Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інтеграція інструментів моделювання з секторів електроенергетики, мобільності та опалення в систему моніторингу потоків енергії, керовану подіями»

Виконав: студент 2 курсу, групи 1АКІТР-24м спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва спеціальності)
Марія ФОРКАЛЮК
(ПІБ студента)

Керівник: д.т.н., професор кафедри АІІТ Олег БІСІКАЛО
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«10» грудня 2025 р.

Опонент: д.т.н., проф. каф. КСУ В'ячеслав КОВТУН
(науковий ступінь, вчене звання / посада, ПІБ опонента)
«10» грудня 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)
«10» грудня 2025 р.

Вінниця ВНТУ – 2025 рік

Vinnytsia national technical university
Faculty of intelligent information technologies and automation
Department of automation and intelligent information technologies

MASTER'S QUALIFICATION PAPER

to the topic:

«Integration of modelling tools from electricity, mobility and heating sectors into an event-driven energy-flow monitoring framework»

Perfomed by: student of 2nd course, group 1АКІТР-24м of speciality 174 – Automation, computer-integrated technologies and robotics
(code and name of specialty)

Mariia FORKALIUK
(full name of student)

Supervisor: PhD, professor of AIIT department
Oleh Bisikalo
(scientific degree, title / position, full name of supervisor)

«10th» December 2025 y.

Opponent: PhD., professor of CCS department
Vyacheslav KOVTUN
(scientific degree, title / position, full name of opponent)

«10th» December 2025 y.

Accepted for defence
Head of AIIT department
PhD., prof. Oleh BISIKALO
(scientific degree, title)
«10th» December 2025 y.

Vinnytsia VNTU – 2025 year

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – Електроніка, автоматизація та електронні комунікації

Спеціальність – 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Освітньо-професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«26» вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Форкалюк Марії Сергіївні
(ПІБ автора повністю)

1. Тема роботи: Інтеграція інструментів моделювання з секторів електроенергетики, мобільності та опалення в систему моніторингу потоків енергії, керовану подіями.
   Керівник роботи: д.т.н., професор каф. АІІТ Бісікало О. В.
   Затверджені наказом ВНТУ від «24» вересня 2025 року № 313.
2. Строк подання роботи студентом: до «10» грудня 2025 року.
3. Вихідні дані до роботи: Три будинки, з'єднані між собою мережею електропостачання та керовані інтелектуальними алгоритмами, визначено за об'єкт моделювання. З них два будинки є житловими будівлями з 2/1 електромобілями, 15/5 кВт фотоелектричними системами та 5/4 кВт системами опалення, вентиляції та кондиціонування повітря. Третій будинок є офісною будівлею з 12 електромобілями, 50 кВт фотоелектричними системами та 50 кВт системами опалення, вентиляції та кондиціонування повітря. Потрібно оцінити взаємозв'язок енергетичних потоків будівлі.
4. Зміст текстової частини: Вступ; Стан проблеми та огляд аналогічних робіт; Інтеграція моделей та підхід до спільного моделювання; Розробка програмного забезпечення; Приклад оцінки багатосекторіального сценарію; Економічний розділ; Висновки; Список використаної літератури.
5. Перелік ілюстративного (або графічного) матеріалу: Архітектура симуляційної платформи; Діаграма послідовності управління симуляцією; Діаграма класів; Графічний інтерфейс користувача; Результати симуляції для 3 різних сезонів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	Геральд ФРАНЦЛ, к.т.н., старший науковий співробітник кафедри інтегрованих сенсорних систем університету безперервної освіти Кремс, Австрія	25.09.2025	05.12.2025
5	Володимир КОЗЛОВСЬКИЙ, к.е.н., проф. каф. ЕПтаВМ	04.11.2025	01.12.2025

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	25.09 – 05.10.2025	Виконано
2	Вибір інструментів для інтеграції та проектування API та інтерфейсів	05.10 – 15.10.2025	Виконано
3	Проектування та розробка графічного інтерфейсу користувача	15.10 – 31.10.2025	Виконано
4	Перевірка та доопрацювання інтеграції інструментів	01.11 – 10.11.2025	Виконано
5	Виконання зразкових завдань моделювання	10.11 – 20.11.2025	Виконано
6	Підготовка економічної частини	до 01.12.2025	Виконано
7	Підготовка пояснювальної записки, графічних матеріалів і презентації	20.10 – 28.11.2025	Виконано
8	Попередній захист магістерської роботи	до 03.12.2025	Виконано
9	Захист магістерської роботи	16.12.2025	Виконано

Студент _____ Марія ФОРКАЛЮК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Олег БІСІКАЛО
(підпис) (прізвище та ініціали)

Vinnytsia national technical university
Faculty of intelligent information technologies and automation
Department of automation and intelligent information technologies
Degree of higher education second (master's)
Branch of knowledge – Electronics, automation and electronic communication
Speciality – 174 - Automation, computer-integrated technologies and robotics
Education program – Intelligent computer systems

APPROVE
Head of AIIT department
PhD, professor Oleh BISIKALO
«26th» September 2025 p.

