POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Master's Degree Thesis

Surrogate modelling and Optimisation in EnergyPlus environments for Smart Buildings

Supervisors

Prof. Giacomo CHIESA

Dott. Paolo GRASSO

Candidate

Paolo CARRISI

Dicembre 2022

Abstract

The work of this thesis is linked to the design and management of smart buildings. A smart building is one that employs technology to make efficient and economical use of resources while also providing residents with a safe and comfortable environment. Smart buildings may utilise a variety of existing technologies and are planned or modified in such a manner that future technological improvements may be integrated. The specific goal of this thesis is to add functionality to the PREDYCE software (Python semi-Realtime Energy DYnamics and Climate Evaluation), which is a Python library developed inside the E-DYCE and PRELUDE projects to work as an EnergyPlus simulation platform, allowing automatic editing of IDF files (building models) and KPIs computation on both simulation results and monitored data. The added functions are optimization methods that use genetic algorithms (e.g., NSGA2) to solve multi-objective problems, as well as "black-box" simulation methods that use the design and training of artificial neural networks (ANN) to simulate or predict the KPIs of a building, using the building's structural parameters or external climate data as input. These functions have been designed to be helpful during the design phase of a building, but also as a tool in the context of smart buildings, useful in terms of saving consumption and comfort management, since the implementation of artificial neural networks allows for a large number of simulations to be performed almost in "real-time."

Acknowledgements

Table of Contents

List of Tables VI

List of Figures VII

Acronyms X

1 Introduction 1
- 1.1 Problem Statement . . . 1
- 1.2 Building Managment System . . . 3
  - 1.2.1 Building Energy Model . . . 4
- 1.3 Mathematical models for complex systems . . . 5
  - 1.3.1 EnergyPlus as white-box model . . . 7
- 1.4 Objectives . . . 8
- 1.5 Thesis Structure . . . 9

2 Sate-of-Arts 10
- 2.1 EnergyPlus Software . . . 10
  - 2.1.1 DesignBuilder . . . 10
  - 2.1.2 OpenStudio . . . 11
  - 2.1.3 BESOS . . . 12
- 2.2 PREDYCE . . . 12
  - 2.2.1 Sensitivity Analysis with PREDYCE . . . 14

3 Mathematical Methods 15
- 3.1 Multi-objectives optimization . . . 15
  - 3.1.1 NSGA-II . . . 16
  - 3.1.2 NSGA-III . . . 19
- 3.2 Surrogate model . . . 20
- 3.3 Artificial Neural Network . . . 22
  - 3.3.1 Multi-Layer Perceptron . . . 22
  - 3.3.2 Mathematical formulation . . . 24

3.4 Python Libraries 25
3.4.1 Pymoo 25
3.4.2 Scikit-Learn 27

4 Methodology 28
4.1 Optimization 28
4.2 Optimization with Surrogate Model 31
4.3 Surrogate model with metereological data 33

5 Results 35
5.1 Example of Optimization scenario - NSGA2 40
5.1.1 Results discussion 45
5.2 Example of Optimization scenario - NSGA3 46
5.2.1 Results discussion 47
5.3 Example of Optimization with Surrogate Model 47
5.3.1 Results discussion 55
5.4 Example of Surrogate Model with meteorological data 56
5.4.1 Results discussion 66

6 Conclusions 67

Bibliography 68

List of Tables

5.1 Range parameters 41
5.2 Optimum solutions (NSGA2) 44
5.3 Optimum parameters (NSGA2) 45
5.4 Initial database configuration 48
5.5 Prediction errors 53
5.6 Range parameters (Surrogate optimization) 54
5.7 Errors on test-Database 57
5.8 Heatmaps legend 57

List of Figures

1.1 World: Primary energy consumption. Source: U.S. Energy Information Administration, International Energy Outlook 2019 [1] 2
1.2 Energy Consumption by sector from [2] 3
1.3 Building Management System from [3] 4
1.4 Mathematical models for complex systems from [4] 6
1.5 EnergyPlus process from [6] 8

2.1 DesignBuilder interface from [10] 11
2.2 OpenStudio interface from [14] 12
2.3 PREDYCE architecture from [17] 13

3.1 Multi-objective Optimization 16
3.2 Non-dominated sorting method from [23] 17
3.3 Crowding distance from [24] 18
3.4 Pseudocode of NSGA-II (adapted from (Coello et al. 2006) 19
3.5 Reference Directions from [27] 20
3.6 Surrogate Modeling from [28] 21
3.7 Multi-layer perceptron network 23

4.1 Input/output sequence of a basic PREDYCE Optimization from [17] 29
4.2 Input JSON file 29
4.3 Optimisation range parameters 30
4.4 Population example 31
4.5 eplusout.csv example 33

5.1 Render view 35
5.2 Building plant 36
5.3 Interior render view 37
5.4 External walls construction layers 38
5.5 Internal walls construction layers 39
5.6 Windows construction layers 40
5.7 Population example 43

5.8 Pareto Front (NSGA2) 44
5.9 Pareto Front (NSGA3) 47
5.10 EnergyPlus results of Cooling demand 48
5.11 EnergyPlus results of Heating demand 49
5.12 Regression Line for Cooling 50
5.13 Regression Line for Heating 51
5.14 Residuals plot for Cooling 52
5.15 Residuals plot for Heating 53
5.16 Pareto Front with Surrogate model 54
5.17 Optimization results with Surrogate model (Cooling demand) . . . 55
5.18 Optimization results with Surrogate model (Heating demand) . . . 55
5.19 Correlation matrix - Aalborg 58
5.20 Correlation matrix - Athens 59
5.21 Correlation matrix - Torre Pellice 60
5.22 Residuals plot Aalborg-PMV 61
5.23 Residuals plot Athens-RadiantTemp[°C] 62
5.24 Residuals plot Torre Pellice-Cooling[J] 63
5.25 Regression line Aalborg-PMV 64
5.26 Regression line Athens-RadiantTemp[°C] 65
5.27 Regression line Torre Pellice-Cooling[J] 66

Acronyms

AI

artificial intelligence

Chapter 1

Introduction

1.1 Problem Statement

Because of global climate circumstances, the subject of energy conservation has taken on, and continues to take on, increasing relevance in Western society. As a result, energy production from fossil fuels is becoming increasingly unsustainable, owing to the high costs and low quality of these raw resources. To address this tendency and the ongoing growth in global energy demands (Figure 1.1) , it is consequently vital to rely more and more on renewable sources that do not exacerbate our planet's climatic circumstances. Furthermore, to support this move, it is vital to invest in processes of crucial energy conservation, discovering increasingly "smart" solutions on all levels that enhance efficiency by restricting waste to a bare minimum.

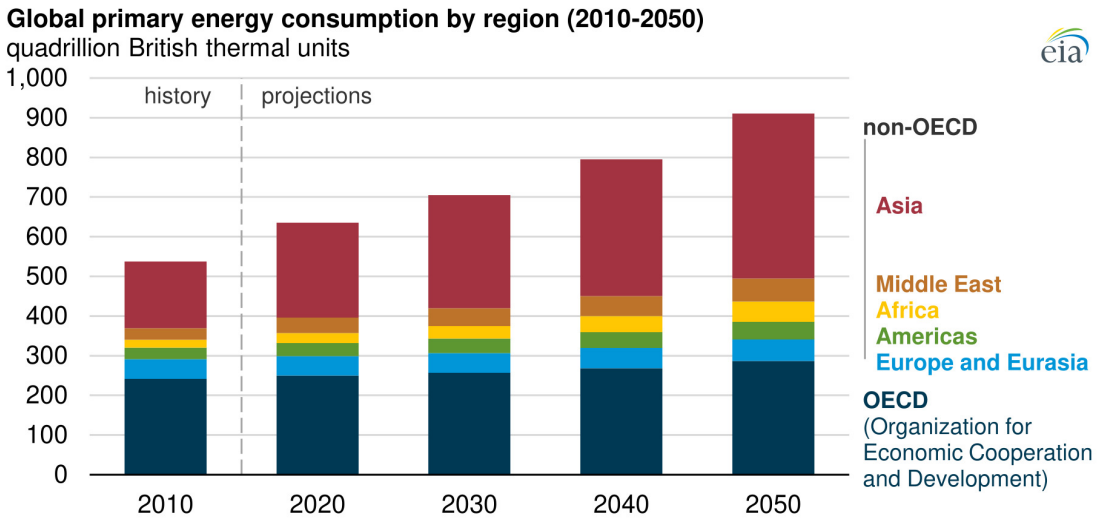


Figure 1.1: World: Primary energy consumption. Source: U.S. Energy Information Administration, International Energy Outlook 2019 [1]

The construction industry consumes a major share of total energy on a worldwide scale (Figure 1.2), and for this reason energy efficiency programs attempt to reduce the usage of thermal or electrical energy in the building sector. Based on this premise, European Union member states are already implementing policy review measures, as well as the deployment of new regulatory instruments and techniques capable of integrating design with a detail attention to the energy component of the structure. As a result, we seek to achieve a specified building process through the application of legal criteria and the concrete potential of achieving optimal performance in terms of energy efficiency and living comfort.

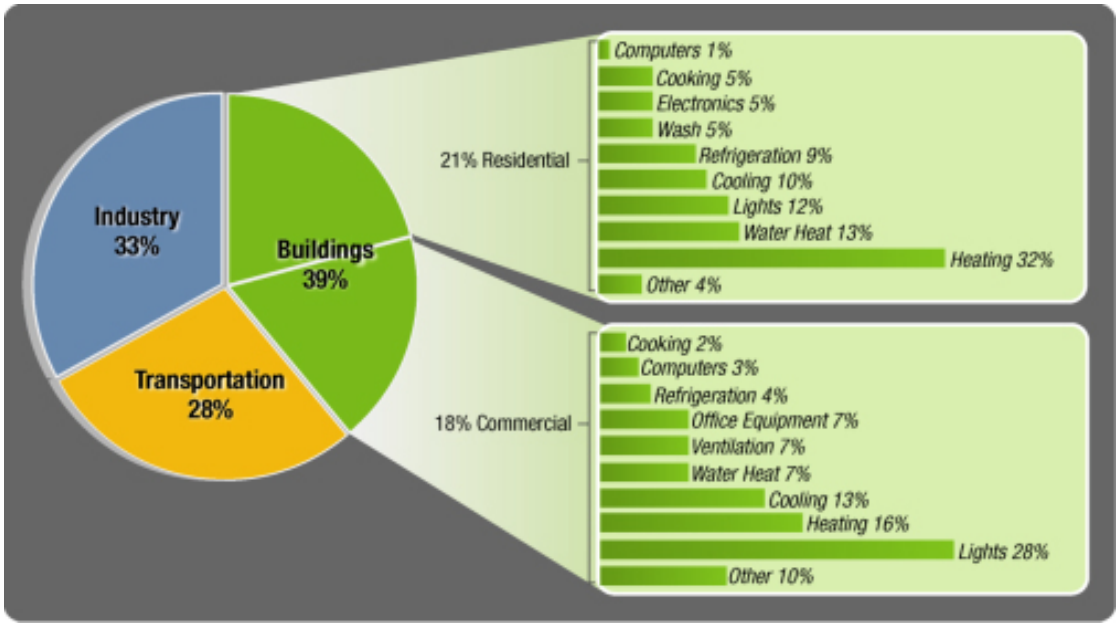


Figure 1.2: Energy Consumption by sector from [2]

1.2 Building Managment System

BMS (Building Managment System) is a computer based control system used to monitor and control mechanical and electrical equipment in a building, or a part of a building. Based on both software and hardware entities, the main goal of this system is to manage heating and cooling of the building, mechanical and natural ventilation and the air conditioning system (HVAC). BMS is now based on open communications protocols like DeviceNet, SOAP, XML, BAcNet, LOnWorks and are WEB enabled. It can be divided into 3 different levels, FIELD level (with all the sensors, actuators, room controls...), AUTOMATION level (including LON to BacNet router/gateway and automation level controllers) ad the last MANAGMENT level (Web Browser server and work stations). An important aspect of BMS is the IoT technology in the sensors and valves at th field level, that enables smart metering and remote monitoring and control. Data collected from IoT devices can be used for different purposes like control policies development, visualization for facility and energy managers, and also for BEM .

Figure 1.3: Building Management System from [3]

1.2.1 Building Energy Model

Building Energy Model (Figure 1.3) is a versatile, multipurpose tool that is used in new building and retrofit design, green certification, real-time building control. Is a physics-based software simulation of building energy use, that takes as input a description of a building (BIM). BIM modelling is a process involving the generation and management of digital representation of physical and functional characteristics of places. The geometry, building materials, lighting, HVAC, refrigeration, water heating, and renewable generating system configurations, component efficiency, and control methods are all included in a BIM file. It also collects details on how the facility is used, such as occupancy schedules, lighting requirements, plug loads, and temperature settings. Building information models (BIMs) are data files that may be extracted, traded, or networked to assist decisions about a building or other architectural asset. The thermal loads, system reaction to those loads, and associated energy usage, along with related metrics like occupant comfort and energy costs, are calculated by a BEM algorithm using these inputs and local meteorological data. BEM programs do computations on an hourly or shorter basis for a whole year. They also take into consideration system interactions, such as those between heating and cooling and lighting.

Applications for BEM make use of its capacity to respond to inquiries that are difficult to resolve through conventional channels. The following are significant

usage cases:

Applications for BEM make use of its capacity to respond to inquiries that are difficult to resolve through conventional channels. The following are significant usage cases:

1. Design of Buildings: BEM is used by architects to create energy-efficient structures, especially to explain quantitative trade-offs between initial construction costs and ongoing energy expenses. BEM may frequently lower both energy expenditures and initial building costs.

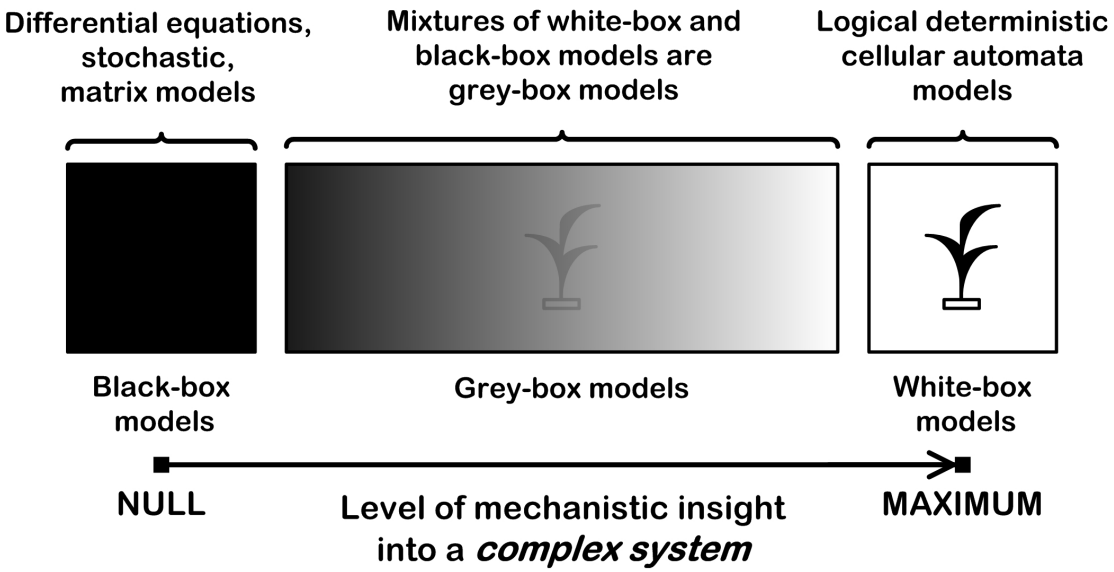
2. Commercial building HVAC systems may be big and sophisticated in terms of design and operation. BEM aids mechanical engineers in creating HVAC systems that effectively satisfy building thermal demands. It aids in the development and evaluation of control techniques for these systems.

3. Building Performance Rating: BEM can be used to evaluate a building's intrinsic performance while accounting for its intended purpose and mode of operation. Processes like code compliance, green certification, and financial incentives are all based on inherent performance rating.

4. Building Stock Study: BEM analysis on prototype models aids in the creation of energy codes and standards and aids in the development of large-scale energy-efficiency initiatives by utilities and municipal governments.

1.3 Mathematical models for complex systems

Which models should be used in an energy consumption prediction framework? There are three different approaches available: white box model, black box model, gray box model (Figure 1.4).



DOI: 10.7717/peerj.948/fig-1

Figure 1.4: Mathematical models for complex systems from [4]

If there is sufficient building information to characterize the set of heat transmission, heat storage, and heat flux, as well as the accompanying physical-significance parameters, then this may be done using a white box model. White-box modeling's degree of difficulty is mostly determined by the precision levels of the known phenomena connected to the building system that will be represented. White-box model parameters have physical relevance (e.g. thermal conductivity of certain materials) Despite the application of fundamental physical principles, mistakes due to random factors that are not represented by the known parameters (such as window openings and air exchange rates in natural ventilation) will always exist.

The settings of the black-box are typically changed automatically. The biggest advantage over white-box devices is this automated calibration modification of the black-box settings. Their inherent connection to the underlying laws of physics is a drawback. When used in challenging circumstances, black box model identification is discovered to be incompatible with physical reality (little building system data). Black-box models are therefore mostly utilized for error detection rather than optimization. Their benefit is the quick, automated detection of thermal energy consumption outputs from buildings. Black-box models can be static or dynamic, linear or nonlinear, exactly as white-box models, depending on the underlying structure of the model.

1.3.1 EnergyPlus as white-box model

Engineers, architects, and researchers utilize the whole building energy modeling tool EnergyPlus[5] to simulate how much energy is used for heating, cooling, ventilation, lighting, plug and process loads, and water use in buildings. EnergyPlus has a number of remarkable characteristics and capabilities, some of which are:

1. Without assuming that the HVAC system system can handle zone loads and with the ability to model unconditioned and under-conditioned spaces, an integrated, simultaneous solution of thermal zone conditions and HVAC system response is possible.

2. Calculations for thermal comfort and condensation are done using a heat balance-based solution to the radiative and convective actions that result in surface temperatures.

3. Sub-hourly, user-definable time steps are used for interactions between thermal zones and the environment, while time steps for interactions with HVAC systems are automatically changed. These enable EnergyPlus to simulate systems with quick dynamics while balancing simulation speed and accuracy.

4. Air flow across zones is taken into consideration via a combined heat and mass transfer model.

5. Controllable window blinds, electrochromic glazings, and layer-by-layer heat balances that estimate the solar energy absorbed by window panes are examples of advanced fenestration models.

6. Calculations of illumination and glare are used to report on visual comfort and to drive lighting settings.

7. HVAC that is component-based and accommodates both common and unusual system designs.

8. Several pre-installed HVAC and lighting control methods, as well as an expandable runtime scripting framework for user-defined control

9. Energy source multipliers are included in all of the standard summary and comprehensive output reports as well as user-defined reports with a choice of time resolution from annual to sub-hourly.

EnergyPlus is a command-line software that receives input and outputs it to text files. It comes with a number of utilities, including IDF-Editor, which allows you to create input files using a simple spreadsheet-like interface, EP-Launch, which allows you to manage input and output files and perform batch simulations, and

EP-Compare, which allows you to compare the results of two or more simulations graphically. EnergyPlus is free, open-source, and cross-platform, developed by Building Technologies Office (BTO) of the United States Department of Energy (DOE).

Figure 1.5: EnergyPlus process from [6]

1.4 Objectives

The primary goals of this thesis are two:

1. to exploit optimisation enablers into EnergyPlus multiple runs
2. to design a tool for the creation of a Surrogate Model after EnergyPlus multiple runs

Concerning Optimisation, the goal is to be able to introduce within the sensitivity analysis of genetic algorithms (NSGA2, NSGA3) capable of selecting the best configurations of a building in terms of consumption and/or comfort as the building construction parameters such as window-to-wall ratio, insulation layer thickness, type of glazing, and so on. In this way, it provides the user with not only a way to evaluate which parameters have the greatest impact on the outcome of building energy simulations, but also the ability to choose which of these configurations may represent a fair trade-off for the various output values that one wants to maximize/minimize.

The surrogate model, on the other hand, is intended to replace traditional simulations such as EnergyPlus with a black-box prediction method such as artificial neural networks, which are significantly less expensive in terms of computational complexity and execution time. This approach's specific goals are as follows:

1. perform sensitivity analysis/optimization of building parameters by utilizing a much larger number of samples while maintaining high accuracy
2. train a network capable of predicting, given a specific building configuration, the consumption and/or comfort score as the external environment's climatic data changes.

Both optimization and surrogate modeling tasks were implemented within the PREDYCE tool, which was developed by the Politecnico di Torino as part of the PRELUDE[7] and E-DYCE [8] project and was discussed in the chapter State-of-Arts that follows. The EU-funded PRELUDE project proposes a proactive optimisation service based on smart technologies. The service provides clear and appropriate feedback and suggests retrofitting operations on a cost-efficient basis. It uses natural ventilation and cooling to reduce the energy consumption of mechanical heating, ventilation and air conditioning (HVAC) systems. Big Data and advanced analytic instruments will encourage demand-side flexibility and moderate the integration into district heating and electricity grids. The innovative service will be demonstrated in a wide range of applications in individual multi-apartment and large-scale residential buildings in cities across Europe. This thesis is part of the PRELUDE project under T9.5 activities (education and citizen science).

