.

These findings suggest that incorporating both indoor and outdoor environmental data leads to a more responsive and efficient building management system. Future enhancements could explore learning-based or occupancy-aware policies.

Chapter 4: Building Management System (BMS)

4.1 Chapter Introduction

This chapter presents the design and implementation of a Building Management System (BMS) that connects simulated data streams to real-time monitoring, storage, visualization, and analytical systems. As the final component of the project, the BMS integrates simulation, data transfer, storage, dashboard visualization, and energy signature analysis into a cohesive, continuous workflow.

4.2 System Architecture

The overall BMS architecture consists of the following components:

- **Publisher**: A lightweight Docker service that reads simulation data from CSV files and publishes them (simulated as one-hour values every minute) to an MQTT broker.
- **Mosquitto MQTT Broker**: Acts as the intermediary between the publishers and receivers (such as OpenHAB).
- **OpenHAB**: The smart building management system that receives MQTT messages using configured *.things* and *.items* files, prepares them for storage, and makes them accessible for display.
- **InfluxDB**: The time-series database used to store incoming BMS data, enabling time-based queries and analytics.
- **Grafana**: A dashboard and analytics tool that connects directly to InfluxDB to visualize and monitor the BMS data.

Figure 13 illustrates how the designed data flow system integrates a Python-based publisher that streams time-stamped simulation data to the MQTT broker. OpenHAB, using its configured *.things* and *.items* layers, receives and routes the data into InfluxDB via defined persistence strategies. Stored time-series data is then visualized in Grafana, creating a continuous real-time feedback loop for monitoring and analysis.

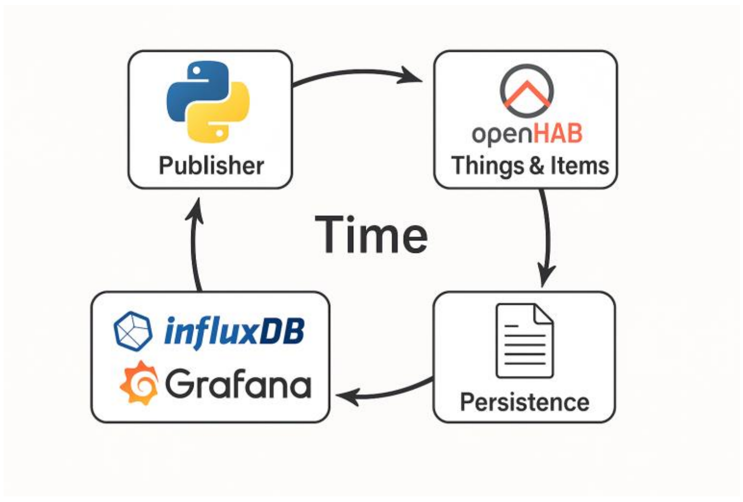


Figure 13: *Real-time data flows from the Python publisher through MQTT and OpenHAB into InfluxDB, with Grafana visualizing the live system loop.*

4.3 System Design and Configuration

4.3.1 Publisher

To simulate real-time data streaming, a Docker-based publisher service was designed using Python (with `pandas` and `paho-mqtt`). It reads the CSV files, applies precise topic mappings between columns and MQTT topics, and publishes the values at fixed intervals (one row per minute).
To ensure that dependent services are ready, a `wait-for-it.sh` script was used, making the publisher wait until OpenHAB and Mosquitto are fully up before starting.

4.3.2 Mosquitto MQTT Broker

A simple Docker service was deployed to manage the messaging between the publisher and OpenHAB. All topic configurations were precisely set in OpenHAB to match the MQTT setup.

4.3.3 OpenHAB

OpenHAB was configured to:

- Use `mqtt.things` and `default.items` files to map each sensor and actuator topic correctly.

- Apply the *`everyChange`* storage strategy in the *`influxdb.persist`* file, ensuring only changed values were written to the database, minimizing redundant writes.
- Connect to InfluxDB using the *`influxdb.cfg`* configuration.