ASSIGNMENT
FOR MASTER'S THESIS

Forkaliuk Mariia
(full name of student)

1. Topic of work: Integration of modelling tools from electricity, mobility and heating sectors into an event-driven energy-flow monitoring framework.
   Supervisor: PhD professor of AIIT department Oleh Bisikalo
   Approved by higher educational institution's order since 12th September 2025, № 313.
2. Work deadline of submission: 10th December 2025
3. Work input data: Exemplary three houses connected to the same feeder network and managed by smart algorithms have been selected as the modeling object shall be simulated integrating existing modeling tools. Two houses are residential buildings with 2/1 EVs and 15/5 kWp PV and 5/4 kW HVAC. The third is an office building with 12 EVs, 50 kWp PV and 50 kW HVAC. The correlation of building's energy flows shall be evaluated.
4. Contents of the text part: Introduction; State of problem and review of analogous works; Model integration and co-simulation approach; Software development; Exemplary multi-sector scenario evaluation; The economic section; Conclusions; List of referenced literature.
5. List of illustrative (or graphic) material: The simulation framework's architecture; Simulation Control sequence diagram; Class diagram of the simulation framework modules; Graphical user interface of the developed cross-sector simulation framework, Simulation results for three different seasons.

6. Consultans of sections of the work

Section	Surname, initials and position of consultant	Signature, data	
		Task issued	Task accepted
1 – 4	Gerald FRANZL, PhD, Senior Researcher at the Department of Integrated Sensor Systems of University for Continuing Education Krems, Austria	25.09.2025	05.12.2025
5	Volodymyr KOZLOVSKYI, PhD, associate professor of BEPM department	04.11.2025	01.12.2025

7. Issue date of assignment: 25th September 2025.

CALENDAR PLAN

№	Name of the stages of the master's qualification work	Accomplishment terms	Note
1	Analysis of the subject area	25.09 – 05.10.2025	Done
2	Chose the tools to integrate and design APIs and interfaces	05.10 – 15.10.2025	Done
3	Design and program the GUI	15.10 – 31.10.2025	Done
4	Test and revise the integration of tools	01.11 – 10.11.2025	Done
5	Perform exemplary simulation tasks	10.11 – 20.11.2025	Done
6	Prepare the economic part	till 01.12.2025	Done
7	Prepare explanatory notes, graphic materials and presentation	20.11 – 28.11.2025	Done
8	Preliminary defense of master's thesis	till 03.12.2025	Done
9	Defense of the master's thesis	16.12.2025	Done

Student _____ Mariia FORKALIUK
(signature) (surname, initials)

Supervisor _____ Oleh BISIKALO
(signature) (surname, initials)

АНОТАЦІЯ

УДК: 681.5 : 519.876.5

Форкалюк М. С. Інтеграція інструментів моделювання з секторів електроенергетики, мобільності та опалення в систему моніторингу потоків енергії, керовану подіями. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2025. 157 с.

На англ. мові. Бібліогр.: 50 назви; рис 58, таблиць 16.

Метою роботи є об'єднання сучасних інструментів моделювання з різних секторів, зокрема електроенергетики, теплопостачання та охолодження, а також мобільності, за допомогою подієво-орієнтованого ядра симуляції у єдину ко-симуляційну платформу, що охоплює взаємозв'язки між системами різних секторів. Це дозволить експертам і інженерам тестувати нові підходи до мікро-балансування та пом'якшення волатильності шляхом часткового коригування попиту відповідно до мінливої та розподіленої енергоподачі від сонця і вітру. Особливий інтерес становить виявлення питань, пов'язаних з юзабіліті, впливом, побічними ефектами та кореляціями, а також виявлення суперечливих і протидіючих підходів та практик. Заплановані сценарії охоплюють як будівлі із спільним виробництвом та зберіганням електроенергії, а також спільним теплопостачанням та охолодженням, так і квартали, з'єднані спільною системою розподілу тепла та спільно керованими точками зарядки електромобілів.