1.5 Thesis Structure

Following is a brief explanation of the structure of the topics covered in this thesis. In the first section of *State of Arts*, a brief description of several software and tools that use EnergyPlus for building modeling and functional and energy optimization is provided. With great care, the tool PREDYCE was presented, which will be used in this thesis project as infrastructure to execute the features described in the *Objectives* chapter. Following that, in the *methods* chapter, a simplified proposal of the algorithms used to accomplish both optimization and surrogate modeling tasks is presented, along with a brief description of the Python libraries used to implement these algorithms within the PREDYCE tool. In the chapter *methodology*, three different functions implemented in the PREDYCE tool are used:

1. Multi-objectives optimization as a result of sensitivity analysis
2. Multi-objectives optimization with the assistance of Surrogate model
3. Surrogate modeling using meteorological data

Then, in the *results* section, three practice examples of using the tool's new features are used, commenting on the results and output graphics obtained.

In the final analysis, in the chapter *conclusion*, the potentialities of using these codes within a real-world energy optimization context are illustrated, as are thoughts and ideas for future changes and improvements.

Chapter 2

Sate-of-Arts

2.1 EnergyPlus Software

EnergyPlus software may run performance-driven energy and comfort evaluations and/or optimization activities using graphical interfaces or code tools that enable the integration of a single dynamic energy simulation into a more organized workflow that includes input editing and output analysis. Several graphical interfaces, which are regularly updated and provide a variety of features, have been created to aid professionals in their use of EnergyPlus.

2.1.1 DesignBuilder

DesignBuilder is the most extensive Energy-Plus interface currently available [9]. The program has a simpler CAD interface, guided processes, and more compact Energy-Plus airflow modeling setups. The typical application of DesignBuilder enables a variety of facade envelope alternatives, solar illumination analysis via site location, fluid dynamics modeling, and HVAC equipment sizing. DesignBuilder was created mainly as a tool to assist all phases of the design process. DesignBuilder is intuitive and straightforward to use, however, it does not currently offer all Energy-Plus features. The software's structure is comprised of HVAC systems that give basic and concise descriptions, but do not provide specific information about their components and topology. Another limitation is that Energy-Plus input files cannot be imported, so a 3D geometry model must be made for each energy analysis.

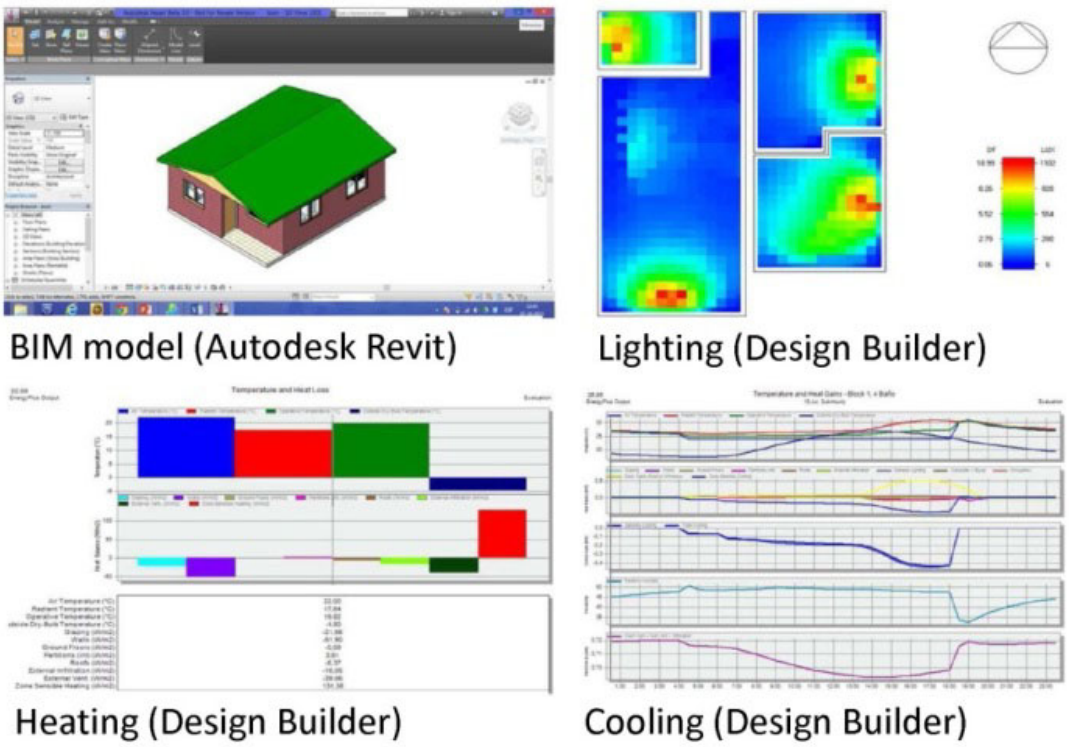


Figure 2.1: DesignBuilder interface from [10]

2.1.2 OpenStudio

OpenStudio [11] is a graphical interface for building energy modeling that uses Energy-Plus and Radiance [12] for lighting analysis. A version of the program is included in a SketchUp plug-in [13], which is highly beneficial for merging architectural and energy design. Modules for viewing and modifying the envelope's constituent components, as well as an interface for simulating interior loads and the air and water-conditioning systems, are included in the application. The Radiance module may be used to integrate energy simulations. The "ParametricAnalysisTool" module gives a variety of energy choices for the given analysis scenario, whilst the "RunManager" module enables you to execute energy simulations using the Energy-Plus engine in parallel. The findings are shown using the "ResultsViewer" module, which provides a basic summary as well as specifics of the energy analysis. There are just a few restrictions while utilizing this program. Those who are skilled with the SketchUp module will have no trouble modeling building geometry. One restriction may be the three-dimensional model's tremendous flexibility, which may induce the user to simulate architectural elements that are insignificant for the aim of energy simulation, increasing modeling time and particularly computational

ones.

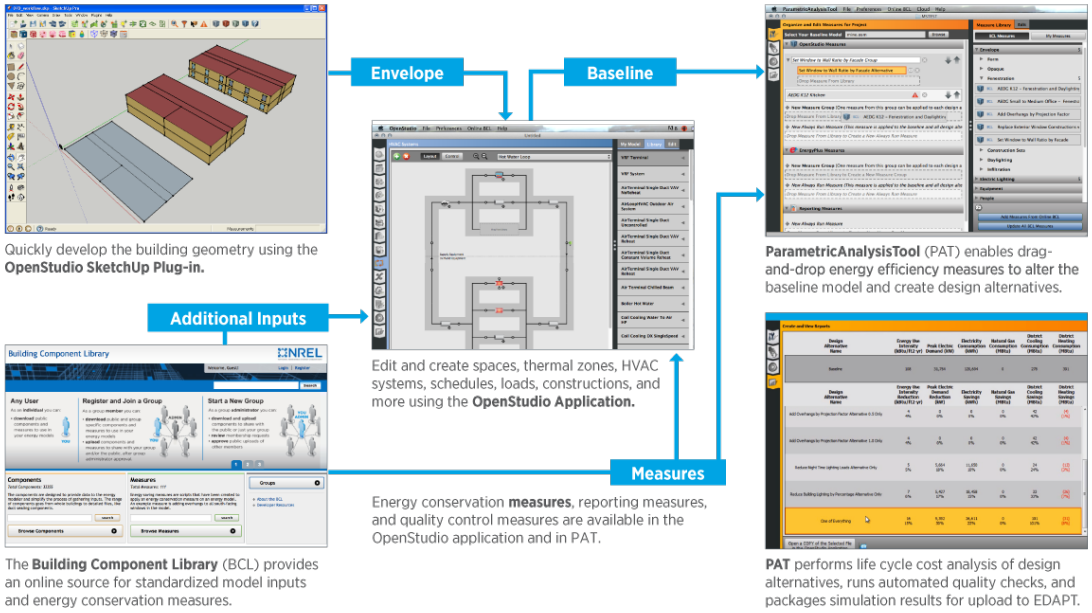


Figure 2.2: OpenStudio interface from [14]

2.1.3 BESOS

To carry out optimizations and other building-related tasks, there are libraries and apps built on the EnergyPlus platform. One option is to bring up BESOS [15], a Python package and a JupyterHub platform that both let users create building optimization tasks using simple coding. It performs parametric IDF editing using the Eppy [16] library, and building optimization is carried out while the parametric analysis is being carried out using genetic or optimization methods (such those found in the Platypus Python library).

2.2 PREDYCE

PREDYCE [17][18] [19](Python Realtime Energy DYnamics and Climate Evaluation) is a new Python-based platform, executable on both personal computers and servers, that can handle parallel simulation runs with an asynchronous multiprocessing strategy, support the integration of additional modules, script existing EnergyPlus input files, support sensitivity analysis, elaborate outputs, and calculate KPIs (such as thermal comfort, indoor air quality, free-running potential, ventilative cooling).

The software tool is built around three key components (Figure 2.3):

1. the EnergyPlus input file editor (*.idf),
2. the KPI (Key Performance Indicators) calculator
3. the runner module, which can execute numerous concurrent simulations automatically.

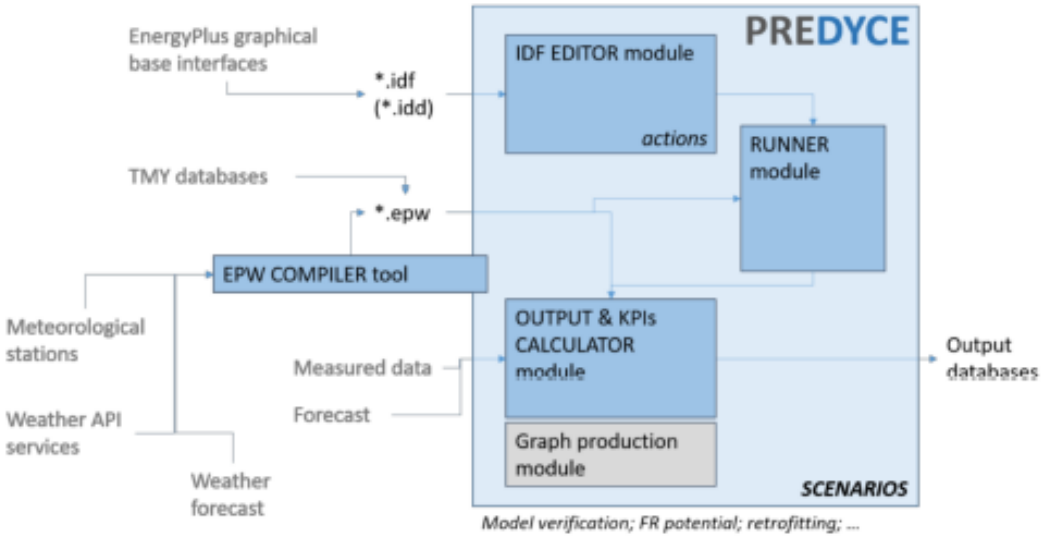


Figure 2.3: PREDYCE architecture from [17]

The weather input file (.epw) and the .idf file itself can both be modified using additional functions. The tool uses an input JSON file to define the actions and KPIs to be performed and calculated, as well as to hold the dictionary-formatted data required to edit the *.idf files (such as the compositions of materials, programs, and construction parts). EnergyPlus input files may be read, edited, and saved using the *.idf editor module. It simplifies the editing process by providing a mechanism to carry out certain complex activities using Python methods that can be quickly called from a custom script. The KPI calculator module may use the EnergyPlus output files to do calculations, analysis, and graph displays. Typically, the studies involve resampling output data and using algorithms to calculate indicators in accordance with European standards. Graphs that can be preserved in subfolder structures can also be included in some computations. Each row represents a simulation, and each column an output (including EnergyPlus base outputs if necessary) or generated KPI. All data are combined into a final CSV file. This module was designed to analyze data received from monitoring

campaigns or simulated by other tools, in addition to EnergyPlus outputs, in order to compute and perhaps compare pertinent KPIs across several datasets. The script called runner module contains a Python class that allows for simultaneous execution of several simulations and analysis. The runner is responsible for setting up a pool of simulations that are subsequently conducted asynchronously by several instances of EnergyPlus on the same system; this pool of simulations can be a straightforward combination of all potential user-described actions in the input JSON file.

2.2.1 Sensitivity Analysis with PREDYCE

Sensitivity analysis [20] is the fundamental and most significant PREDYCE scenario since it serves as the base for all others. By automatically updating the building model in accordance with combinations of all the parameters given in the input JSON file and computing the desired KPIs, it enables the performance of parametric analysis. Other situations could use other logics to choose or combine parameters, but at its core, PREDYCE is capable of running EnergyPlus simulations in parallel and storing the results in an aggregated and lightweight format. Currently, analysis on the best value combinations must be performed after looking through all of the results in the CSV file using scripting, because optimization procedures using optimization algorithms or surrogate modeling have not yet been implemented in order to speed up the scanning of the parameter space and identify the best building configuration. However, the tool's versatility makes it easy to integrate optimization techniques in the future while also employing the numerous Python libraries already in use for this purpose.

Chapter 3

Mathematical Methods

3.1 Multi-objectives optimization

Multi-objectives optimization [21], or MOOP (Multi-Objectives Optimisation), is a procedure that seeks the best solution to a problem with multiple objectives. Because the "best solution" to a genuine problem is typically influenced by several variables, these sorts of challenges have been the topic of extensive research since the mid-1900s. If these variables do not contradict each other (or if there is a hierarchy of priority for the different objectives), the complexity of the issue may be readily reduced by converting a MOOP to a SOP (Single Objective Problem) and seeking solutions in a one-dimensional space. However, when we consider the subject of building energy optimization, we see that this strategy is not always relevant. For example, we can reduce the size of a window to maximize cooling usage during the summer, but this would increase heating consumption during the winter and lower the house's comfort score. It becomes evident, then, that in these sorts of situations, obtaining a univocal answer is not feasible; hence, the right strategy will be to seek a collection of solutions, as compact as possible, in a multi-dimensional space, that can represent "a reasonable compromise" between the many objectives. Formally, the MOOP is defined as follows:

$$\begin{cases} min/MAX & f_m(x)\ for\ m = 1,2,..,M \\ & g_j(x) >= 0\ for\ j = 1,2,...,J \\ & h_k(x) = 0\ for\ k = 1,2,...,K \\ & x_i^{(L)} <= x_i <= x_i^{(U)}\ for\ i = 1,2,...,N \end{cases}$$

Where:

1. M number of goals
2. J number of constraints

3. K number of equality constraints
4. n number of decision variables
5. $x^{(L)} and\ x^{(U)}$ limit of decision variables

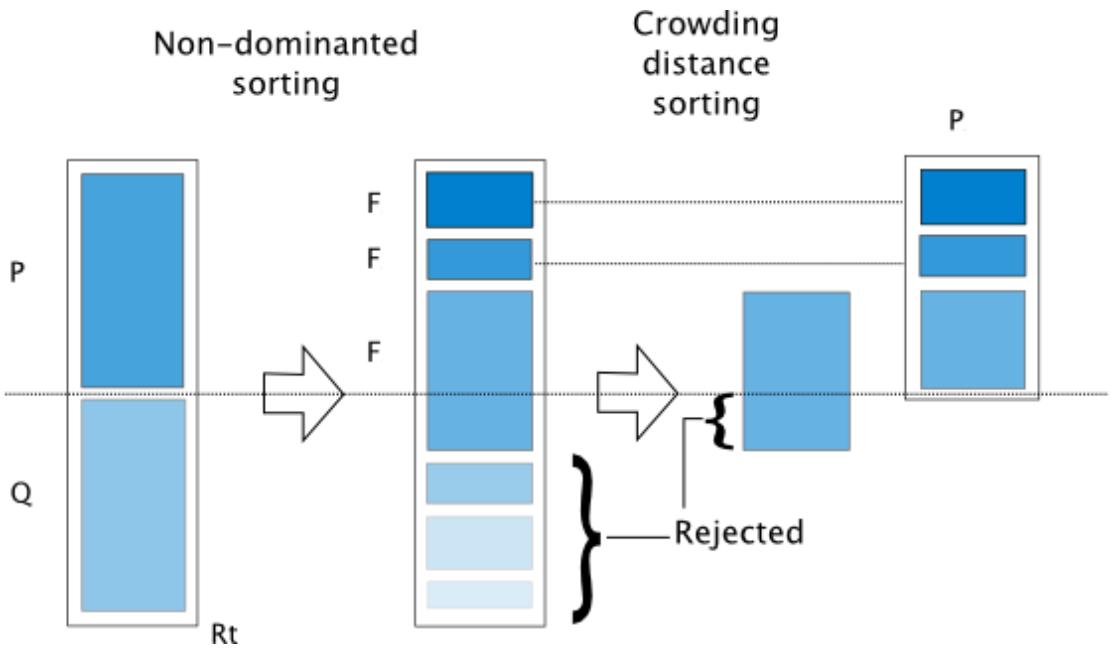


Figure 3.1: Multi-objective Optimization

In a space of multidimensional solutions, it is consequently required to specify a criterion that allows for the unambiguous determination of whether one solution is superior to another. In this context, the idea of Pareto dominance is presented, which states that a solution x_1 dominates ($\triangleright$) a solution x_2 if and only if:

1. $f_i(x^{(1)}) \not\triangleright f_i(x^{(2)})\ for\ i = 1...M$
2. $\exists i \mid f_i(x^{(1)}) \triangleleft f_i(x^{(2)})\ for\ i = 1...M$

3.1.1 NSGA-II

NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm), proposed by Deb in 2000 [22], is one of the most extensively utilized algorithms for this type of issue (especially when addressing 2-objective situations). It is an advanced refinement of NSGA, in which the concepts of solution dominance and "crowding distance" are employed to preserve solution variety. Its strategy is to rank the solutions in a given population by giving a level based on the dominance criterion. Non-dominant

solutions in the entire population will be ranked first. Non-dominant solutions among the whole population, except for rank 1 solutions, will have rank 2, and so on. Figure 3.2 depicts a ranking example in a minimization problem with two objective functions.

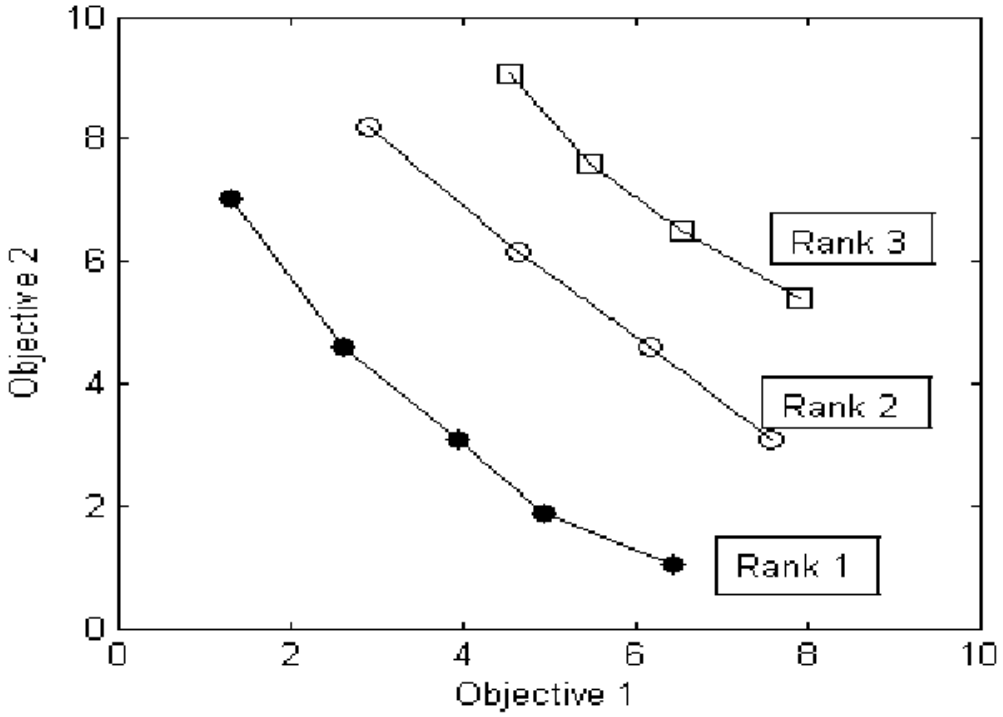


Figure 3.2: Non-dominated sorting method from [23]

After all solutions have been ranked, the crowding distance (Figure 3.3), which measures how isolated a solution is, is taken into account. The crowding distance is applied to each group of solutions with the same rank individually. With respect to each goal function, the solutions are repeatedly ranked in pejorative order. As illustrated in Figure 5, the extreme solutions with the highest and worst fitness in a given goal have a crowding distance value of infinity, while the rest have a value proportional to the distance between the i-th solution and the preceding and next solutions. The more isolated a solution's crowding distance value is, the more viable it is. When comparing two solutions, the rank of the two solutions is first compared; if the rank is the same, the crowding distance is compared. When the rank of two solutions is the same, the element with the greater crowding distance value is chosen.