4.3.4 InfluxDB and Grafana

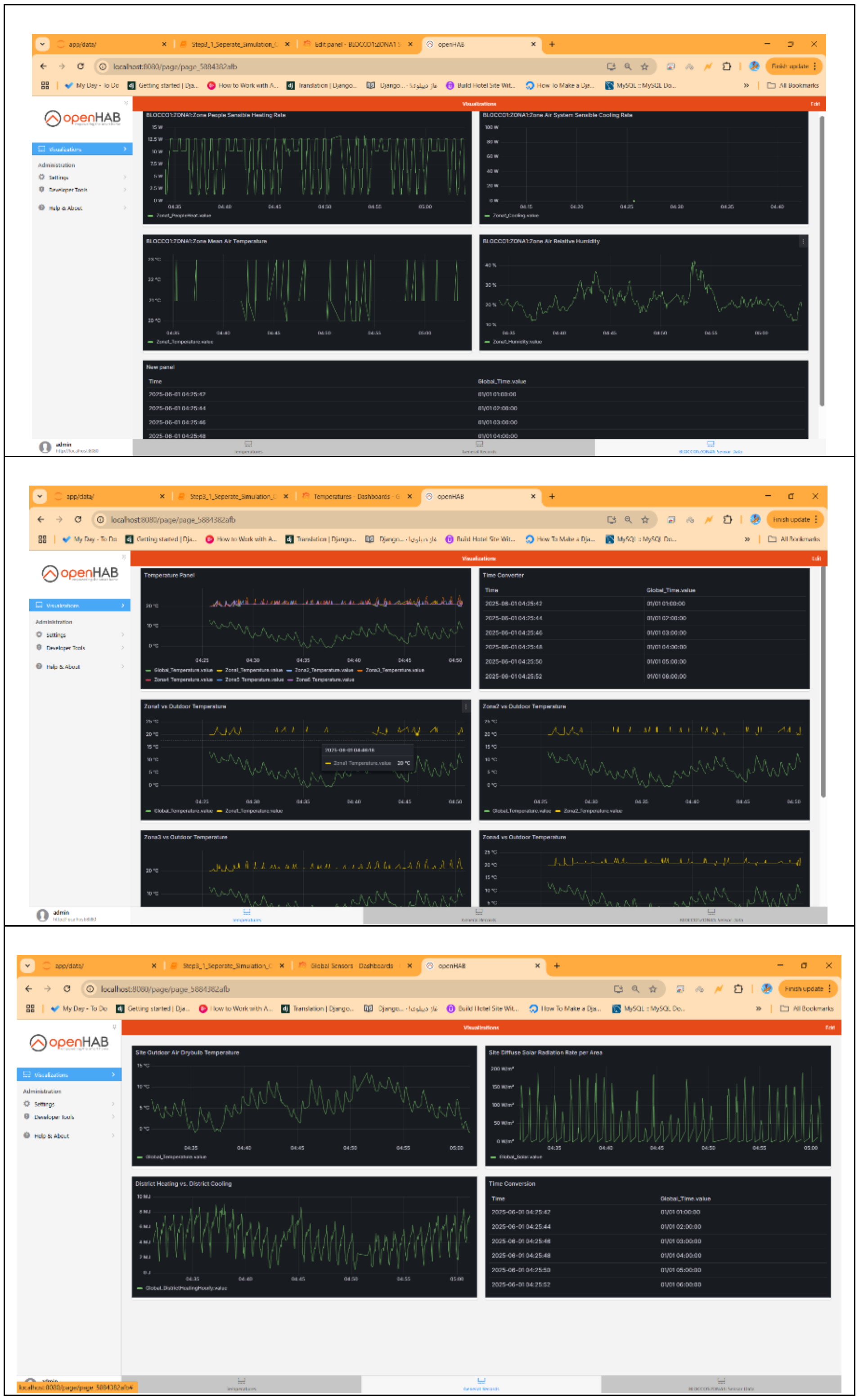
InfluxDB was selected as the main time-series storage engine, initialized with the required database and user credentials.
Grafana was connected to InfluxDB to display real-time dashboards and visual analytics, offering insights into system behavior, such as temperature variations, energy usage, and setpoint dynamics.