Ключові слова: спільне моделювання на основі подій, міжгалузева інтеграція, управління енергетичними потоками, координація попиту на електроенергію, мікробалансування, розумні мережі.

ANNOTATION

Mariia Forkaliuk. Integration of modelling tools from electricity, mobility and heating sectors into an event-driven energy-flow monitoring framework. Master's qualification paper of specialty 174 – Automation, computer-integrated technologies and robotics, curricula – Intelligent computer systems. Vinnytsia: VNTU, 2025. 157 pages.

In English language. Bibliography: 50 titles; fig.: 58; tabl.: 16.

The purpose of the work is to link existing state-of-the-art simulation tools from different sectors, in particular electricity, heating and cooling, and mobility, by an event-driven simulation core into a co-simulation framework that covers the inter-relations of the systems from different sectors. This shall enable experts and engineers to test novel approaches toward micro-balancing and volatility mitigation by partially adjusting the demand to the volatile and distributed energy supply from sun and wind. Of particular interest is identifying issues related to usability, impact, side-effects and correlations, as well as revealing conflicting and counteracting approaches and practices. The intended scenarios shall extend from buildings with shared electricity generation and storage, as well as shared heating and cooling, to neighbourhoods connected by a shared heat-distribution system and jointly managed EV charging points.

Key words: Event-Based Co-Simulation, Cross-Sector Integration, Energy-Flow Management, Electricity-Demand Coordination, Micro-Balancing, Smart Grid.

ABBREVIATIONS, UNITS AND VARIABLES

Abbreviations

AC — alternating current
ASHP — air source heat pump
BEMS — building energy management systrem
BEV — battery electric vehicle
CAPEX — capital expenditure
CCDF — complementary cumulative distribution function
CDH — cold district heating
CHP — combined heat and power system
CP — charging point
CS — charging site
DC — direct current
DSF — demand side flexibility
EFM — energy flow management
EH — energy hub
EMS — energy management system
EMT — energy modelling tool
ESS — energy storage system
EV — electric vehicle
EVSE — electric vehicle supply equipment
FMI — functional mock-up interface
FO — flexibility offer
HEMS home energy management system
HIL — hardware-in-the-loop (testing)
HP — heat pump
HPC — high-performance computing
HV — high voltage (grid level)

HVAC — heating, ventilation and air conditioning

IoT — Internet of things

LNN — liquid neural network

LV — low voltage (grid level)

MES — multi-energy system

MV — medium voltage (grid level)

OPEX — operating expense

PI — proportional-integral (control)

PID — proportional-integral-derivative (control)

PV — photovoltaic (electricity generation system)

RC — resistance/capacitance module

REC — renewable energy community

RED — renewable energy directive

RGB — red green blue

RNN — recurrent neural network

SoC — state-of-charge

TLS — traffic light system

V2G — vehicle to grid discharging

Units

$^{\circ}C, K$ — grad Celsius, Kelvin (temperature)

h — hour (time)

Ω/km — Ohm per kilometre cable length (resistance)

V, kV — Volt, kilo-Volt (power potential)

W, Wh, — Watt (power), Watt-hour (energy)

TABLE OF CONTENTS

INTRODUCTION

Relevance of the Work. The transition to renewable energy sources, enforced to reduce fossil CO_2 emissions [1] and dependency on imports from unreliable countries [2], introduces huge amounts of volatile energy generation from sunlight and wind [3]. Maintaining a reliable energy supply from volatile sources requires either massive over-provisioning of generation capacities or huge energy storage capacities to compensate the intermittency and volatility of energy generation from sunlight and wind [3, 4]. Over-provisioning and energy buffering are costly and shall be minimised in whatever combination they become realised. The best way to reduce the demand for volatility compensation, is to reduce the energy consumption when renewable supply is scarce [5]. Cross-sector energy management can support that with marginal impact on customer comfort [6].

Purpose of the work. Compose, extend and apply a simulation framework, a collection of linked software tools, to analyse local cross-sector energy balancing strategies. The resultant tool shall be able to model integrated provisioning of electricity, heating and cooling, and mobility (EV charging). It shall speed up the evaluation of control algorithms by possibly 50 % and save the related personnel costs. If iterating through the tools used stand-alone costs 10 kUAH, than using the developed cross-sector simulation framework saves 5 kUAH.

Object of study. Novel cross-sector energy management that supports the energy transition toward 100 % renewable energy sources shall be evaluated, in particular strategies for local energy balancing among heating, EV charging and PV production. The intended scenario shall extend from buildings with electricity generation and storage as well as heating and cooling, to neighbourhoods connected by a shared heat-distribution system and jointly managed EV charging points, as enabled in renewable energy communities (REC [7]).