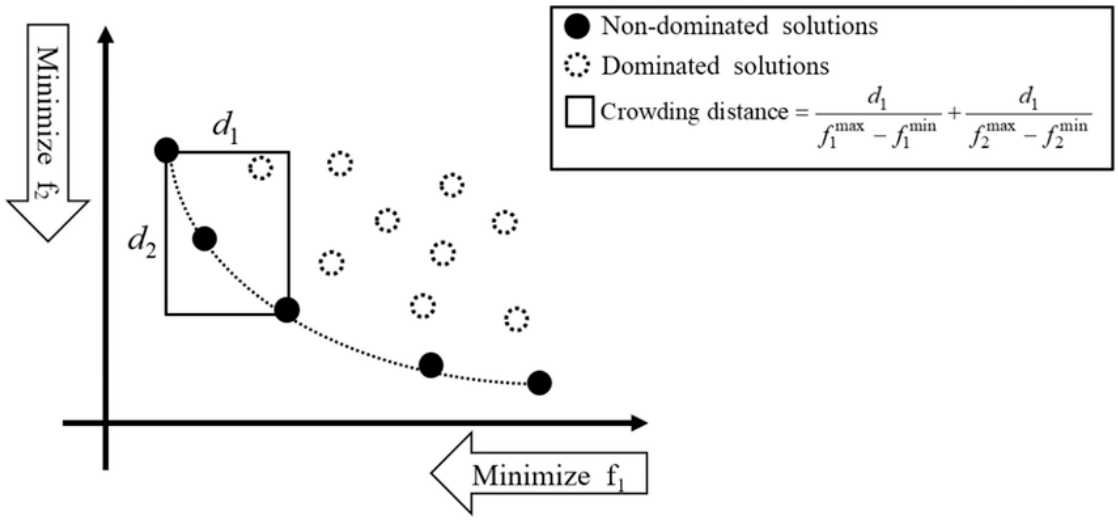


Figure 3.3: Crowding distance from [24]

```
Procedure NSGA-II
   Input: N', g, f_k(X) ▷ N' members evolved g generations to solve f_k(X)
 1 Initialize Population P';
 2 Generate random population - size N';
 3 Evaluate Objectives Values;
 4 Assign Rank (level) based on Pareto - sort;
 5 Generate Child Population;
 6   Binary Tournament Selection;
 7   Recombination and Mutation;
 8 for i = 1 to g do
 9     for each Parent and Child in Population do
10         Assign Rank (level) based on Pareto - sort;
11         Generate sets of nondominated solutions;
12         Determine Crowding distance;
13         Loop (inside) by adding solutions to next generation starting from
           the first front until N' individuals;
14     end
15     Select points on the lower front with high crowding distance;
16     Create next generation;
17       Binary Tournament Selection;
18       Recombination and Mutation;
19 end
```

Figure 3.4: Pseudocode of NSGA-II (adapted from (Coello et al. 2006) [25]

3.1.2 NSGA-III

The Non Dominating Sorting Genetic Algorithm (NSGA-III) is a many-objective algorithm based on the non-dominated sorting for NSGA-II and a new diversity preservation technique based on "reference directions" (Figure 3.5).

The crowding distance employed in NSGA-II is replaced in NSGA-III by a survival mechanism that takes into account the distance of equivalent solutions, i.e. those belonging to the same front, to a set of reference locations. The list of reference points can be supplied by the user, simulating the most interesting search paths, or it can be produced automatically using Das and Denni's approach [26].

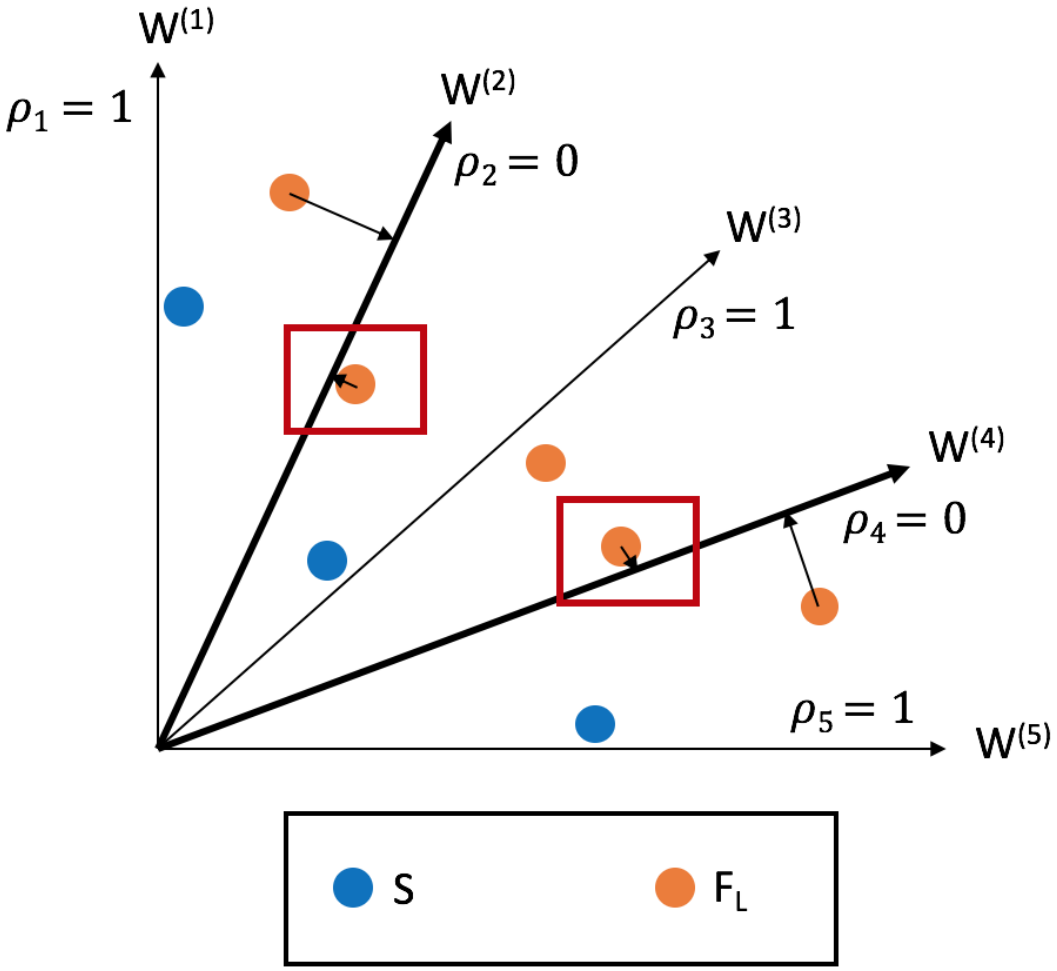


Figure 3.5: Reference Directions from [27]

3.2 Surrogate model

Computer simulations are intended to forecast the behavior of a physical system by solving the mathematical equations related to the physical process. EnergyPlus simulations, for example, may be used to determine a building's performance in terms of energy usage and comfort score.

These simulations are useful because they give engineers with extensive insights into building performance without the need to physically create the building, making them essential for virtual prototyping. Engineers are often required to execute the following tasks in order to promote an effective and reliable design

process:

1. sensitivity analysis, which investigates product performance when design parameters change;

2. optimizations, to determine the best design in order to maximize/minimize the objective function.

Both of these tasks require several simulation runs, with different combinations of design parameters as inputs in each run. However, computer simulations are typically expensive, and analyses that require multiple simulation runs would result in high processing costs, making them impractical in real-world settings.

Surrogate modeling [28] (Figure 3.6) is the process of developing a statistical model (or surrogate model) that accurately approximates the simulation output. Following that, this trained statistical model can be used to replace the original computer simulation in tasks such as optimization and sensitivity analysis.

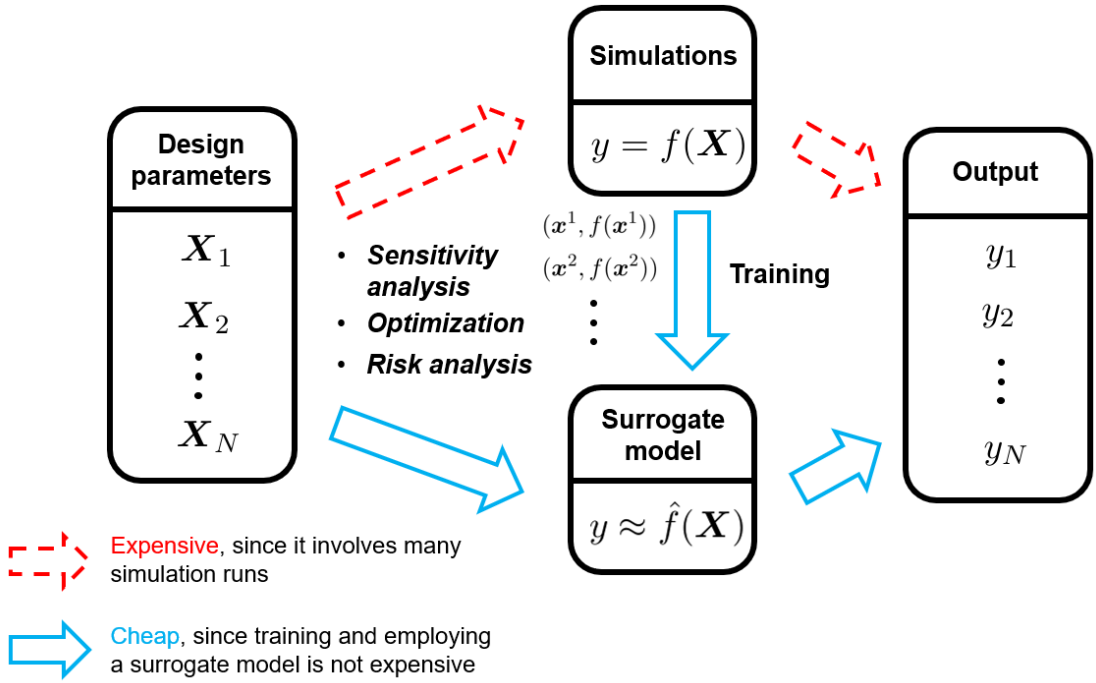


Figure 3.6: Surrogate Modeling from [28]

Because a single assessment of the trained statistical model is typically significantly faster than a single evaluation of the original simulation, completing hundreds of thousands of outputs given different combinations of input parameters is much easier, and make such costly analyses more affordable. A data-driven

technique is used to train a surrogate model. It obtains training data by probing simulation outputs at various rationally or randomly chosen points in the design parameter space. A complete simulation is run at each of these positions to determine the associated simulation outcome. The training dataset is built by collecting the pairs of inputs (design parameters) and outputs from which the statistical model may be developed. Surrogate modeling is a kind of supervised machine learning that is used in engineering design. Popular machine learning approaches, such as polynomial regressions, support vector machines, Gaussian Processes, neural networks, and so on, are also commonly used as surrogate models to speed up the product design and analysis processes.

3.3 Artificial Neural Network

Artificial neural networks (ANNs) [29], also known as neural networks (NNs) or neural nets, are computer systems that are inspired by the biological neural systems in animal brains.

An ANN is built from a network of linked units or nodes known as artificial neurons, which are roughly modeled after the neurons in the human brain. Each link, like synapses in a human brain, has the ability to send a signal to other neurons. An artificial neuron receives impulses, analyses them, and can signal neurons to which it is linked. Each neuron's output is generated by some non-linear function of the sum of its inputs, and the "signal" at a connection is a real number. The connections are referred to as edges. Neurons and edges usually have a weight that changes as learning progresses. The weight changes the intensity of the signal at a connection. Neurons may have a threshold that causes a signal to be transmitted only if the aggregate signal passes it.

3.3.1 Multi-Layer Perceptron

The Multi-layer Perceptron (MLP) [30] is a supervised learning technique that learns a function $f(\cdot) : R^m \rightarrow R^n$ by training on a dataset, in which $m$is the number of dimensions for input and n is the number of dimensions for output. Given a collection of features and an objective y, it can learn a non-linear function approximator for both classification and regression. It differs from logistic regression in that one or more non-linear layers, known as hidden layers, can exist between the input and output layers (Figure 3.7).

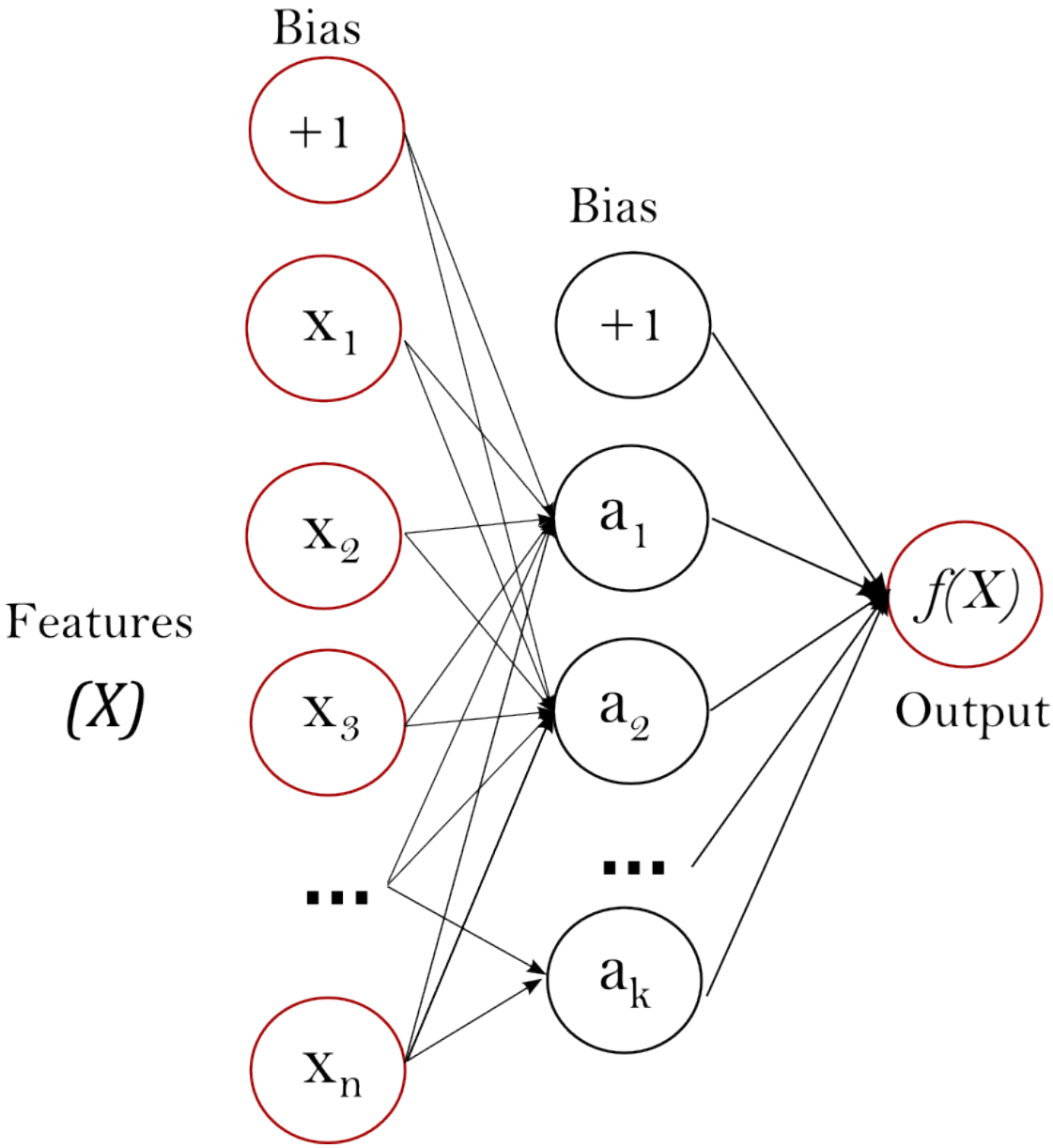


Figure 3.7: Multi-layer perceptron network

The first layer, called the input layer, is made up of neurons that represent the input parameters. Each neuron in the hidden layer performs a weighted linear summation on the data from the preceding layer, followed by a non-linear activation function, such as the hyperbolic tan function. The values from the last hidden layer are received by the output layer and transformed into output values.

The following are the benefits of Multi-layer Perceptron:

1. Possibility of learning non-linear models.
2. The ability to learn models in real-time.

The following are some of the drawbacks of Multi-layer Perceptron (MLP):

1. MLP with hidden layers has a non-convex loss function when there are several local minimums. As a result, different random weight initialisation might result in varying validation accuracy.
2. MLP necessitates the adjustment of several hyperparameters, including the number of hidden neurons, layers, and iterations.
3. Prone to feature scaling

MLP uses Stochastic Gradient Descent, Adam, or L-BFGS to train. SGD updates parameters by applying the gradient of the loss function with respect to a parameter that requires adaptation, i.e.:

$$\omega \leftarrow \omega - \eta(\alpha\frac{\partial R(\omega)}{\partial \omega} + \frac{\partial Loss}{\partial \omega}) \tag{3.1}$$

where η is the learning rate that determines the step size in the parameter space search. The loss function for the network is $Loss$.

3.3.2 Mathematical formulation

Given a set of training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $x_i \in \mathbf{R}^n$ and $y_i \in \{0, 1\}$, a one hidden layer one hidden neuron MLP learns the function $f(x) = W_2 g(W_1^T x + b_1) + b_2$ where $W_1 \in \mathbf{R}^m$ and $W_2, b_1, b_2 \in \mathbf{R}$ are model parameters. W_1, W_2 represent the weights of the input layer and hidden layer, respectively; and b_1, b_2 represent the bias added to the hidden layer and the output layer, respectively. $g(\cdot) : R \rightarrow R$ is the activation function, set by default as the hyperbolic tan. It is given as,

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{3.2}$$

For binary classification, $f(x)$ passes through the logistic function $g(z) = 1/(1+e^{-z})$ to obtain output values between zero and one. A threshold, set to 0.5, would assign samples of outputs larger or equal 0.5 to the positive class, and the rest to the negative class. If there are more than two classes, $f(x)$ itself would be a vector of size $(n_classes,)$. Instead of passing through logistic function, it passes through the softmax function, which is written as,

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)} \tag{3.3}$$

where z_i represents the i th element of the input to softmax, which corresponds to class i, and K is the number of classes. The result is a vector containing the probabilities that sample x belong to each class. The output is the class with the highest probability. Regression results are always expressed as $f(x)$, hence the output activation function is simply the identity function. Different loss functions are used by MLP depending on the type of problem. Average Cross-Entropy, which is presented as the binary case's loss-function for classification, is as follows:

$$Loss(\hat{y}, y, W) = -\frac{1}{n}\sum_{i=0}^{n}(y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)) + \frac{\alpha}{2n}||W||_2^2 \tag{3.4}$$

where $\alpha||W||_2^2$ is an L2-regularization term (aka penalty) that penalizes complex models; and $\alpha > 0$ is a non-negative hyperparameter that controls the magnitude of the penalty.

For regression, MLP uses the Mean Square Error loss function; written as,

$$Loss(\hat{y}, y, W) = \frac{1}{2n}\sum_{i=0}^{n}||\hat{y}_i - y_i||_2^2 + \frac{\alpha}{2n}||W||_2^2 \tag{3.5}$$

Starting with random weights, the multi-layer perceptron (MLP) minimizes the loss function by updating these weights periodically. A backward pass propagates the loss from the output layer to the previous layers, supplying each weight parameter with an update value aimed to reduce the loss.

In gradient descent, the gradient $\nabla Loss_W$ of the loss with respect to the weights is computed and deducted from W. More formally, this is expressed as,

$$W^{i+1} = W^i - \epsilon \nabla Loss_W^i \tag{3.6}$$

where i is the iteration step, and ϵ is the learning rate with a value larger than 0.

The algorithm stops when it reaches a preset maximum number of iterations; or when the improvement in loss is below a certain, small number.

3.4 Python Libraries

3.4.1 Pymoo

A multi-objective optimization problem may be implemented in Python using the Pymoo packag [31]. In Pymoo, every optimization issue must descended from the Problem class. The problem's attributes, including the number of variables (n var), objectives (n obj), and constraints (n constr), are first initialized by invoking the super() method. Additionally, a NumPy array with the bottom (xl) and higher (xu) variable limits is provided. Additionally, the superclass's evaluation method

_evaluate needs to be rewritten. A two-dimensional NumPy array with n rows and m columns is the input for the procedure. An individual is represented by each row, and an optimization variable by each column. The objective values and conditions are then added to the dictionary using keys F and G, respectively, following the appropriate computations.

```
import autograd.numpy as anp
from pymoo.model.problem import Problem

class MyProblem(Problem):
    def __init__(self):
        super().__init__(n_var=2,
                         n_obj=2,
                         n_constr=2,
                         xl=anp.array([-2,-2]),
                         xu=anp.array([2,2]))
    def _evaluate(self, x, out, *args, **kwargs):
        f1 = x[:,0]**2 + x[:,1]**2
        f2 = (x[:,0]-1)**2 + x[:,1]**2
        g1 = 2*(x[:, 0]-0.1) * (x[:, 0]-0.9) / 0.18
        g2 = - 20*(x[:, 0]-0.4) * (x[:, 0]-0.6) / 4.8
    out["F"] = anp.column_stack([f1, f2])
    out["G"] = anp.column_stack([g1, g2])
```

The problem optimization algorithm must then be initialized. For optimization in Pymoo, an algorithm object must be defined. An API description is supplied for each algorithm, and by providing various parameters, algorithms may be modified.

```
from pymoo.algorithms.nsga2 import NSGA2
from pymoo.factory import get_sampling, get_crossover, get_mutation

algorithm = NSGA2(
            pop_size=40,
            n_offsprings=10,
            sampling=get_sampling("real_random"),
            crossover=get_crossover("real_sbx", prob=0.9, eta=15),
            mutation=get_mutation("real_pm", eta=20),
            eliminate_duplicates=True
            )
```

The initialized algorithm object is then used to optimize the defined problem. As a result, the minimize function is called with both the instances problem and the algorithm as inputs. Furthermore, we provide the termination condition of running the algorithm for 40 generations, resulting in 40 + 40 x 10 = 440 function

evaluations. In addition, to assure repeatability, we specify a random seed and activate the verbose parameter to view printouts for each generation.

```
from pymoo.optimize import minimize

res = minimize(MyProblem(),
               algorithm,
               ('n_gen', 40),
               seed=1,
               verbose=True
               )
```

3.4.2 Scikit-Learn

The MLPRegressor class of Scikit-Learn python library implements a multi-layer perceptron (MLP) that trains using backpropagation with no activation function in the output layer, which may alternatively be thought of as employing the identity function as an activation function. As a result, the square error is used as the loss function, and the output is a set of continuous numbers.

```
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
X, y = make_regression(n_samples=200, random_state=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
    random_state=1)
regr = MLPRegressor(random_state=1, max_iter=500).fit(X_train,
    y_train)

regr.predict(X_test[:2])

regr.score(X_test, y_test)
```

Chapter 4

Methodology

4.1 Optimization

Figure 4.1 depicts the input/output sequence of a basic PREDYCE Optimization use scenario. Currently, in order to run a PREDYCE script from the command line, an EnergyPlus command line launch structure made of options and an IDF model was maintained (e.g., -w option is followed by the EPW weather file, -i by the IDD version), easing the tool's usage for users already familiar with EnergyPlus software. The following are the major required input files for the task:

1. the building model in IDF format
2. the weather file in EPW format
3. a PREDYCE JSON file organized to include all user requests (e.g., KPIs to be computed, parameters to be modified, run period)
4. a OPTIMIZATION JSON file with the maximum and lowest values that may be assumed for each parameter that the user desires to change.