4.4 Challenges and Solutions

- **Repeated Data Transmission**: It was observed that the publisher was looping over the January dataset repeatedly. This behavior was resolved by introducing proper loop controls or termination conditions in the Python script.
- **Redundant Storage in InfluxDB**: Changing the storage strategy from everyMinute to everyChange reduced unnecessary database growth, ensuring only meaningful data points were retained.
- **Ensuring Service Readiness**: The use of wait-for-it.sh ensured that dependent services (like OpenHAB and Mosquitto) were fully ready before the publisher began transmitting data.

4.5 System Performance and Outcomes

Once the BMS was fully deployed, the simulated data was successfully streamed to Mosquitto, received by OpenHAB, stored in InfluxDB, and visualized on Grafana dashboards, both within the Grafana service itself and as embedded panels inside the OpenHAB interface. Figure 14 and Figure 15 showcase examples of these visualizations, displaying real-time tracking of indoor and outdoor temperatures, energy flows, and setpoint dynamics across multiple zones. Together, these dashboards provide clear, actionable insights for system monitoring and performance analysis, demonstrating the full integration and responsiveness of the BMS pipeline.


openHAB
Visualizations
Administration
Settings
Developer Tools
Help & About
BLOCCO1:ZONA1:Zone People Sensible Heating Rate
BLOCCO1:ZONA1:Zone Air System Sensible Cooling Rate
BLOCCO1:ZONA1:Zone Mean Air Temperature
BLOCCO1:ZONA1:Zone Air Relative Humidity
Temperature Panel
Zone1 vs Outdoor Temperature
Zone2 vs Outdoor Temperature
Zone3 vs Outdoor Temperature
Zone4 vs Outdoor Temperature
Site Outdoor Air Drybulb Temperature
Site Diffuse Solar Radiation Rate per Area
District Heating vs. District Cooling
Zone1_Temperature.value
Zone1_Humidity.value
Global_Temperature.value
Global_Solar.value
Global_DistrictHeatingHourly.value
Finish update
All Bookmarks

New panel

Time	Global_Time.value
2025-06-01 04:25:42	01/01 01:00:00
2025-06-01 04:25:44	01/01 02:00:00
2025-06-01 04:25:46	01/01 03:00:00
2025-06-01 04:25:48	01/01 04:00:00