Subject of study. The developed simulation framework shall as far as possible integrate existing tools to analyse:

- Electric load management,
- EV charging management,

- Heating and cooling management system,
- Cross sector energy demand adjustment signalling.

The following problems should be solved to achieve a goal. (**Research tasks.**) The interfaces and the interoperation of the above listed sector specific tools shall be identified, designed and integrated to connect the different tools. The interoperation performance will be evaluated and documented. To achieve both, the following task shall be performed:

1. Find candidates for integration by literature survey and tool trials
2. Analyse possibilities to link different tools, integration feasibility
3. Select fitting tools and develop the simulation framework architecture
4. Design APIs, message parsers, and wrappers needed for the integration
5. Model assets and appliances as well as user behaviours stochastically
6. Define, monitor, log, and save key parameters for stochastic analysis
7. Processed monitored parameters for appealing visualisation and saving
8. Analyse the simulation performance identify and fix shortcomings
9. Approve scalability and adaptability of the developed simulation tool

The overarching research task is to evaluate the implications of integrating tools that were developed independently and intended to be used on their own. In addition thereto, but not the prime task, is exemplary analysing of the gained energy management and shifting options across the connected sectors.

The practical tasks cover system-of-systems modelling, simulation environment design and integration, fixing interoperation issues, and the interpretation and analysis of exemplary simulation results.

Research methods. The research methods applied in the course of this master work include: literature survey, tool comparison and selection, software tools integration, design of energy management and distributed control schemes, stochastic modelling and system behaviour evaluation, fragmentation and simplification of complex systems-of-systems, model based simulation experiments, conclusive interpretation and summarising of simulation results.

Scientific novelty. For the first time, an event-driven cross-sector modelling of smart energy flow management related to systems from different sectors has

been developed. It integrates power-flow calculation, electric vehicle charging demand, photovoltaic power generation, as well as heating, ventilation and air conditioning needs. This is for example needed to coordinate for example the demands of a complex of residential and office buildings, or across a renewable energy community (REC [7]).

Applying smart algorithms ensures a reduction in the dynamic deviation from normal voltage at the local electricity supply connection point. How effective, depends on the scenario. The performed experimental simulation study of three automation objects presented in Section 4.2.3, being three buildings connected by the local feeder network, each with smart controlled PV system, EV charging, and heating and cooling, showed an absolute reduction in deviation from normal voltage from 2.15 to 6.1 % for the winter period, from 1.54 to 7.12 % for spring, and from 2.64 to 7.97 % for summer. The largest improvements achieves the building at the far end of the feeder line.

Cross-sector integration on the energy management level is today rather non-existent or in an infant state [8]. The *energy efficiency* focus [9] has improved the efficiency of individual appliances within their domain. However, a top down view on efficiency reveals the need for bottom up system operation flexibility [10].

Scientific and technical result of the work. Aside from the lessons learned (a) on designing and implementing a simulation architecture integrating several tools, (b) on the use of distributions to model stochastic real world processes, and (c) on proper visualisation of statistical results in publishable quality with scientific rigour, the results of the conducted research work are:

1. A simulation framework to evaluate cross-sector energy management.
2. Cross-sector simulation addressing multi-dimensional opportunities.
3. Insights into the dynamic behaviour of interlinked processes.
4. Examples that reveal potential pitfalls and saving opportunities.
5. A framework that may be reused and extended for future studies.

Practical value of the work. The developed simulation framework will be used in a living lab, the Austrian public funded R&D project *Reallabor Waldviertel* [11], to encourage clients and rural municipalities to invest in regional,

sustainable, sector overarching energy sharing systems. The developed simulation framework shall indirectly proof the value of healthy flexibility over extreme efficiency for controlable energy appliances, in particular: electric room heating and heat pumps in general, ventilation and air conditioning, hot water and steam provisioning, and EV charging.

Approbation and publications. The main objectives of the simulation environment was first published in spring 2025, in the LIV All-Ukrainian Scientific and Technical Conference of the Faculty of Intelligent Information Technologies and Automation [12], after the tools to be integrated had been preliminary selected. Simulation results obtained using the developed simulation framework will be submitted for publication in international conferences, and maybe an international scientific journal, once the framework is operational.