While the main outcomes will be:

1. a CSV file called data res.csv that contains aggregated KPIs for all simulations executed within the specified run period.
2. Plots to display optimization outcomes

As was mentioned earlier, the tool uses input and database JSON files to describe actions and KPIs that will be carried out and retrieved during a pool of simulations as well as to store information needed to adjust IDFs in dictionary form (e.g., materials composition, schedules, construction elements). The format of a typical PREDYCE JSON input file is seen in Figure 4.2: The tool uses the

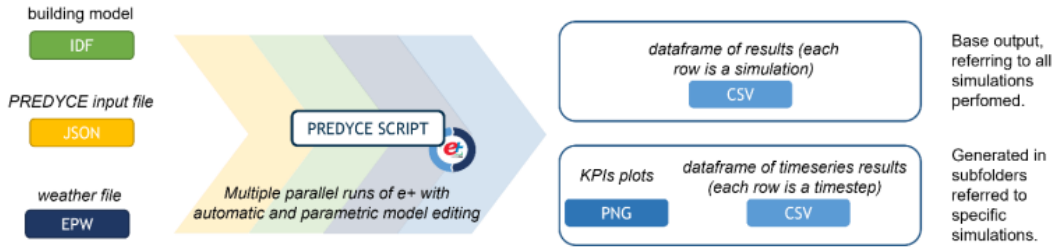


Figure 4.1: Input/output sequence of a basic PREDYCE Optimization from [17]

building name, which is the name of the IDF's central structure, to determine which zone components to update and do calculations on. Before performing the simulations for the parametric analyses, the preliminary actions are the steps that are only taken once. All simulated buildings share the identical alterations that are specified in the preliminary action section. The actions are the parametric changes that must be made to the structure; all actions and their parameters are combined to produce a series of simulations, each of which comprises a different variant (combination of actions) of the original structure. The final indicators that are calculated at the conclusion of each simulation are included in the KPI section. Figure 4.3 shows the OPTIMIZATION JSON file structure in which the threshold values of the individual parameters are reported, from which the genetic algorithm that will perform optimization draws the information to create a random population of parameters within the space defined by the user.

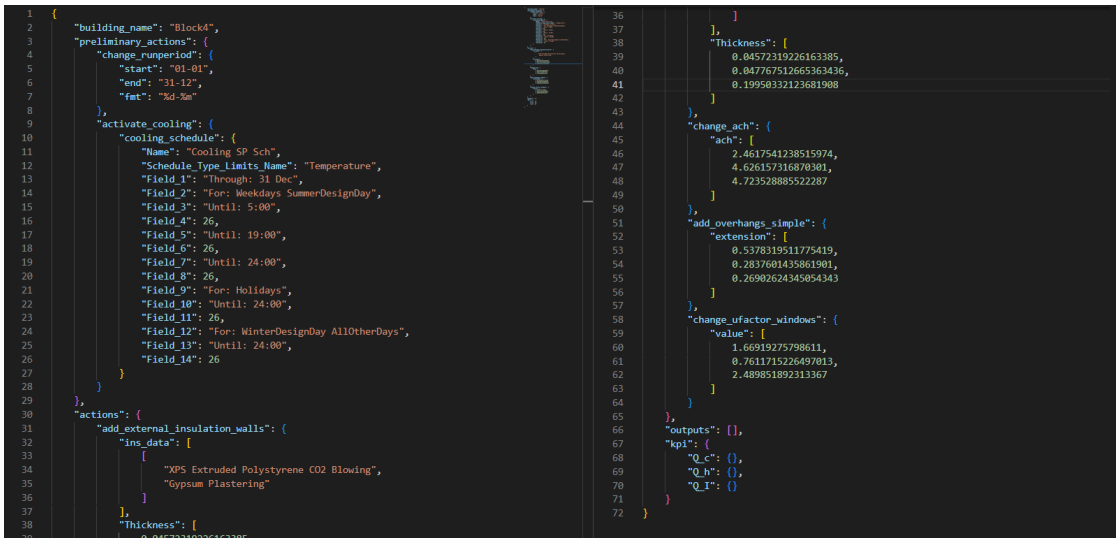
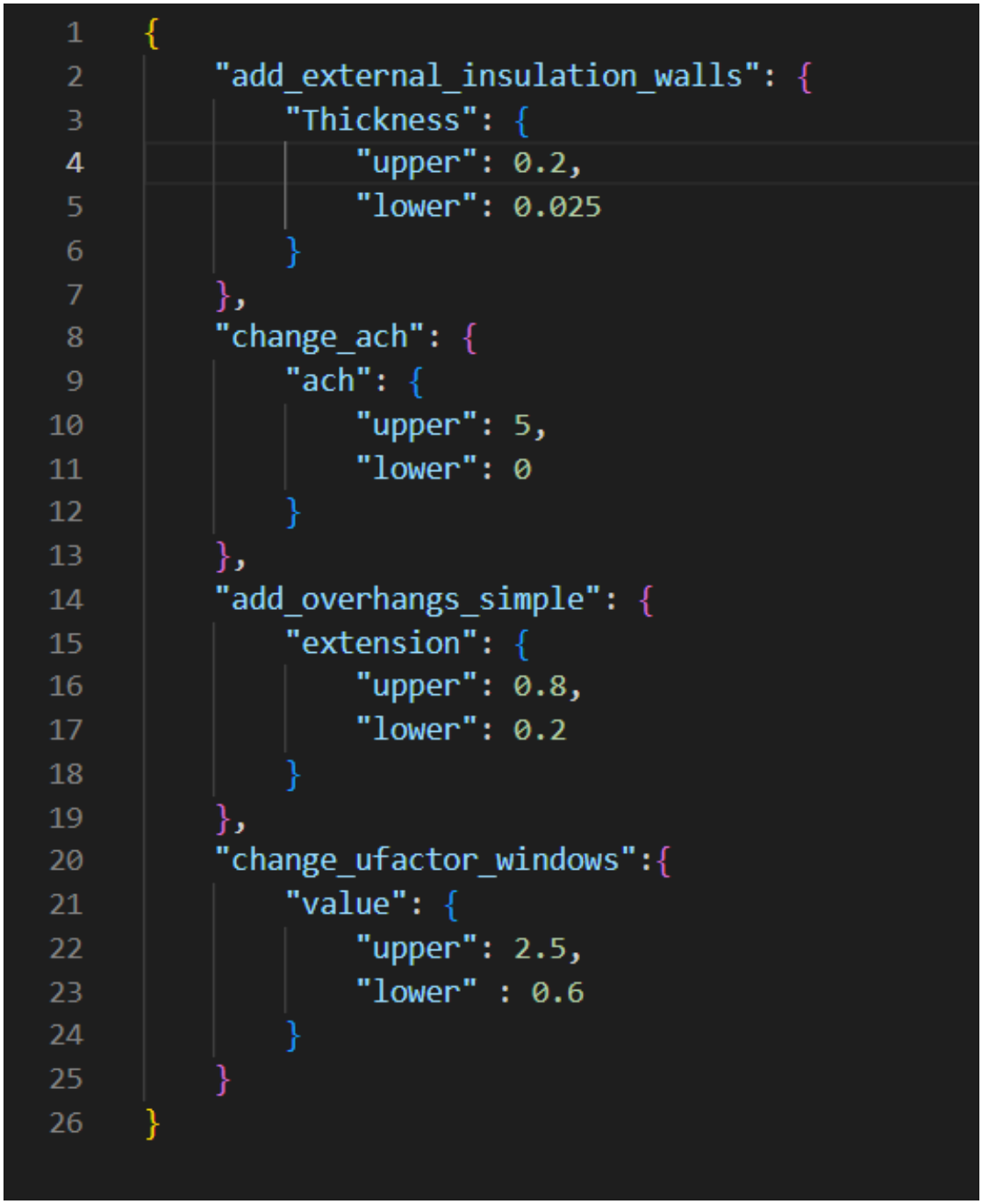


Figure 4.2: Input JSON file



```
{
    "add_external_insulation_walls": {
        "Thickness": {
            "upper": 0.2,
            "lower": 0.025
        }
    },
    "change_ach": {
        "ach": {
            "upper": 5,
            "lower": 0
        }
    },
    "add_overhangs_simple": {
        "extension": {
            "upper": 0.8,
            "lower": 0.2
        }
    },
    "change_ufactor_windows":{
        "value": {
            "upper": 2.5,
            "lower" : 0.6
        }
    }
}
```

Figure 4.3: Optimisation range parameters

To perform the optimization on the parameters described in the JSON above, the functions of the pymoo library are used (see Chapter 3). The user must specifically indicate:

1. the type of problem (e.g. maximization / minimization)
2. the particular algorithm to be used (for example, nsga2 / nsga3)
3. the number of population samples generated by the algorithm at each iteration
4. how many times the algorhythm must recreate a new population (number of generations)
5. the number of reference points (only in the case of NSGA3)

When the software is launched, the input JSON file is modified (using a specific JSONeditor.py script), resulting in an initial population of samples that follows the user's requests with random values within the specified ranges. EnergyPlus will next execute a number of simulations equal to the number of samples in the population, and the user's requested KPI will be generated for each of them(figure 5.7). The optimization method picks and saves the "non-dominated" solutions from among these. The operation is repeated as many times as the number of generations specified by the user.

The script will deliver a list of all optimal solutions, that is, the points in the constructed space that belong to the Pareto Front, at the end of the operation. When the number of KPI (i.e., objectives) required by the user is less than or equal to 3, a graph displaying all points belonging to the optimal front can be visualized.

```
Number of simulations: 10
Number of processes allocated in the pool: 10
Number of CPUs: 12
Started at: 2022-10-15 14:50:06
Finished at: 2022-10-15 14:51:55
Elapsed time: 0:01:49.257704
   add_external_insulation_walls.ins_data add_external_insulation_walls.Thickness change_ach.ach  ... change_ufactor_windows.value        Q_c       Q_h
5  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.04051      4.218266  ...                         2.499907  37.794584  5.405875
2  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.191508     3.832032  ...                         1.95019   42.072974  4.238782
0  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.072372     0.030156  ...                         2.336169  39.007302  4.509628
1  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.058594     0.079184  ...                         1.95498   37.32511   4.774107
3  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.051991     0.125488  ...                         2.498757  38.692798  4.938681
7  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.073401     0.091848  ...                         2.336169  39.202464  4.494803
8  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.041612     3.989609  ...                         2.351363  36.655918  5.370178
4  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.02976      4.166337  ...                         1.95019   34.370905  6.198514
6  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.026711     3.911077  ...                         2.351363  34.686177  6.461128
9  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.178118     3.711192  ...                         1.95498   41.833579  4.238782
```

Figure 4.4: Population example

4.2 Optimization with Surrogate Model

This scenario is very similar to the previous one, but the KPI requested by the user will no longer be calculated through an EnergyPlus simulation, but will be predicted by a neural network, which will be assisted by a relatively small number of simulations carried out using the previously mentioned "white-box" method. The first step is to run the code from the command line, specifying the same

input files used in the first scenario, to obtain a file CSV containing all parameter combinations and relative KPI calculated by Energyplus, which will be used for network training. The user at this point, through the scikit-learn library described in the paragraph 3 will have to specify the settings for the model training phase. In particular:

1. the type of model you intend to use (e.g. MLPregression)

2. The hyperparameters that are intended to be modified in the gridSearch (e.g. "hidden-layer-size" , "learning-rate" etc...)

3. The percentage of simulations used for train and test (e.g. 70-30 respectively)

The program then performs gridsearch, and once the best hyperparameters are found, it proceeds with the model fit using the train database. The performance of the neural network is measured using the following information, which the program will output:

1. Mean square error, mean absolute percentage error and R squared (calcolati confrontando i risultati ottenuti dal modello rispetto al test database)

2. Residuals plot

3. scatter plot with real values versus predicted values

4. Model score evaluated on train and test database and hyparameters selected from gridsearch

Using the Python library "pickle [32]," the model that has been created thus far will be saved and reused in the final step of this scenario. With the exception of substituting the simulation with EnergyPlus within the evaluate module of the pymoo library with the KPI calculation using the newly trained neural network, retrieved using the same "pickle" library that had made it possible to save it in memory, we perform an optimization process very similar to the one described in the previous step in this final section.

Finally, the following conclusive plots are returned from the script:

1. Visualisation of the obtained Pareto Front

2. Two graphs showing the correlation between the value of individual parameters and the calculated KPI. the first referring to the initial simulations performed with EnergyPlus and used to train and test the network. The second for the values obtained using the neural network.

4.3 Surrogate model with metereological data

The approach in the latter scenario is different. Its goal is to create a surrogate model that can predict building KPI without relying on the building's construction parameters. However, after selecting a building configuration, one wishes to estimate the KPI required by the user, using daily or hourly meteorological data from a given area as input to train the neural network.

The input to the code's launch are the same as in previous cases, with the exception of the JSON file, which contains the range of values used to vary the building's parameters, which are fixed in this scenario. Furthermore, for the same reason, in the JSON file where the actions are listed, there is no need to add more than one value for each parameter that needs to be specified. As a result, only one EnergyPlus simulation will be run, but this time the values calculated by PREDYCE will be ignored in favor of the EnergyPlus file eplusout.csv. This specific file (4.5) contains as many rows as there are timesteps within the user-selected run period. As for timestep, you can choose between monthly, daily, hourly, and sub-hourly frequencies.

The rows in the CSV file are the simulation predictions for each timestep. From this file, the columns containing information about:

1. climatic variables of interest (e.g. "external dry-bulb temperature")

2. the EnergyPlus output that we want to predict using the surrogate model (e.g. heating consumption, comfort score, etc..)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date/Time	Environment:Site Outdoor Air Drybulb Temperature [C]	Enviro	Enviror	Enviro	Enviro	Environment:Site Diffuse Solar Radiation Rate	Environme	Environme	Environme	BLOCK4:C/	BLOCK4:C/	BLOCK4:C
2	09/01	16.47916667	8.75	1E+05	4.2	106	56.42708333	314.8634	176.0796	7.814691	0.471769	1.082729	0.4389
3	09/02	16.67638889	13.4	1E+05	3.68	116	102.2372685	114.5845	176.1563	7.484176	0.667537	1.532024	0.62111
4	09/03	17.02673611	12.5	1E+05	0.68	221	81.40972222	205.1354	176.2342	7.15158	0.667537	1.532024	0.62111
5	09/04	17.87430556	13.2	1E+05	1.78	202	56.06597222	279.2951	176.3136	6.816987	0.471769	1.082729	0.4389
6	09/05	17.46284722	13.8	1E+05	2.57	138	44.51041667	324.8403	176.3948	6.480482	0.471769	1.082729	0.4389
7	09/06	16.15868056	12.9	1E+05	2.02	85.8	76.47916667	121.0104	176.477	6.142148	0.471769	1.082729	0.4389
8	09/07	13.47291667	9.86	1E+05	4.19	125	95.26736111	126.7083	176.5602	5.80207	0.471769	1.082729	0.4389
9	09/08	13.43715278	9.89	1E+05	8.33	106	93.69097222	25.66667	176.6441	5.460333	0.471769	1.082729	0.4389
10	09/09	11.92361111	8.25	1E+05	5.48	219	94.8900463	39.52778	176.7288	5.117022	0.667537	1.532024	0.62111
11	09/10	11.50381944	7.48	1E+05	4.14	140	97.80902778	81.18403	176.8142	4.772223	0.667537	1.532024	0.62111
12	09/11	10.23611111	5.63	1E+05	5.03	126	43.24537037	296.2211	176.9001	4.426022	0.471769	1.082729	0.4389
13	09/12	10.59791667	9.69	1E+05	5.05	159	30.29861111	0	176.9866	4.078505	0.471769	1.082729	0.4389
14	09/13	11.03819444	10.3	1E+05	3.34	123	58.53587963	0.5	177.0734	3.729758	0.471769	1.082729	0.4389
15	09/14	14.81041667	11.6	1E+05	5.77	201	44.02083333	78.25	177.1607	3.37987	0.471769	1.082729	0.4389
16	09/15	13.02083333	11.7	1E+05	7.15	168	28.03472222	0	177.2482	3.028927	0.471769	1.082729	0.4389
17	09/16	13.66006944	9.56	1E+05	7.58	168	68.75	203.4097	177.3358	2.677018	0.667537	1.532024	0.62111
18	09/17	9.730902778	6.81	1E+05	5.19	157	50.38541667	97.29167	177.4236	2.32423	0.667537	1.532024	0.62111
19	09/18	12.92847222	6.31	1E+05	2.37	113	25.29861111	348.8333	177.5114	1.970653	0.471769	1.082729	0.4389
20	09/19	12.384375	8.04	1E+05	3.08	112	66.37152778	159.125	177.5992	1.616374	0.471769	1.082729	0.4389
21	09/20	12.540625	8.87	1E+05	2.33	141	59.37847222	190.7917	177.6867	1.261485	0.471769	1.082729	0.4389
22	09/21	13.17430556	11	1E+05	2.68	117	58	73.04167	177.7741	0.906074	0.471769	1.082729	0.4389
23	09/22	12.79895833	8.34	1E+05	4.56	193	28.17013889	320.9873	177.8611	0.550231	0.471769	1.082729	0.4389
24	09/23	14.15520833	9.72	1E+05	5.9	193	78.05555556	43.07639	177.9478	0.194048	0.667537	1.532024	0.62111

Figure 4.5: eplusout.csv example

As a result, a database is created using the library "numpy" [33] using a procedure similar to the previous one, dividing the train and test datasets and running a

gridsearch to find the best hyperparameters to use for the neural network's fit and test.

In this case, the code will also be output:

1. Mean square error, mean absolute percentage error and R squared (Calculations that compare the model's results to the test database)
2. Residuals plot
3. Scatter plot with real values versus predicted values
4. Model score evaluated on train and test database and hyparameters selected from gridsearch

In addition, in this type of scenario, a heat-map will be displayed to show the degree of correlation between all input and output variables taken into account, allowing the user to make changes to the parameter selection in the event of a second code launch.

Chapter 5

Results

Some practical examples of applying the methods described in Chapter 5 will be provided in the following paragraphs. To carry out these analyses, a reference model of a building (Figure 5.1 5.2 5.3) was created using the software DesignBuilder.