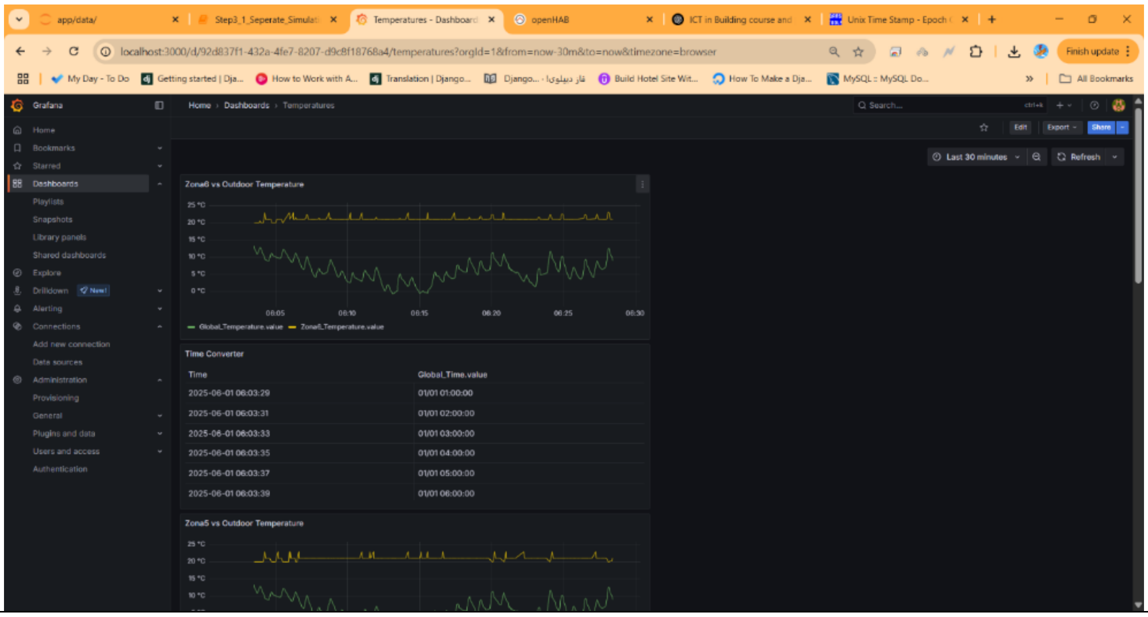
Time Converter

Time	Global_Time.value
2025-06-01 04:25:42	01/01 01:00:00
2025-06-01 04:25:44	01/01 02:00:00
2025-06-01 04:25:46	01/01 03:00:00
2025-06-01 04:25:48	01/01 04:00:00
2025-06-01 04:25:50	01/01 05:00:00
2025-06-01 04:25:52	01/01 06:00:00

Time Conversion

Time	Global_Time.value
2025-06-01 04:25:42	01/01 01:00:00
2025-06-01 04:25:44	01/01 02:00:00
2025-06-01 04:25:46	01/01 03:00:00
2025-06-01 04:25:48	01/01 04:00:00
2025-06-01 04:25:50	01/01 05:00:00
2025-06-01 04:25:52	01/01 06:00:00

Figure 14: embedded Grafana *dashboard views in OpenHAB displaying real-time visualization of indoor zone metrics, global temperature and humidity, and system-level energy and environmental variables streamed through the integrated BMS.*


Grafana
Home › Dashboards › Temperatures
Last 30 minutes
Refresh
Zona6 vs Outdoor Temperature
Global_Temperature.value
Zona6_Temperature.value
Zona5 vs Outdoor Temperature

Time Converter

Time	Global_Time.value
2025-06-01 06:03:29	01/01 01:00:00
2025-06-01 06:03:31	01/01 02:00:00
2025-06-01 06:03:33	01/01 03:00:00
2025-06-01 06:03:35	01/01 04:00:00
2025-06-01 06:03:37	01/01 05:00:00
2025-06-01 06:03:39	01/01 06:00:00