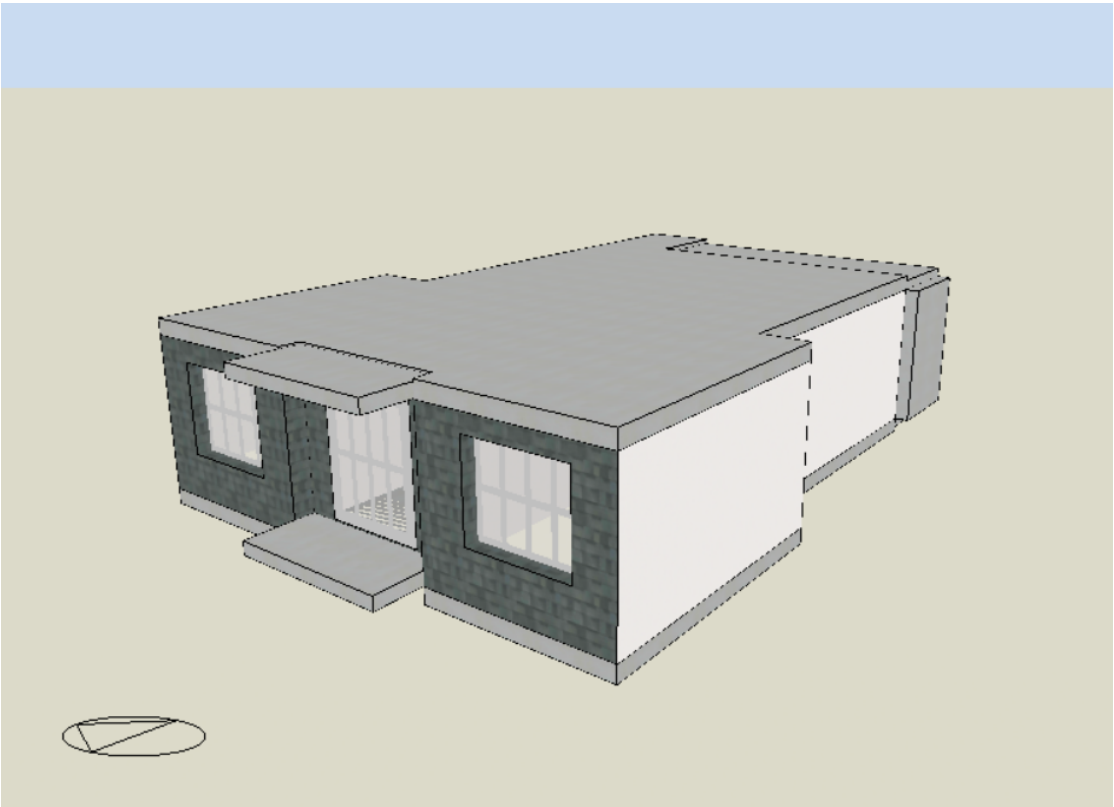


Figure 5.1: Render view

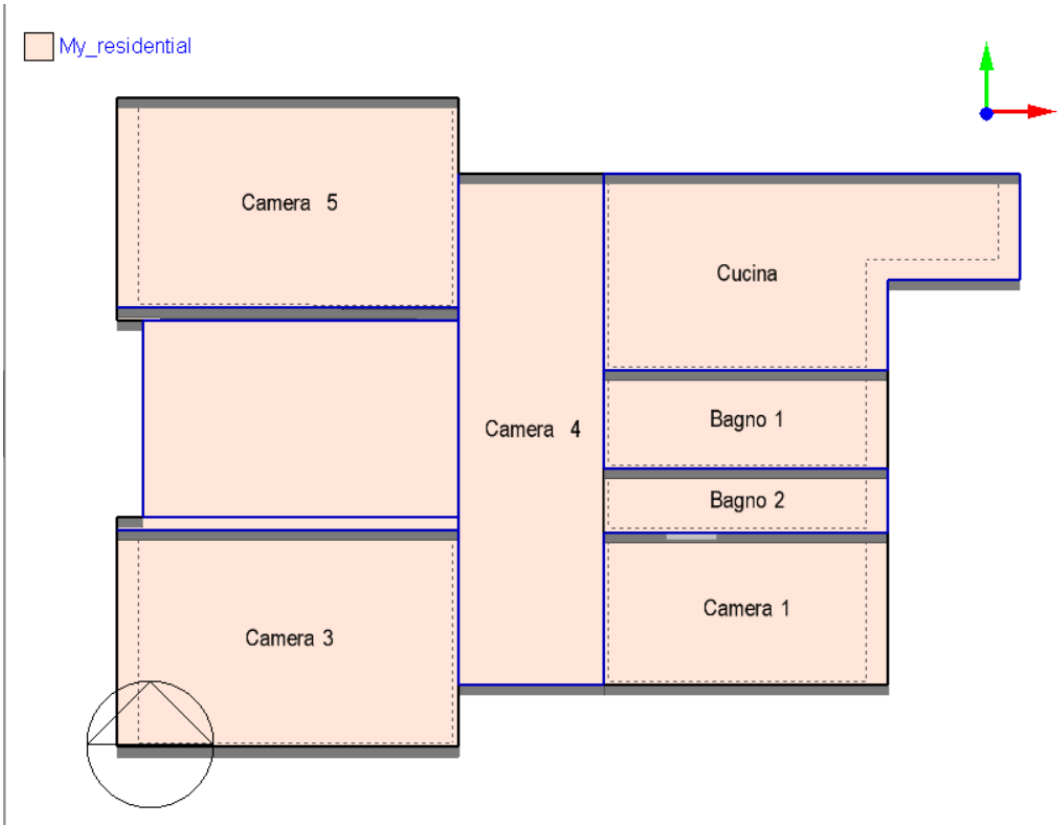


Figure 5.2: Building plant

Because it is a residential apartment on the seventh floor of a building, the ceiling and floor are modeled as adiabatic surfaces, as are the north and south walls that connect the apartment to the rest of the building. There are three bedrooms, a living room, a kitchen, and two bathrooms on $110m^2$ of space. The apartment's eight windows and two balconies are located on the east and west sides. The most important information about building materials for walls and windows can be found in the figures 5.4 for exterior walls, 5.5 for interior partitions, 5.6 for windows.

While as far as the .epw files is concerned, the weather file of Torre Pellice (Turin) from the year 2022 was used for the first 3 cases, and in the last scenario also the weather files of Aalborg (Denmark) and Athens (Greece), both referring to the year 2022.

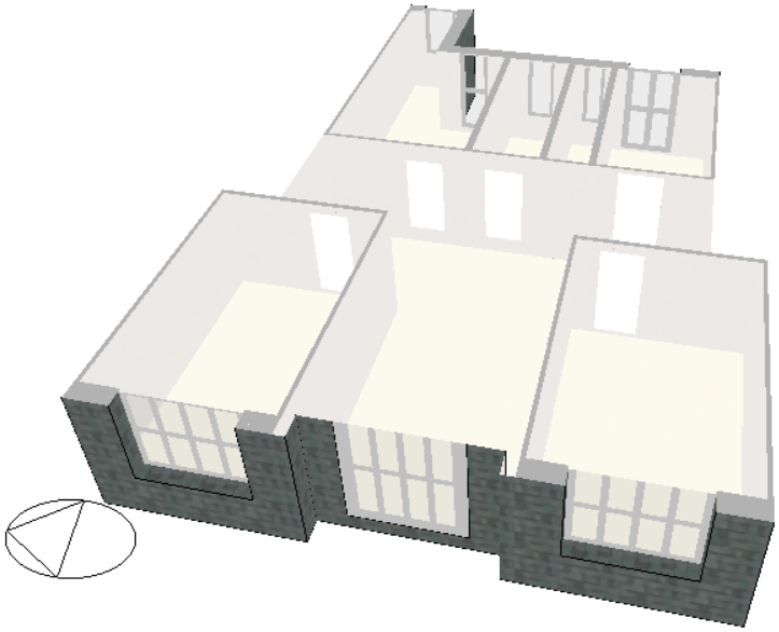


Figure 5.3: Interior render view

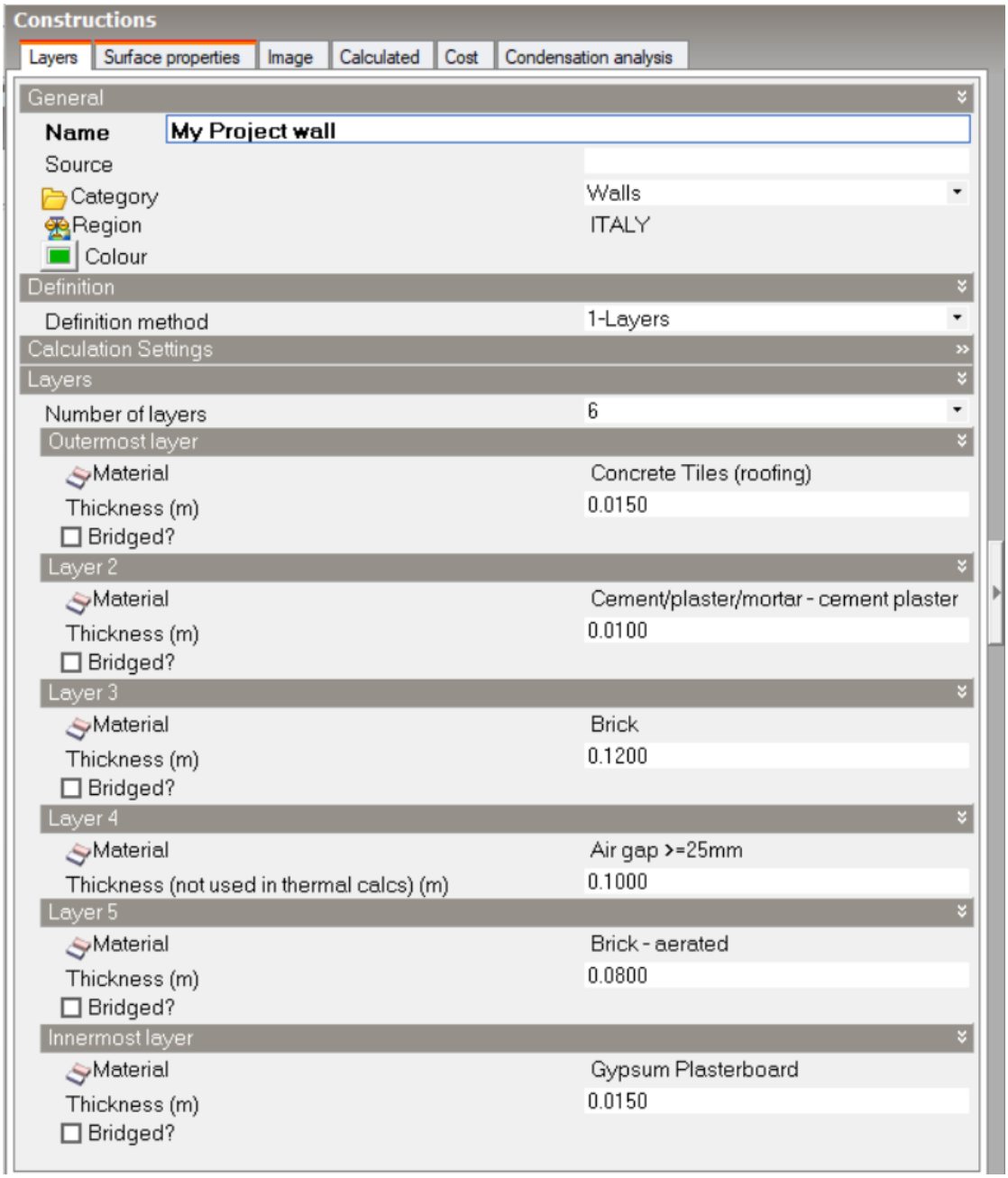


Figure 5.4: External walls construction layers

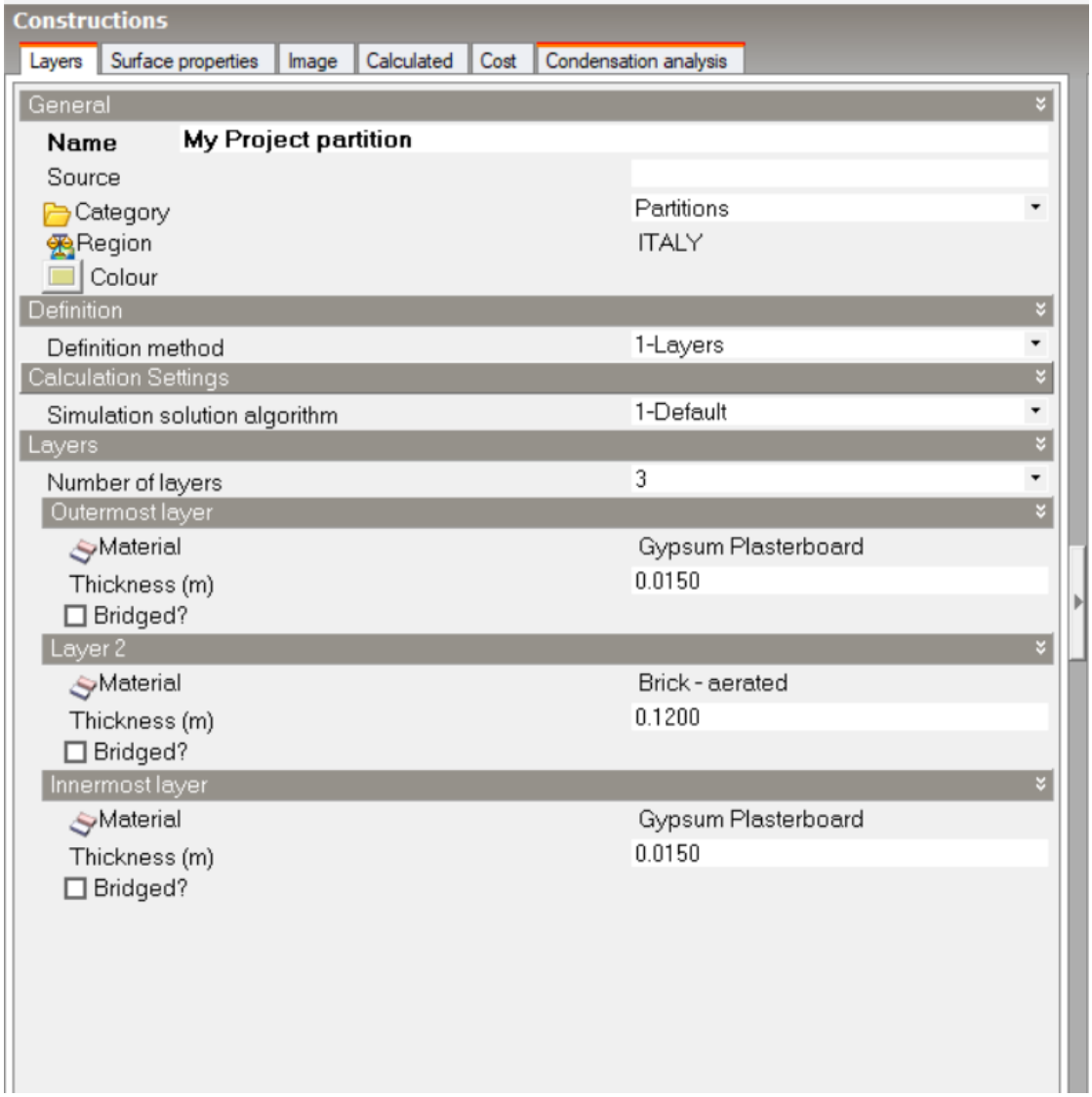


Figure 5.5: Internal walls construction layers

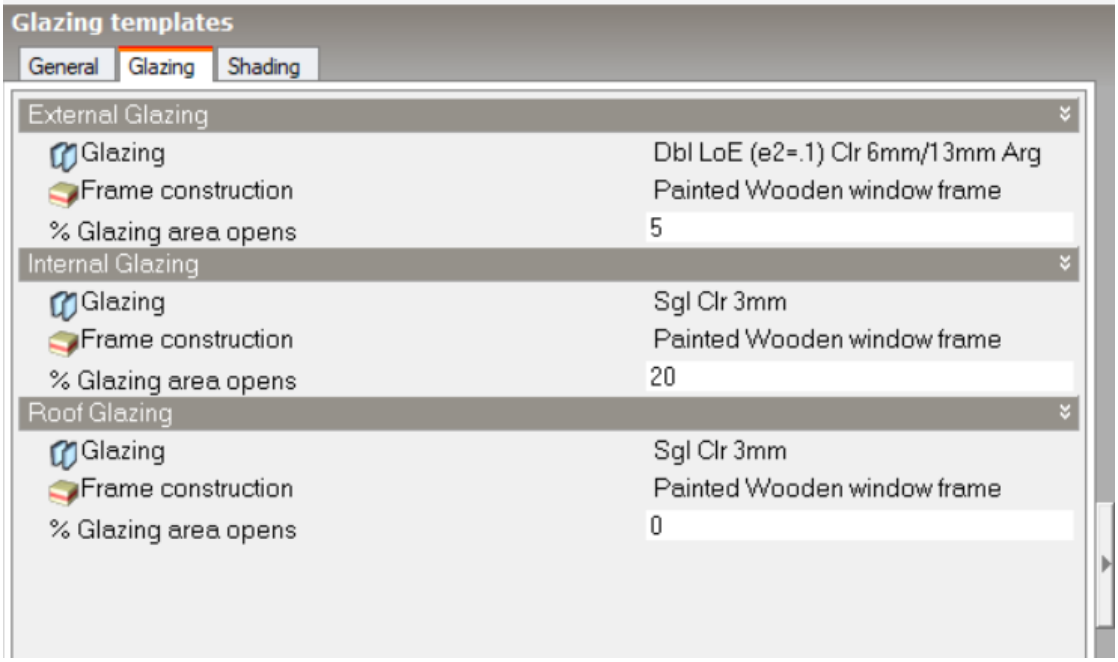


Figure 5.6: Windows construction layers

5.1 Example of Optimization scenario - NSGA2

To evaluate the effectiveness of the NSGA2 optimization algorithm, it was decided to use the yearly consumption of the building in terms of insulation and heating. The following four "actions" have been used to modify some aspects of the building (as reported in the input file PREDYCE JSON):

1. `predyce.idf_editor.change_ufactor_windows(idf, value)`

   Set an U-Factor value to all windows of type SimpleGlazingSystem. Parameters:

   (a) `idf (class:predyce.IDF_class.IDF)`
   – IDF object

   (b) `value (float)`
   – New U-Factor value

2. `predyce.idf_editor.add_overhangs_simple`

   `(idf, extension=1, tilt=90, shift=0.04)`

Add simple overhang.
Parameters:

(a) `idf (class:predyce.IDF_class.IDF)`
  – IDF object

(b) `extension (int, optional)`
  – Extension of the overhang, defaults to 1

(c) `tilt (int, optional)`
  – Tilt of the overhang, defaults to 90

(d) `shift (float, optional)`
  – Shift of the overhang, defaults to 0.04

---

3. `predyce.idf_editor.add_external_insulation_walls`

   `(idf, ins_data=None, filter_by='', **fields)`

   Add external insulation layer and plaster layer on external walls. Since layers are added externally internal building area and volume are not impacted. Parameters:

   (a) `idf (class:predyce.IDF_class.IDF)`
     – IDF object

   (b) `ins_data (list of two string elements, optional)`
     – Construction objects list (insulation material and plaster), defaults to None

   (c) `filter_by (str, optional)`
     – Filter zones by name. Can be the block name or specific zone name, defaults to “”

   (d) `fields`
     – Additional fields to be set, passed as keyword arguments

The range of values within which these four parameters can vary is defined in the OPTIMIZATION JSON file and is shown in the table 5.1 below:

Parameter	Upper value	Lower value
external insulation layers (thickness)	0.2	0.025
ach	5	0
overhangs extension	0.8	0.2
U-factor windows	2.5	0.6

Table 5.1: Range parameters

In addition, to quantify consumption, the PREDYCE KPI "Q_h" and "Q_c" are defined in the JSON file.

1. `class predyce.kpi.EnergyPlusKPI`

   `Q_h`

   `(em_factor=1,reg_factor=1,distr_factor=1,gen_factor=1,ep_factor=1)`

   Prepare eplosout.csv to compute primary energy need for heating in kWh/m2, applying correction factors for emission, regulation, distribution and generation losses, then multiplying by primary energy factor. Parameters;

   (a) `em_factor`(float, default 1) – correction factor for emission losses, in range [0,1].

   (b) `reg_factor` (float, default 1) – correction factor for regulation losses, in range [0,1].

   (c) `distr_factor` (float, default 1) – correction factor for distribution losses, in range [0,1].

   (d) `gen_factor` (float, default 1) – correction factor for generation losses, in range [0,1].

   (e) `ep_factor` (float, default 1) – primary energy conversion factor, in range [0,1].

   Primary energy need for heating Return type float

2. `class predyce.kpi.EnergyPlusKPI`

   `Q_c`

   `(em_factor=1,reg_factor=1,distr_factor=1,gen_factor=1,ep_factor=1)`

   Prepare eplosout.csv to compute primary energy need for cooling in kWh/m2, applying correction factors for emission, regulation, distribution and generation losses, then multiplying by primary energy factor. Parameters;

   (a) `em_factor`(float, default 1) – correction factor for emission losses, in range [0,1].

   (b) `reg_factor` (float, default 1) – correction factor for regulation losses, in range [0,1].

(c) `distr_factor` (float, default 1) – correction factor for distribution losses, in range [0,1].

(d) `gen_factor` (float, default 1) – correction factor for generation losses, in range [0,1].

(e) `ep_factor` (float, default 1) – primary energy conversion factor, in range [0,1].

Primary energy need for cooling Return type float

In terms of optimization algorithm settings, a `population_size` = 10 and a `number_of_generations` = 10 have been chosen, for a total of 100 different simulations that will be run by EnergyPlus. In the figure 5.7, an example of a population generated by the program in a single iteration of the algorithm is shown.

```
Number of simulations: 10
Number of processes allocated in the pool: 10
Number of CPUs: 12
Started at: 2022-10-15 14:50:06
Finished at: 2022-10-15 14:51:55
Elapsed time: 0:01:49.257704
   add_external_insulation_walls.ins_data add_external_insulation_walls.Thickness change_ach.ach  ... change_ufactor_windows.value       Q_c       Q_h
5  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.04051       4.218266  ...                          2.499907  37.794584  5.405875
2  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.191508      3.832032  ...                          1.95019   42.072974  4.238782
0  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.072372      0.030156  ...                          2.336169  39.007302  4.509628
1  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.058594      0.079184  ...                          1.95498   37.32511   4.774107
3  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.051991      0.125488  ...                          2.498757  38.692798  4.938681
7  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.073401      0.091848  ...                          2.336169  39.202464  4.494803
8  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.041612      3.989609  ...                          2.351363  36.655918  5.370178
4  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.02976       4.166337  ...                          1.95019   34.370905  6.198514
6  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.026711      3.911077  ...                          2.351363  34.686177  6.461128
9  [XPS Extruded Polystyrene CO2 Blowing, Gypsum ...                        0.178118      3.711192  ...                          1.95498   41.833579  4.238782
```

Figure 5.7: Population example

The optimization algorithm's results are collected in the table 5.2and 5.3 and plotted in 5.8.