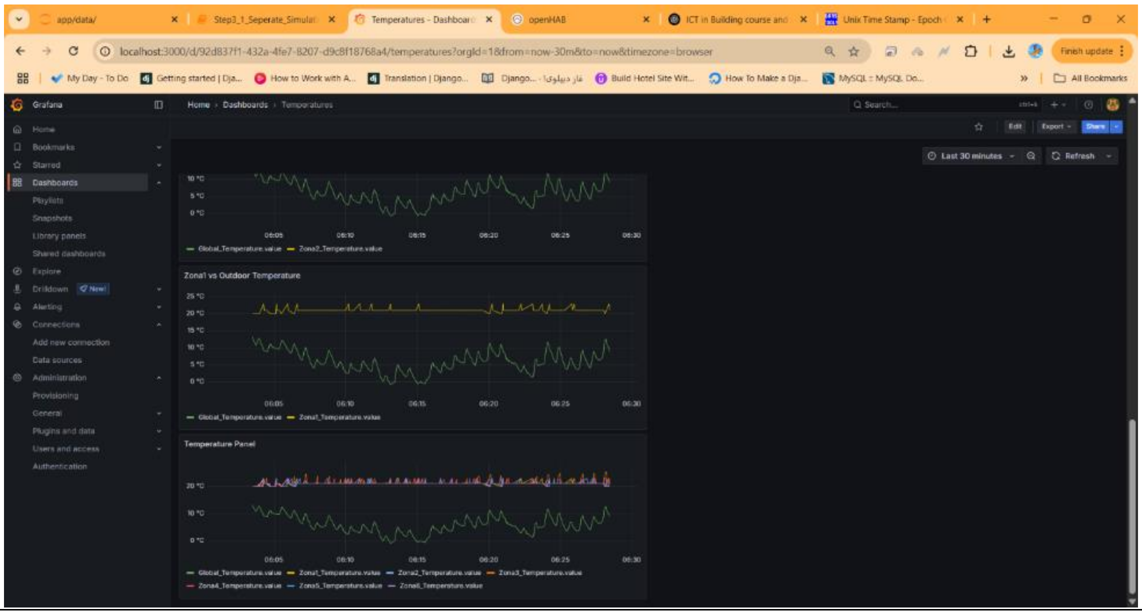
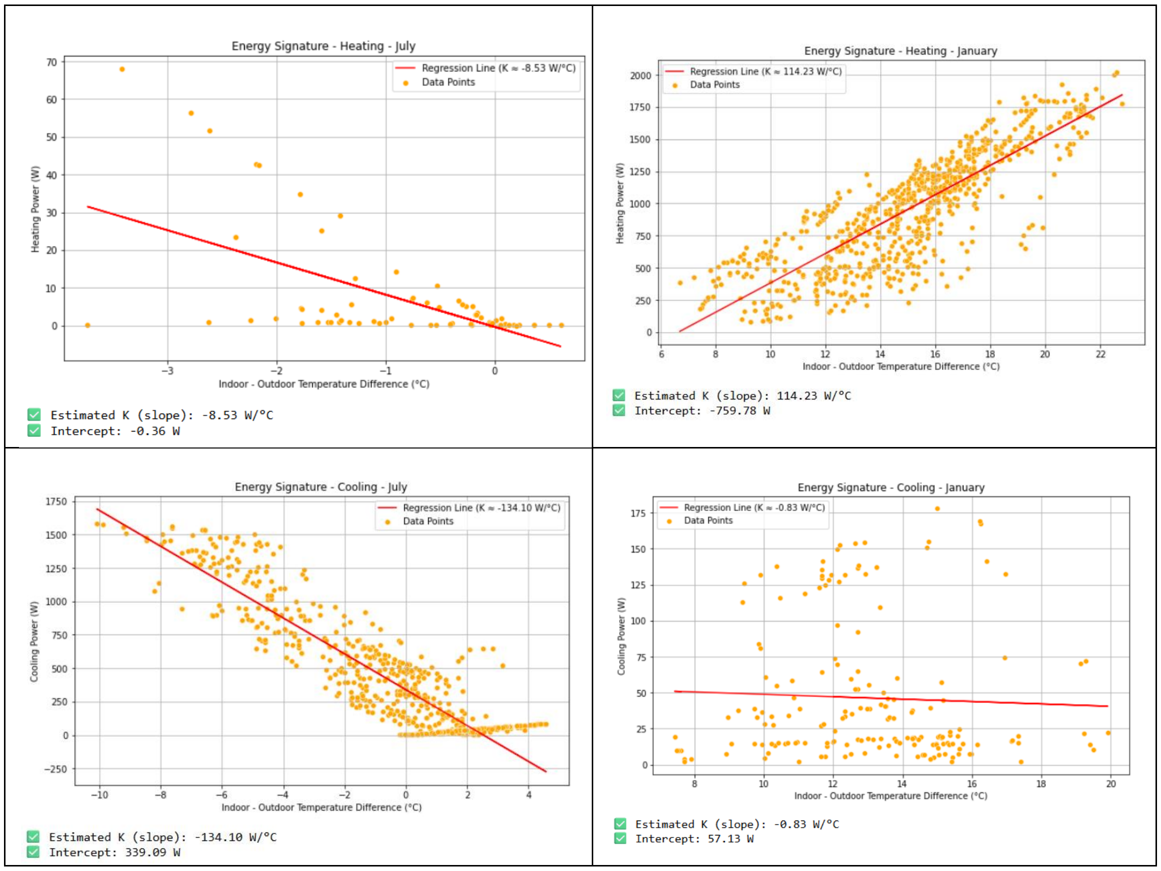


Figure 15: Grafana dashboard visualizations comparing global and zone-level temperature trends over time, showcasing real-time data integration and cross-panel analysis within the BMS.

4.6 Energy Signature

To deepen the evaluation of the system’s performance, Figure 16 presents the results of an energy signature analysis conducted using the time-series data generated and collected through the full simulation-to-database pipeline. This analysis focuses on examining the linear relationship between the indoor–outdoor temperature difference and the corresponding heating or cooling power demands, visualized through scatter plots and quantified using regression models.

For the cooling mode, the first regression yielded a strong negative slope (K ≈ –134.10 W/°C), indicating a clear reduction in cooling demand as the temperature difference between indoors and outdoors decreased. This suggests the system effectively reduces cooling loads under milder conditions. A second cooling signature with a much flatter slope (K ≈ –2.94 W/°C) highlighted zones or periods with minimal cooling sensitivity to temperature differences, possibly reflecting low-activity periods or baseline system loads.



```
✅ Estimated K (slope): -8.53 W/°C
✅ Intercept: -0.36 W
```

```
✅ Estimated K (slope): 114.23 W/°C
✅ Intercept: -759.78 W
```

```
✅ Estimated K (slope): -134.10 W/°C
✅ Intercept: 339.09 W
```

```
✅ Estimated K (slope): -0.83 W/°C
✅ Intercept: 57.13 W
```

Figure 16: Seasonal energy signatures for July and January

In heating mode, the regression lines revealed contrasting behaviors: in one case, the slope was slightly negative (K ≈ –8.53 W/°C), suggesting minimal or negligible heating response within that range; in another, the slope was strongly positive (K ≈ 117.29 W/°C), confirming that as the temperature difference increased, the heating system proportionally ramped up its power output. This latter result aligns well with theoretical expectations of a responsive heating system adjusting to higher thermal loads.

These analyses are critical because they not only provide empirical K-values (thermal conductance estimates) but also help verify whether the simulated system behaves according to its design intent. By comparing these regression-derived signatures across zones and operation modes, the BMS can be fine-tuned to better match real-world conditions or optimized scenarios. Furthermore, embedding such analytical processes into the real-time data loop ensures that the system is not just passively monitoring but actively diagnosing and enabling evidence-based control adjustments.

4.7 Summary

This chapter concluded the project's implementation phase by building a fully integrated BMS capable of simulation, monitoring, storage, visualization, and analytical evaluation. All components were orchestrated using Docker, enabling seamless and reliable data flow from simulation to database. In addition to setting up the data infrastructure, an energy signature analysis was performed, providing key insights into the building's thermal behavior across seasons. With the applied optimizations, the system achieved accurate, non-redundant, and stable performance, supporting both real-time monitoring and in-depth energy diagnostics.

Refernces

[1]: M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979, doi: 10.1080/00401706.1979.10489755.

[2]: S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *arXiv preprint arXiv:1503.06462*, 2015. [Online]. Available: https://arxiv.org/abs/1503.06462

[3]: K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.

[4]: J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95 - Int. Conf. Neural Networks*, Perth, Australia, 1995, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[5]: J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012. [Online]. Available: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

[6]: C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems*, vol. 8, pp. 514–520, 1996. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1995/file/1048-gaussian-processes-for-regression.pdf

[7]: J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011, doi:10.1162/EVCO_a_00009.