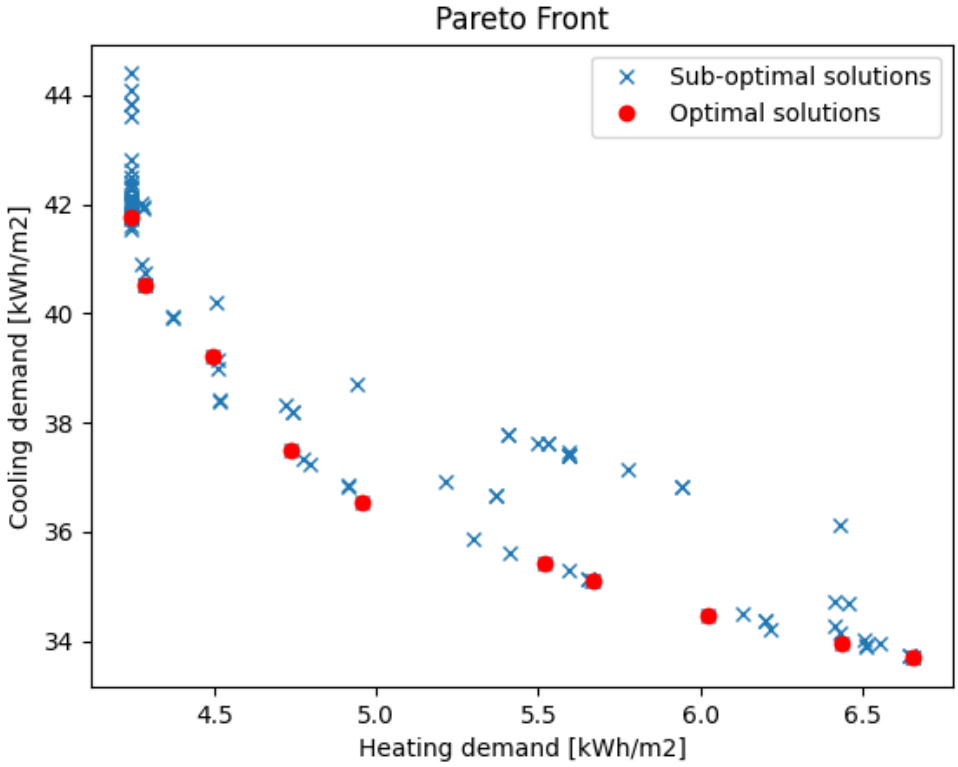


Figure 5.8: Pareto Front (NSGA2)

Q_h[**kWh/m²**]	Q_c[**kWh/m²**]
6.65994887	33.69224876
4.23878238	41.26321231
4.49480282	39.20246354
4.73772248	32.50229932
4.95736529	36.54262851
5.52018012	35.41922911
6.02619611	34.48299198
5.67160084	35.11029891
4.28262026	49.52289614
6.43572347	33.94868526

Table 5.2: Optimum solutions (NSGA2)

InsulationLayer[m]	Ach	Overhangs.extension[m]	U-factor.windows
0.02508885	3.84243725	0.75451836	1.95532076
0.1748817	3.83203216	0.78925307	1.95533367
0.07340091	0.0918481	0.57414189	2.3361689
0.06006117	3.83203216	0.74366178	2.41522721
0.05237747	3.83173585	0.79341151	1.95026845
0.03929166	3.83206607	0.78925307	2.42160995
0.03187373	3.84243725	0.79839574	1.98786399
0.03678158	3.82979811	0.79341151	2.42322217
0.1066005	4.80877817	0.64102581	1.99565975
0.02725227	3.82921768	0.77196224	1.95555858

Table 5.3: Optimum parameters (NSGA2)

5.1.1 Results discussion

The results show that the optimization algorithm can find the configurations that best satisfy the problem, in this case minimization of the annual heating and cooling consumption of the building, in a set of numerous simulations. However, the points on the pareto front (5.8) could be subjected to further sorting, depending on which of the two objectives is more important to the user. Indeed, the figure shows that the points at the two extremes of the front tend to minimize one of the two consumptions rather than the other, whereas the "middle" points could be considered the "right compromise" for both objectives. However, this approach, while effective, has a flaw. In fact, analyzing the entire space of combinations of our input parameters necessitates a large number of simulations, which, as previously stated, can be very wasteful in terms of computational complexity and execution time. Another interesting aspect of using this approach when optimizing building parameters is that we can identify which of our input parameters have the property of influencing the outputs in a concordant way. If, for example, we see in the table (**??**) that a specific parameter takes values in a very narrow range at all points belonging to the pareto front, we can assume that that specific parameter minimizes Heating and Cooling consumption simultaneously. If, on the other hand, a parameter in the table has a similar range of variation to that given as input by the user, we can conclude that that parameter affects the value of the two outputs in opposite ways.

5.2 Example of Optimization scenario - NSGA3

The previous studio case was used again to test the NSGA3 optimization algorithm, with some minor changes. Instead, from the time this algorithm is recommended for resolving optimization problems with three or more objectives, it takes into account, in addition to heating and cooling, the building's electric consumption. This final calculation was performed using the KPI "Q I" suggested by the PREDYCE tool, as shown below.

```
def Q_I(self, em_factor=1,

reg_factor=1, distr_factor=1, gen_factor=1,ep_factor=1):
```

> Prepare eplosout.csv to compute primary energy need for lighting in kWh/m2, applying correction factors for emission, regulation, distribution and generation losses, then multiplying by primary energy factor.
>
> Parameters:
>
> - em_factor: correction factor for emission losses, in range [0,1].
> - reg_factor: correction factor for regulation losses, in range [0,1].
> - distr_factor: correction factor for distribution losses, in range [0,1].
> - gen_factor: correction factor for generation losses, in range [0,1].
> - ep_factor: primary energy conversion factor, in range [0,1].
>
> return: Primary energy need for lighting
> rtype: float

Furthermore, as previously stated in the understanding of "methodology," the use of this algorithm necessitates defining, in addition to `n_population = 33` and `n_generations = 3`, the number of reference directions and the method for generating the same. In this example, `reference_directions = 3` were selected using the `das-dennis` method.

The obtained results are summarized below.

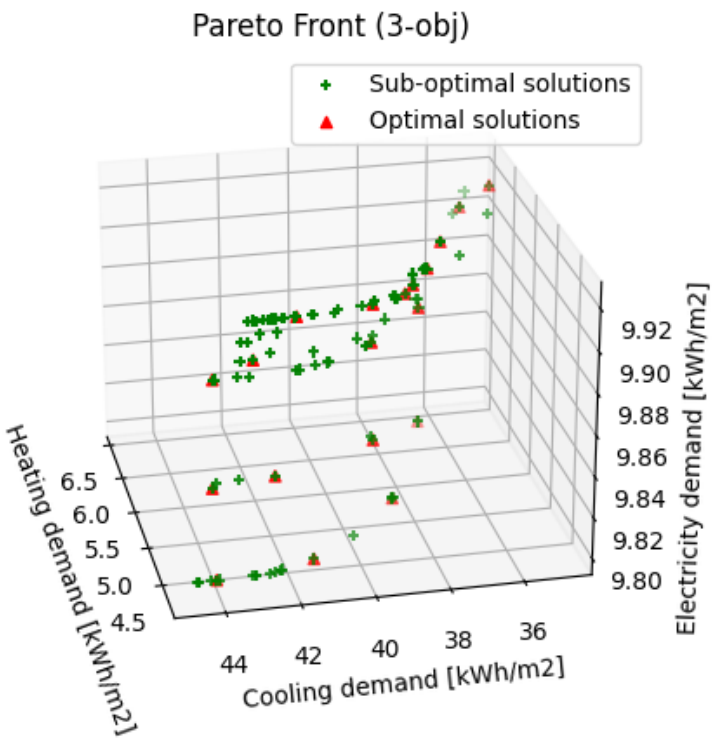


Figure 5.9: Pareto Front (NSGA3)

5.2.1 Results discussion

All of the considerations made in the preceding paragraph are obviously valid in this case. Another thing to think about is the placement of the points on the tridimensional front, as seen in the figure 5.9. It is worth noting that the distribution of optimal values does not have the typical sail shape that one would expect from a graph of this type, but instead maintains the hyperbolic shape that was revealed in the previous case. This characteristic can be attributed to the fact that the third objective that is considered is clearly unaffected by any of the four parameters that we can change within our space.

5.3 Example of Optimization with Surrogate Model

In this scenario the same parameters as in the previous paragraph have been chosen as input.

The initial database with which to train the net was created using the parameters shown in the table 5.4. A total of 81 different simulations were therefore performed by EnergyPlus, considering all the possible combinations assumed by the input parameters, and the KPI calculated by PREDYCE were "`Q_h`" and "`Q_c`".

Parameter	Values
external insulation layers (thickness[m])	0.15 0.10 0.01
ach	5 2.5 1
overhangs extension [m]	0.9 0.7 0.3
U-factor windows	0.6 1.2 1.8

Table 5.4: Initial database configuration

And the results obtained by EnergyPlus in terms of consumption are shown in the figure 5.10 5.11.

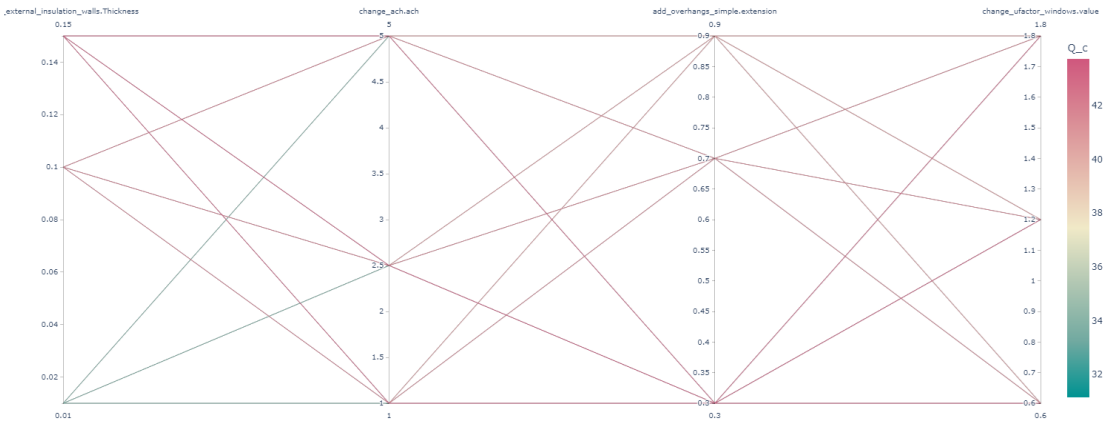


Figure 5.10: EnergyPlus results of Cooling demand

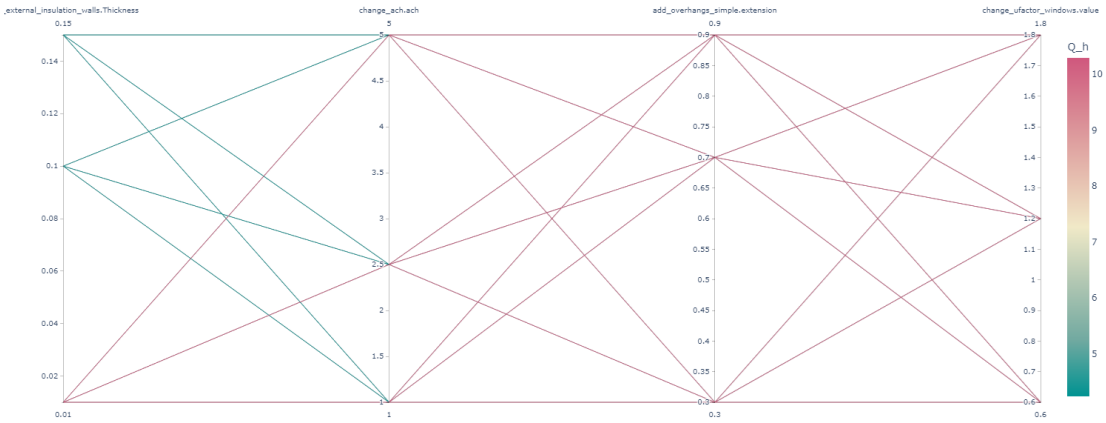


Figure 5.11: EnergyPlus results of Heating demand

The database thus created was used for training the network, dividing the 144 simulations between Train and Test with a ratio of 70/30. The figures 5.12 5.13 5.14 5.15 below show the performance report of the neural network created and tested on the portion of simulations not used during the training phase, and in the table 5.5 the value of the calculated errors.

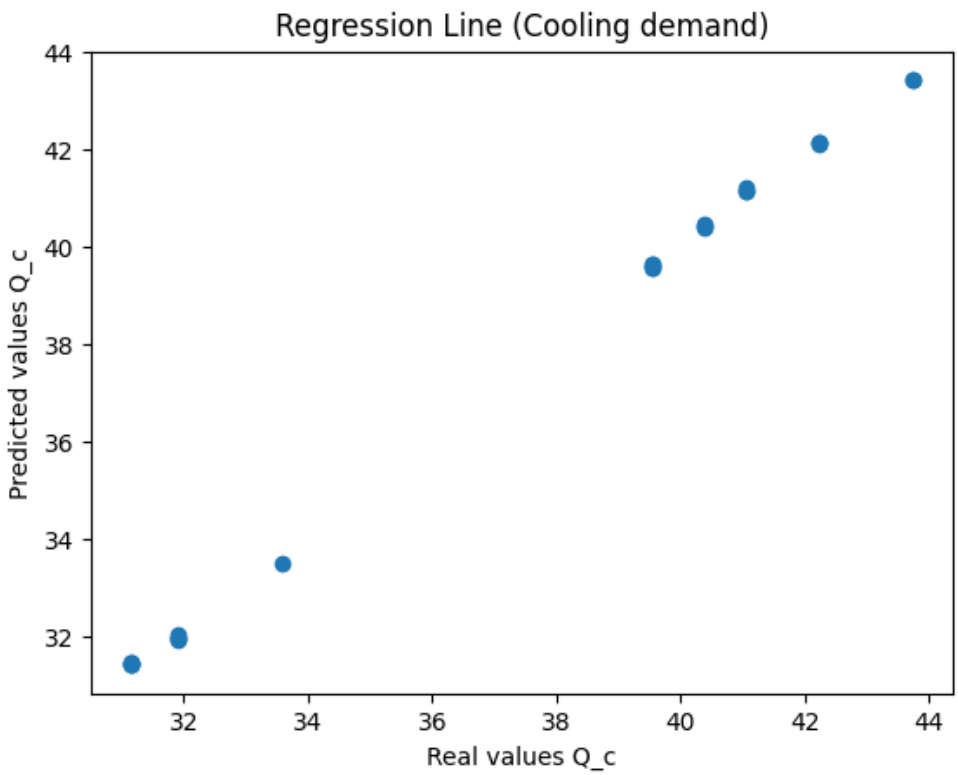


Figure 5.12: Regression Line for Cooling

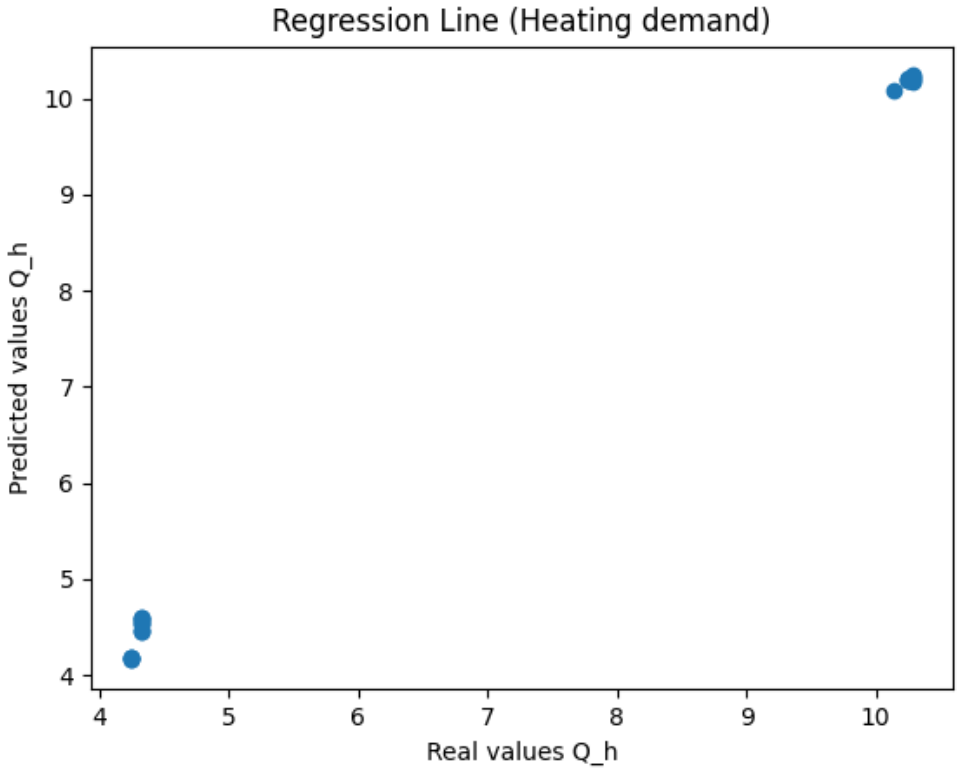


Figure 5.13: Regression Line for Heating

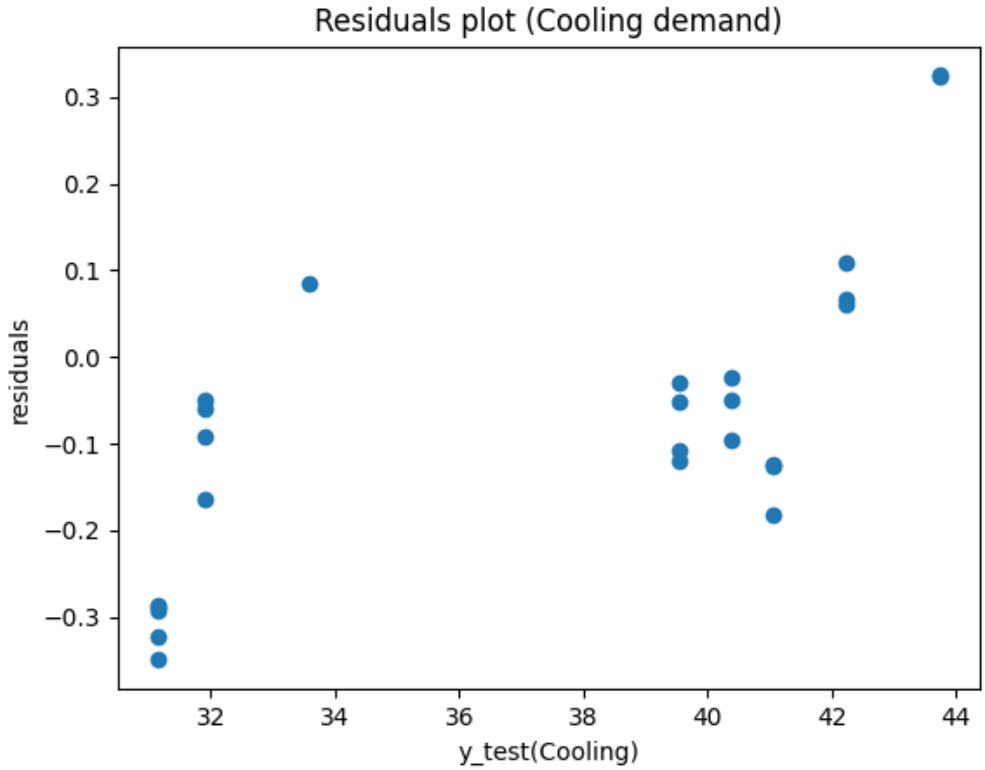


Figure 5.14: Residuals plot for Cooling

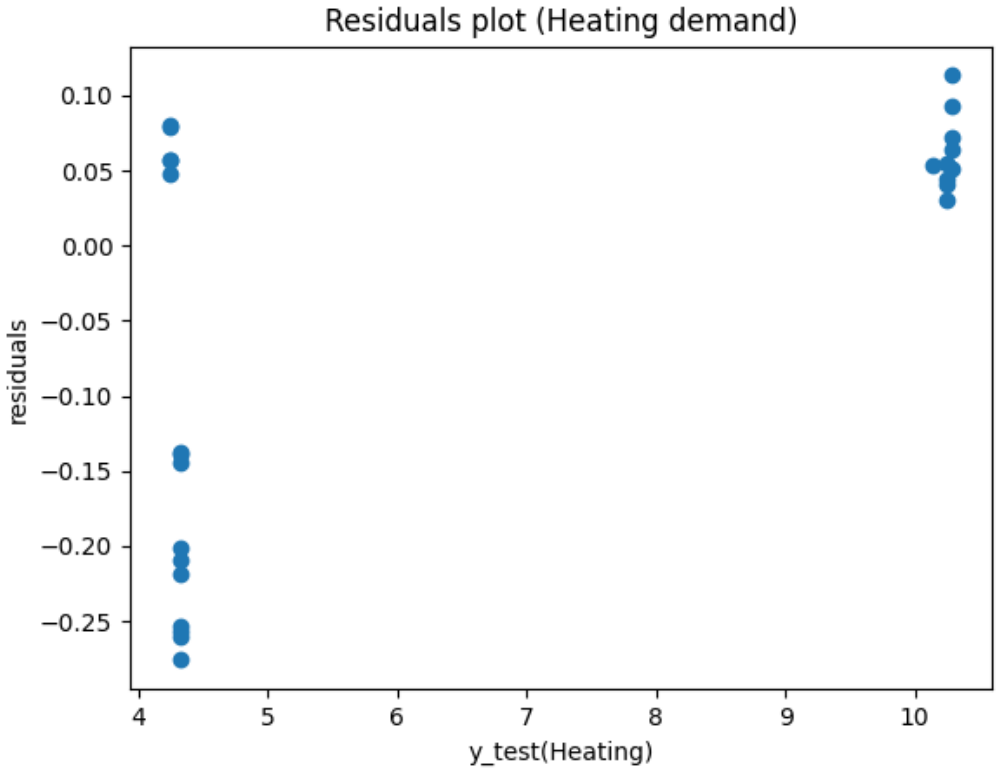


Figure 5.15: Residuals plot for Heating

MSE	MAPE	R2
0.378	0.005	0.998

Table 5.5: Prediction errors

At this point it was possible to carry out the optimization algorithm using the neural network instead of EnergyPlus, always having as reference the 4 parameters and the two outputs described above. In particular, the algorithm (NSGA2 in this case) carried out 10 iterations (`n_generations = 100`) generating five hundred samples each time (`population_size = 500`), for a total of 5000 different simulations analyzed by the algorithm in search of optimal points.

Parameter	Upper value	Lower value
external insulation layers (thickness[m])	0.15	0.02
ach	5	1
overhangs extension[m]	0.95	0.2
U-factor windows	2.2	0.6

Table 5.6: Range parameters (Surrogate optimization)

As in the simple case of Optimization, here too it was necessary to specify the ranges of values within which the parameters could vary.

In total 319 optimum results were found, shown in the figure 5.16.Finally, a parallel cordinates plots equal to that shown in figures 5.17 and 5.18 were generated, but this time reporting the values of the points belonging to the pareto front just calculated.

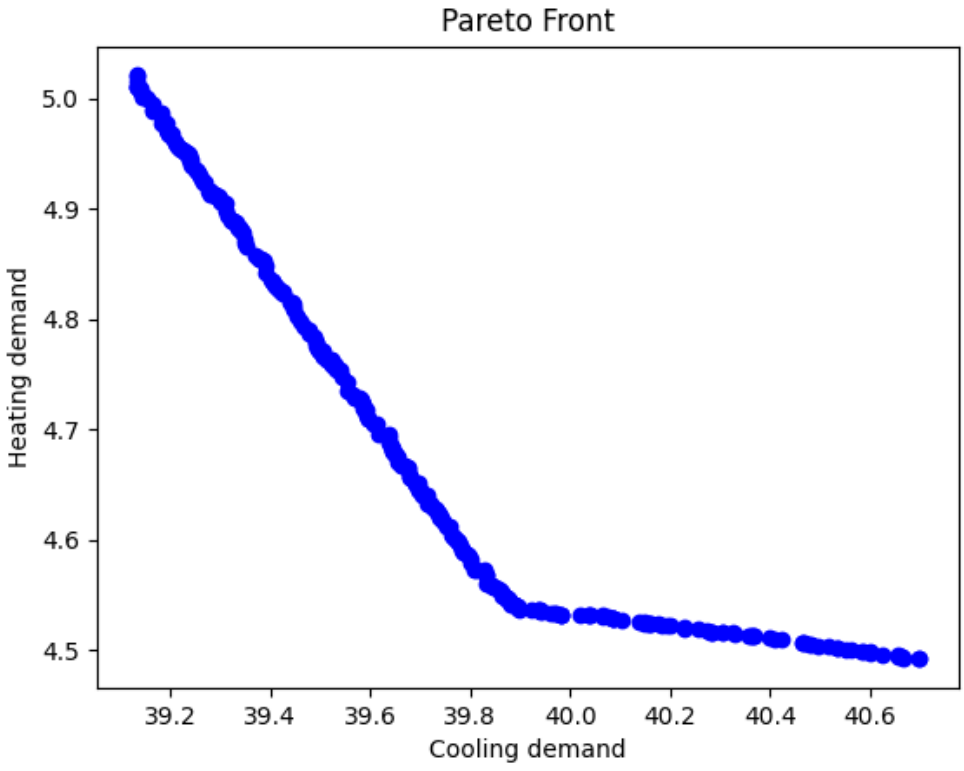


Figure 5.16: Pareto Front with Surrogate model

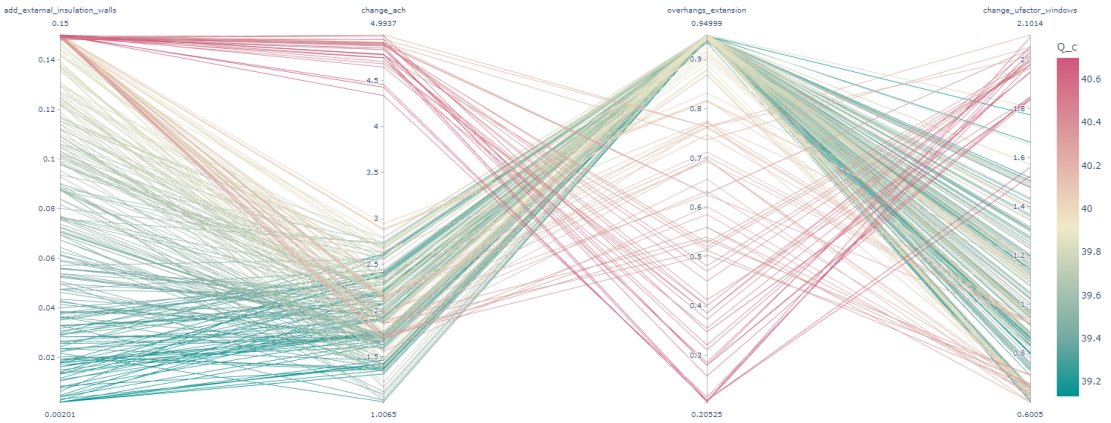


Figure 5.17: Optimization results with Surrogate model (Cooling demand)

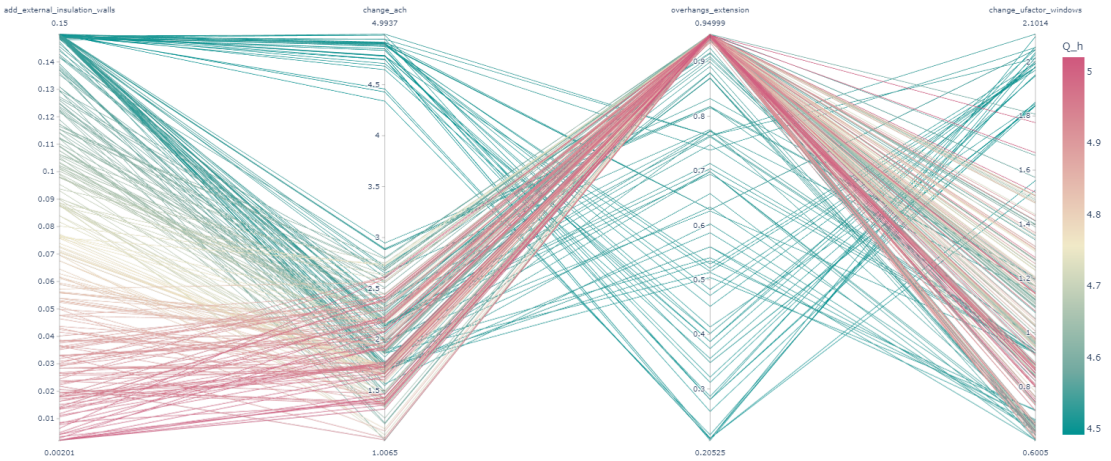


Figure 5.18: Optimization results with Surrogate model (Heating demand)

5.3.1 Results discussion

The first point to emphasize in this example is the high accuracy that the neural network achieves in consumption prediction. As shown in the table 5.5, the errors compared to the values calculated with EnergyPlus are negligible, and this ensures that the neural network (when properly tuned) can be an effective tool for launching an infinite number of "simulations," which can then be used to scan the entire space of the building's possible input parameter configurations in search of the best. Obviously, the quality of these consumptions forecasts is directly related to

the number of simulations that are performed prior to training the network, which are generally reduced in comparison to the number of simulations required for a "white-box" optimization process. With this approach, you can see how the Pareto front graph 5.16 looks much better, and you can also create a parallel cordinates graph to see if the objectives are in conflict with one another. As seen in Figure 5.17 and 5.18, the two graphs are mirrored and have parallel lines that cross over each other, indicating that the parameters chosen have an opposing influence on the two consumptions.

5.4 Example of Surrogate Model with meteorological data

In this scenario, since the emphasis is on developing a neural network that can predict a building's performance based on meterological data, it was necessary to run a single EnergyPlus simulation with predetermined building structural parameters for each of the three meterological files used in the various tests.

To evaluate the script's efficiency, three different scenarios were created in which the same building described in the previous pragraphs is accompanied by three different epw files relating to the locations of Torre Pellice, Aalborg, and Atene (year 2022). For training the network, the file CSV that EnergyPlus returns in output after starting the simulation until the end of the selected period, with daily data frequency, is used. In this particular case, the following measurements were chosen as network input parameters:

1. Outdoor Air Drybulb Temperature [°C][Daily]
2. Diffuse Solar Radiation Rate per Area [W/m2][Daily]
3. Direct Solar Radiation Rate per Area [W/m2][Daily]

Instead, based on output data, the following values were chosen:

1. DistrictCooling:Facility [J](Daily)
2. DistrictHeating:Facility [J](Daily)
3. Electricity:Facility [J](Daily)
4. Zone Mean Radiant Temperature [°C](Daily:ON) (average value calculated between all zones of the building)
5. Zone Thermal Comfort Fanger Model PMV [](Daily) (average value calculated between all zones of the building)

Despite the fact that the neural network can predict all of the objectives at the same time, it was decided to report the model's accuracy graphics while only considering one target per climate zone. The results of the surrogate model are reproduced in this section using the following combinations:

1. Aalborg - PMV (winter simulation)

2. Athens - Zone mean radiant temperature (annual simulation)

3. Torre Pellice - Heating demand (annual simulation)

The network, as described in the methodology chapter, was built by using the file eplusout.csv as a database, with a number of rows equal to the number of days in which the simulation is run. The created database has been divided into two parts, with one part (70 per cent) being used for training and the other (30 per cent) being used as a gauge for the predicted results from the network. The results for the three different cases are followed by graphs and tables that quantify the accuracy of the surrogate models.

Error	Aalborg-PMV	Athens-Radiant Temp.[°C]	Torre Pellice-Cooling[J]
MSE	0.031	0.168	14545849.72
MAPE	0.089	0.006	??
R2	0.784	0.953	0.969

Table 5.7: Errors on test-Database

Index	Parameter
1	Outdoor Temperature [°C][Daily]
2	Diffuse Solar Radiation [W/m2][Daily]
3	Direct Solar RAdiation [W/m2][Daily]
4	Cooling demand[J]
5	Heating demand[J]
6	Electricity demand[J]
7	Fanger model PMV
8	Fanger model PPD
9	Zone mean radiant Temp. [°C]
10	Tot. consumption[J]

Table 5.8: Heatmaps legend

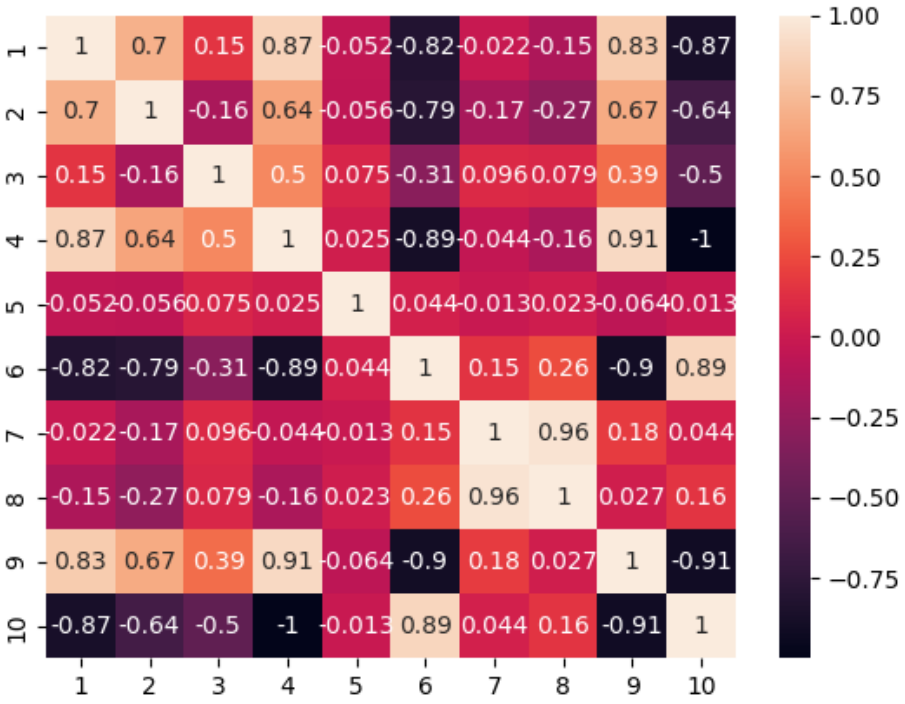


Figure 5.19: Correlation matrix - Aalborg

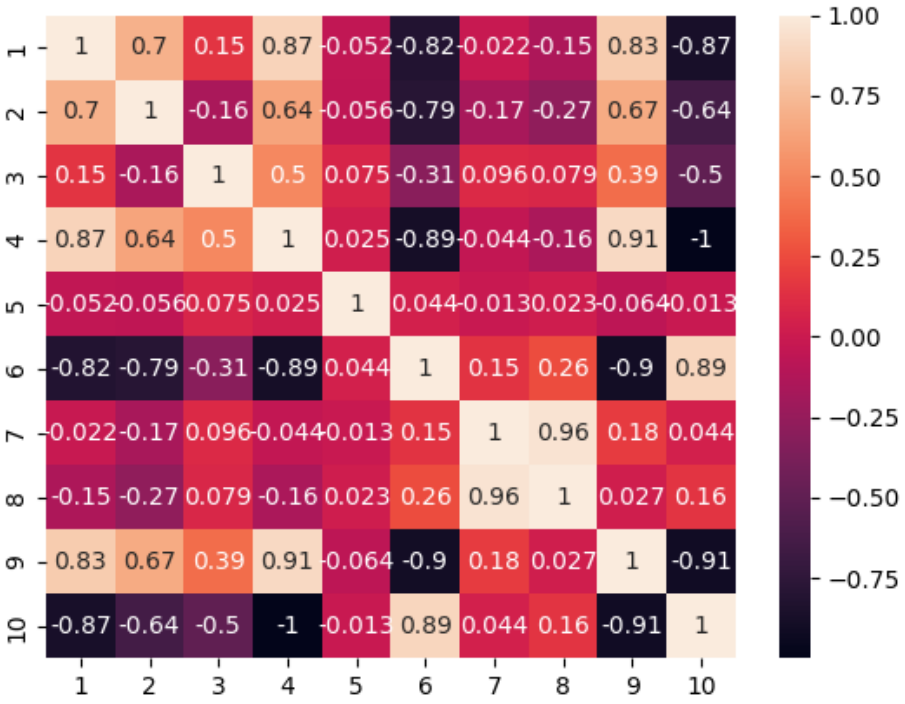


Figure 5.20: Correlation matrix - Athens

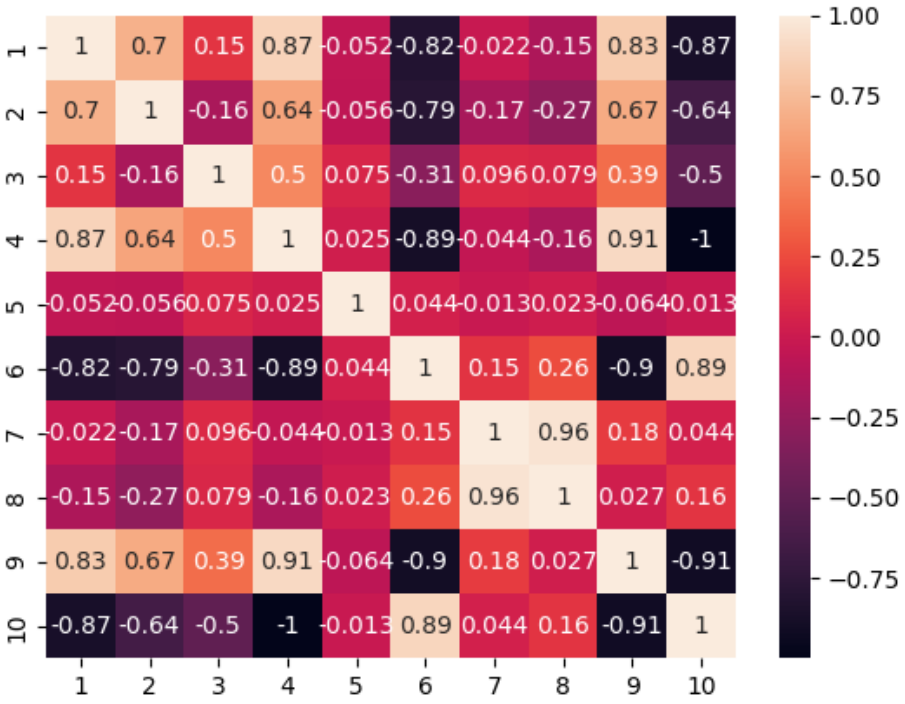


Figure 5.21: Correlation matrix - Torre Pellice

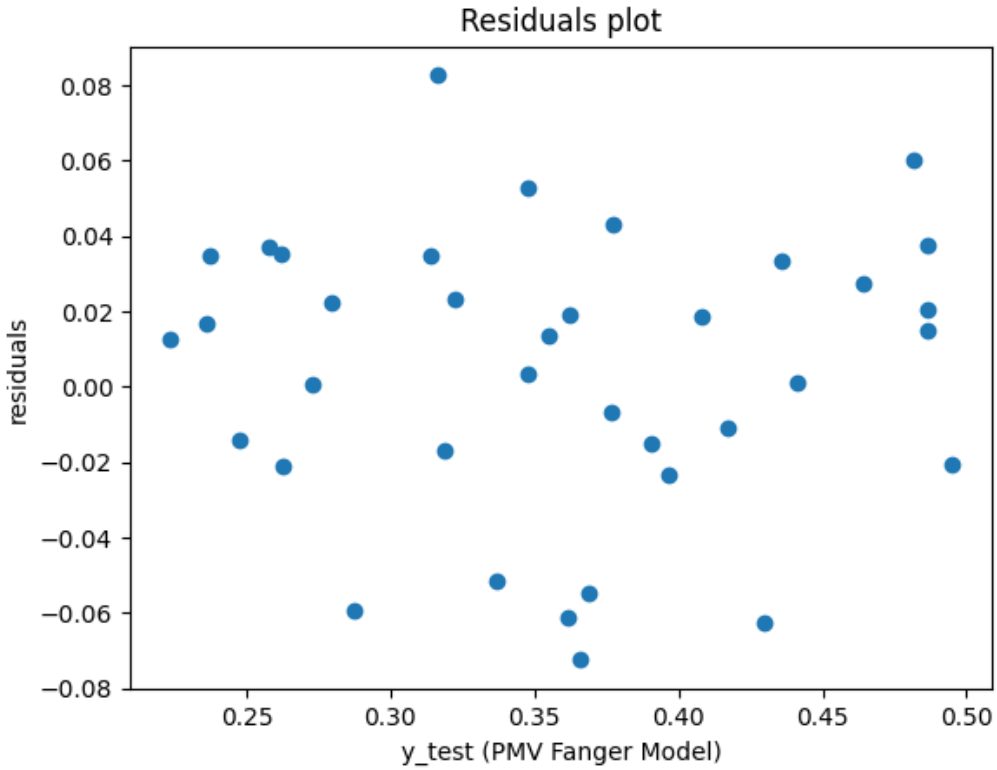


Figure 5.22: Residuals plot Aalborg-PMV

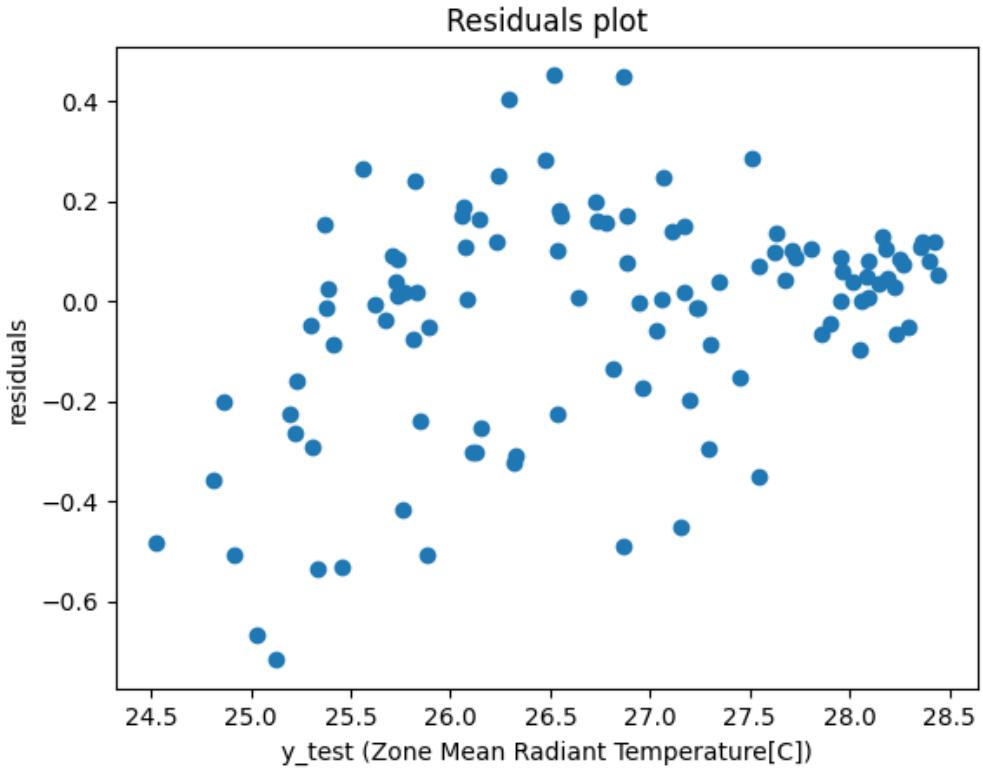


Figure 5.23: Residuals plot Athens-RadiantTemp[°C]

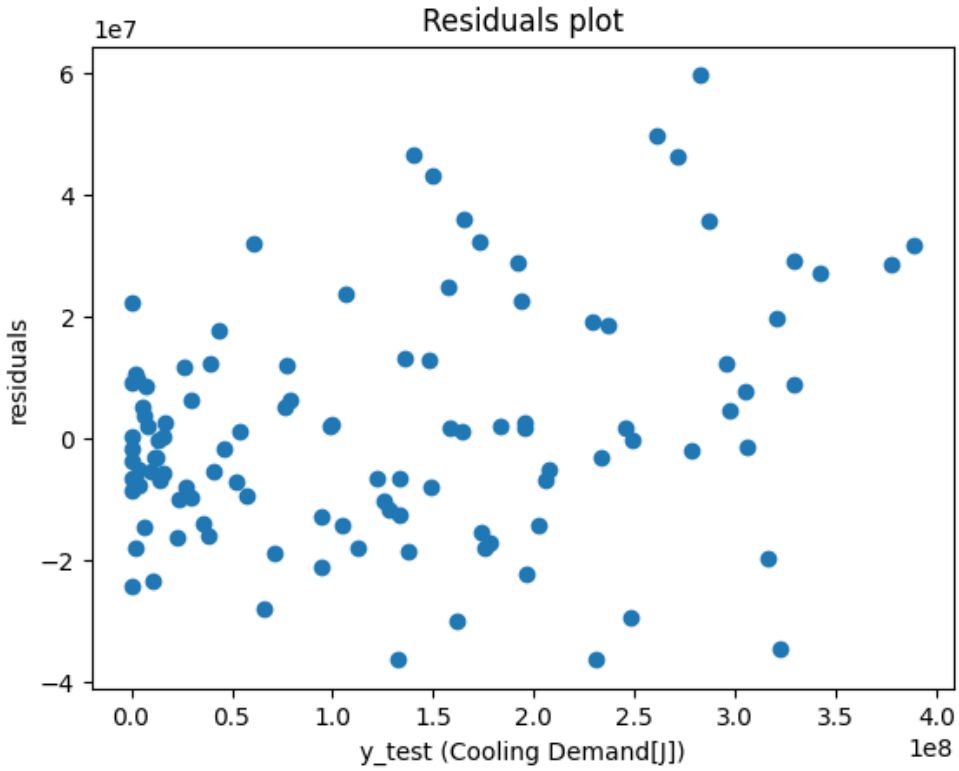


Figure 5.24: Residuals plot Torre Pellice-Cooling[J]

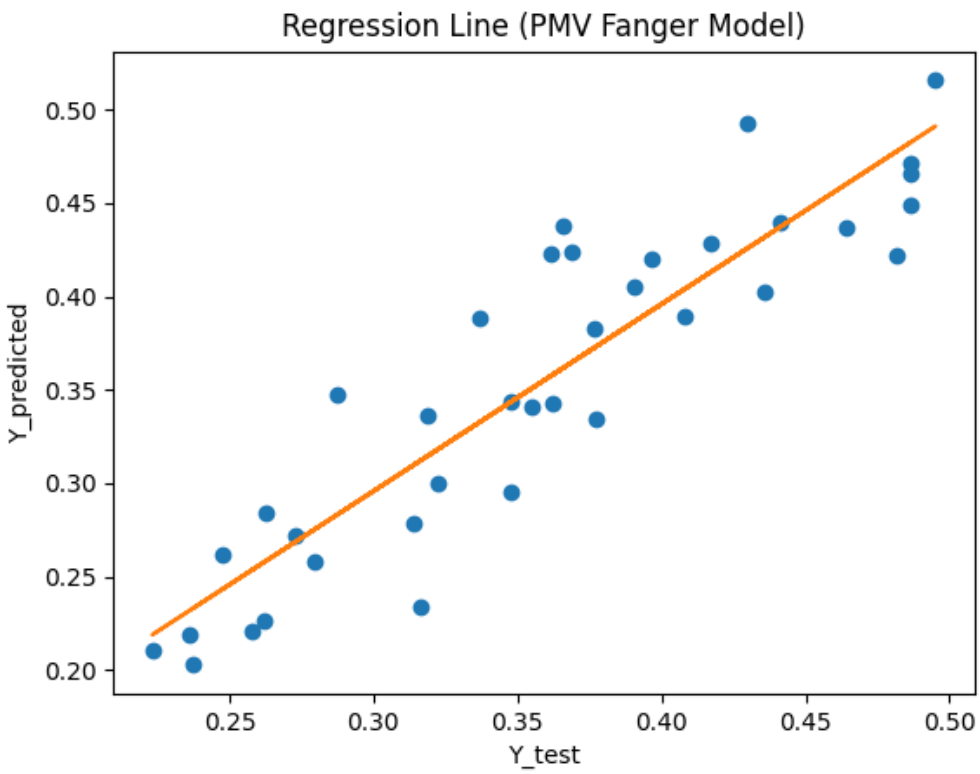


Figure 5.25: Regression line Aalborg-PMV

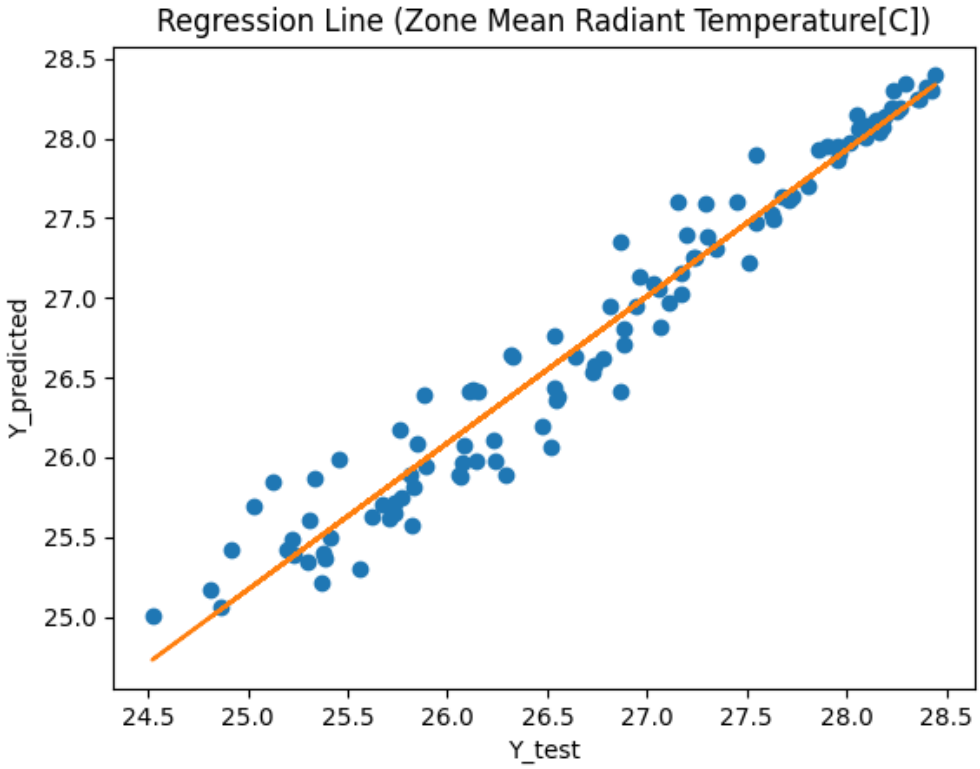


Figure 5.26: Regression line Athens-RadiantTemp[°C]

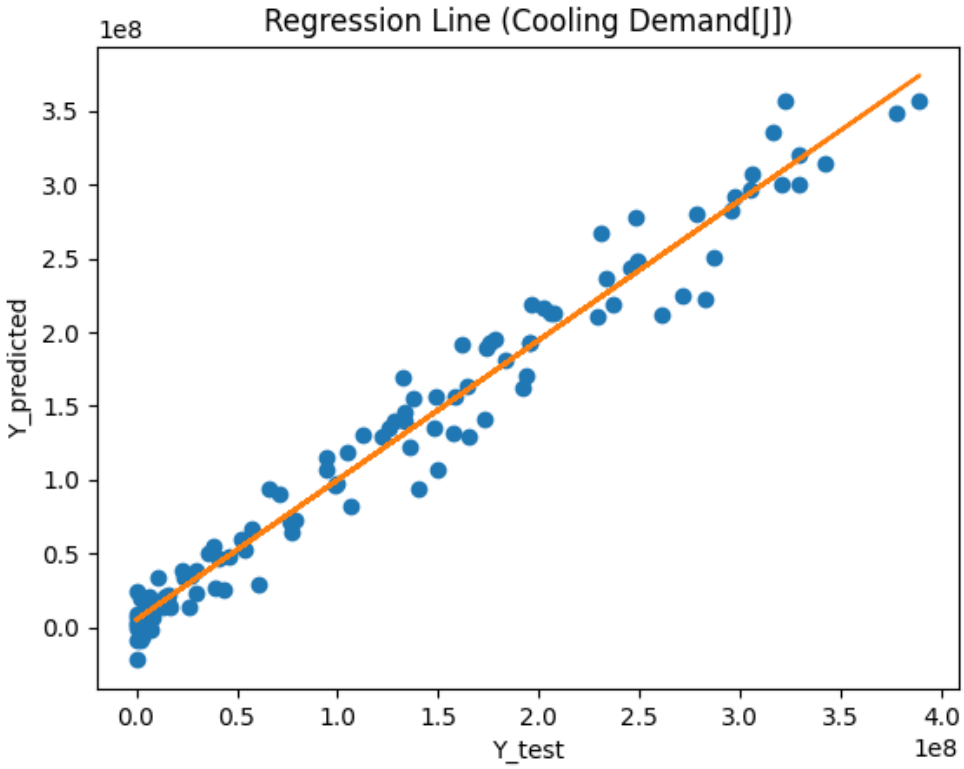


Figure 5.27: Regression line Torre Pellice-Cooling[J]

5.4.1 Results discussion

Just as in the previous case, the neural network trained using input meteorological data also manages to achieve very good accuracy in all three scenarios reported. In particular, it can be seen from the table 5.7 how the annual simulations have r2 values close to 1, while the seasonal Aalborg simulation shows a lower accuracy, due to the smaller number of points with which the network was trained, but still satisfactory. This aspect is also evidenced by the graphs showing the regression line 5.25 5.26 5.27 in which a larger spread of points is evident in the graph for Aalborg-PMV. It should also be pointed out that the results reported were obtained by considering as input only 3 values related to the outdoor environment. Thanks to the correlation matrix shown in Figure 5.20 5.21 5.19 it is possible to investigate what other factors might have a positive correlation with the target values that are intended to be predicted, so as to expand the amount of information in the training matrix, improving performance.

Chapter 6

Conclusions

The examples provided in the previous chapter demonstrate how the new features implemented in the PREDYCE tool produce satisfactory results. Particularly for the optimization task, genetic algorithms have been found to be perfectly suited for completing and refining PREDYCE's sensitivity analysis. The new scripts provide the user with a tool for more clear and immediate visualization of results, allowing them to assess the impact of input parameters on the building's KPIs, and automatically isolate the best parameter configurations for the building in analysis. These functionalities may be very important in the planning phase because they provide the user with support in the selection of specific structural elements (type and thickness of wall insulation layers, types of windows, U-value, ventilation systems, shading, etc.) in order to find the right balance between cost, comfort, and maximizing a building's consumption. The implementation of Surrogate modeling has also demonstrated significant potential since it appears to be capable of surmounting one of the most significant limitations of white-box simulations such as EnergyPlus, specifically calculation complexity and execution time delays. It was possible to explore the whole space of parametric configurations and predict the corresponding results with a high degree of accuracy using machine learning methods and a limited number of simulations. This type of black-box approach, due to its low computational complexity, is ideal for use within the ecosystem of a Smart Building, which, even with limited computational power (as in the RaspBerry Pi), would be able to make accurate predictions in a short period of time by inputting climatic and/or structural parameters and providing solutions to optimize comfort and consumption in real time within the building.

Bibliography

[1] *EIA projects nearly 50 per cent increase in world energy usage by 2050, led by growth in Asia*. https://www.eia.gov/todayinenergy/detail.php?id=42342. [Online; accessed 2022-10-30]. Oct. 2022 (cit. on p. 2).

[2] . *Working Toward the Very Low Energy Consumption Building of the Future - Berkeley Lab*. https://newscenter.lbl.gov/2009/06/02/working-toward-the-very-low-energy-consumption-building-of-the-future/. [Online; accessed 2022-10-28]. June 2009 (cit. on p. 3).

[3] *BMS*. https://intelik.eu/sample-page/building-management-system-bms/. [Online; accessed 2022-10-28] (cit. on p. 4).

[4] Lev Kalmykov and Vyacheslav Kalmykov. «A white-box model of S-shaped and double S-shaped single-species population growth». In: *PeerJ* 3 (May 2015), e948. DOI: 10.7717/peerj.948 (cit. on p. 6).

[5] U.S. Department of Energy's (DOE) Building Technologies Office (BTO). «EnergyPlus». In: (2022). URL: https://energyplus.net/ (cit. on p. 7).

[6] Germán Campos Gordillo, Germán Ramos Ruiz, Yves Stauffer, Stephan Dasen, and Carlos Fernández Bandera. «EplusLauncher: An API to Perform Complex EnergyPlus Simulations in MATLAB® and C». In: *Sustainability* 12.2 (2020). ISSN: 2071-1050. DOI: 10.3390/su12020672. URL: https://www.mdpi.com/2071-1050/12/2/672 (cit. on p. 8).

[7] . *Home - Prelude project*. https://prelude-project.eu/. [Online; accessed 2022-11-14]. May 2022 (cit. on p. 9).

[8] *homepage - E-DYCE*. https://edyce.eu/. [Online; accessed 2022-10-26]. Sept. 2022 (cit. on p. 9).

[9] «https://www.designbuilderitalia.it/». In: () (cit. on p. 10).

[10] Danny Lobos, Gerth Wandersleben, and Lorena Castillo. «Interoperability Map between BIM and BPS Software». In: June 2014, pp. 601–608. DOI: 10.1061/9780784413616.075 (cit. on p. 11).

[11] *OpenStudio*. [Online; accessed 2022-10-26]. URL: https://openstudio.net/ (cit. on p. 11).

[12] *Radiance*. https://floyd.lbl.gov/radiance/. [Online; accessed 2022-10-26] (cit. on p. 11).

[13] *3D Design Software | 3D Modeling on the Web | SketchUp*. [Online; accessed 2022-10-26]. URL: https://www.sketchup.com/ (cit. on p. 11).

[14] *OpenStudio 2.9.1 Download - ArchSupply.com*. [Online; accessed 2022-10-28]. URL: https://download.archsupply.com/get/download-openstudio/ (cit. on p. 12).

[15] *BESOS platform*. https://besos.uvic.ca/. [Online; accessed 2022-10-26] (cit. on p. 12).

[16] *Welcome to eppy's documentation! - eppy 0.5.59 documentation*. [Online; accessed 2022-10-26]. URL: https://eppy.readthedocs.io/en/latest/index.html (cit. on p. 12).

[17] Fasano Francesca Chiesa Giacomo and Grasso Paolo. «"A new Tool for Building Energy Optimization: First Round of Succesful Dynamic Model Simulations" Energies 14, no. 19: 6429.» In: (2021). URL: https://doi.org/10.3390/en14196429 (cit. on pp. 12, 13, 29).

[18] Fasano Francesca Chiesa Giacomo and Grasso Paolo. «Simulated Versus MOnitored Building Behaviours: Sample Demo Applications of a Performance Gapp Detection Tool in a Northern Italian Climate.» In: *Sayigh, A. (eds) Towards Net Zero Carbon Emissions in the Building Industry. Innovative Renewable Energy. Springer, Cham.* (2023). URL: https://doi.org/10.1007/978-3-031-15218-4_6 (cit. on p. 12).

[19] Fasano Francesca Chiesa Giacomo and Grasso Paolo. «Thermal Comfort anc Climatic Potential of Ventilative Cooling in Italian Climates.» In: *Sayigh, A. (eds) Achieving Building Comfort by Natural Means. Innovative Renewable Energy. Springer, Cham.* (2022). URL: https://doi.org/10.1007/978-3-031-04714-5_18 (cit. on p. 12).

[20] G. Chiesa. «Energy flexible DYnamic building CErtification.» In: (). URL: https://edyce.eu/wp-content/uploads/2022/03/E-DYCE_D3.1_Dynamic-simulation-platform_28.02.2022_Final.pdf (cit. on p. 14).

[21] S. Panzieri (Univ. Roma Tre) F. Moretti (ENEA). «METODOLOGIA DI OTTIMIZZAZIONE MULTI-OBIETTIVO DELLA CLIMATIZZAZIONE TERMICA DI EDIFICI. VALIDAZIONE SU SISTEMA DI SIMULAZIONE.» In: (2013) (cit. on p. 15).

[22] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. «A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II». In: *Parallel Problem Solving from Nature PPSN VI*. Ed. by Marc Schoenauer, Kalyanmoy Deb, Günther Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 849–858. ISBN: 978-3-540-45356-7 (cit. on p. 16).

[23] Kamel Goudjil and Badreddine Sbartai. «Optimization of Shear Wave Velocity (Vs) from a Post-Liquefaction Settlement Using a Genetic Algorithm Multi-Objective NSGA II». In: *WSEAS Transactions on Applied and Theoretical Mechanics* 18 (July 2018) (cit. on p. 17).

[24] Choi and Kim. «Self-Adaptive Models for Water Distribution System Design Using Single-/Multi-Objective Optimization Approaches». In: *Water* 11 (June 2019), p. 1293. DOI: 10.3390/w11061293 (cit. on p. 18).

[25] «A multi-objective test data generation approach for mutation testing of feature models - Scientific Figure on ResearchGate.» In: (). URL: %22https://www.researchgate.net/figure/Pseudocode-of-NSGA-II-adapted-from-Coello-et-al-2006_fig7_305662662 (cit. on p. 19).

[26] «Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems | SIAM Journal on Optimization». In: *SIAM Journal on Optimization* (Jan. 1998) (cit. on p. 19).

[27] *pymoo - NSGA-III*. https://pymoo.org/algorithms/moo/nsga3.html. [Online; accessed 2022-10-30] (cit. on p. 20).

[28] Shuai Guo. «An introduction to Surrogate modeling, Part I: fundamentals». In: (2020). URL: https://towardsdatascience.com/an-introduction-to-surrogate-modeling-part-i-fundamentals-84697ce4d241 (cit. on p. 21).

[29] Wikipedia. «Artificial neural network». In: (). URL: https://en.wikipedia.org/wiki/Artificial_neural_network (cit. on p. 22).

[30] scikit-learn developers (BSD License). «Neural network models (supervised)». In: (). URL: https://scikitlearn.org/stable/modules/neural_networks_supervised.html (cit. on p. 22).

[31] Julian Blank and Kalyanmoy Deb. «Pymoo: Multi-Objective Optimization in Python». In: *IEEE Access* 8 (2020), pp. 89497–89509. DOI: 10.1109/ACCESS.2020.2990567 (cit. on p. 25).

[32] *pickle — Python object serialization — Python 3.11.0 documentation*. [Online; accessed 2022-11-26]. URL: https://docs.python.org/3/library/pickle.html (cit. on p. 32).

[33] *NumPy Documentation*. https://numpy.org/doc/. [Online; accessed 2022-11-25] (cit. on p. 33).