Meanwhile, results gained using the simulation developed in the bachelor work [13] were published at international conferences. First the *16th EAI International Conference on Simulation Tools and Techniques* (EAI SIMUTools 2024), December 9–10, 2024, in Bratislava, Slovakia [14], second the *30th IEEE International Conference on Emerging Technologies and Factory Automation* (IEEE ETFA 2025), September 9–12, 2025, in Porto, Portugal [15], where I presented the paper *Modelling and Simulation of Smart Electric Vehicle Charging Sites*.

A conference paper that addresses the likelihood of flexibility offer fulfilment is presented at the IEEE PES ISGT EUROPE 2025 (*Innovative Smart Grid Technologies* conference), October 20–23, 2025, in Valletta, Malta [16]. A conference paper that resulted from learning to use *pandapower* has been accepted for publication at the *17th International Conference on Applied Energy* (ICAE2025), December 8–12, 2025, at the United Nations Conference Centre in Bangkok, Thailand [17].

1 STATE OF PROBLEM AND REVIEW OF ANALOGOUS WORK

The energy transition has been discussed and investigated for decades, and is now overdue. As a consequence, the European Commission lastly enabled the customers to become active and independent of the established energy business [7, 18]. Organised in so called Energy Communities, customers are now allowed to share and trade self-produced renewable energy independent of the energy market. In the case of Renewable Energy Communities (RECs) specified in [7], this freedom covers not only electricity but any form of renewable energy.

Existing and newly developed software tools shall be joined into a simulation framework to enable the evaluation of the interdependent operation of energy systems from different sectors, here electricity, heat and mobility, for this novel option to combine electricity and heat sharing within RECs.

Being one of the steepest growing source for increased electricity demand, the charging of battery electric vehicles (BEVs) is considered as demand side flexibility (DSF) for the joint energy management. Accordingly, simulating Smart Charging options in a holistic energy flow management (EFM) constitute a core part of the developed simulation framework.

The simulation framework shall enable the clients using it to show that smart assets from different sectors can overcome their efficiency disadvantage if wisely operated across sector boundaries.

1.1 Review of existing tools

First, tools that might be integrated into the simulation framework are outlined in the following subsections. Than, solutions that address joint energy management across sectors are discussed. Finally, the selection of tools intended for the simulation framework is sketched.

1.1.1 ***pandapower*** – a popular open source electricity grid analyser

The pandapower tool [19] is based on the data analysis library *pandas* [20], which enables efficient processing of large datasets. It supports the widely used MATPOWER / PYPOWER case format [21, Appendix B] and [22] respectively, which serves as a standard for many existing grid models. In addition, it allows the use of different solvers: by default an improved Newton-Raphson power flow implementation, all PYPOWER solvers, the C++ library solvers for fast steady-state distribution power system analysis of the PowerGridModel [23], the Newton-Raphson power flow solvers in the C++ library lightsim2grid [24], and the PowerModels.jl library [25].

Basically, pandapower provides the means to do power flow analysis for arbitrary grid topologies, power demands, and power generation assets connected to the evaluated grid section. Complex electricity flow evaluation calculates also the power angle variation along buses (lines) and how the insertion of reactive power mitigates local voltage issues.

Figure 1.1 illustrates a network topology example, a simple linear feeder line, which is used to present the essential features of pandapower for analysing power distribution on the low-voltage level. The red node represents the external high-voltage (HV) grid, which supplies power through the medium-voltage (MV) bus to the low-voltage (LV) transformer, which feeds the evaluated power distribution with $U_1 = 230\,V$ per electric phase.

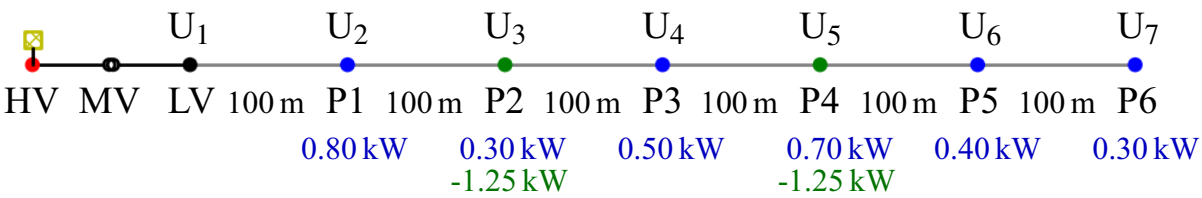


Figure 1.1 – Low-voltage feeder line example with six customers connected in line, where two (green) insert locally generated power

Blue and green nodes depict customer connection points (P_i) at node voltages U_2 to U_7. Each customer is characterized by a different level of power consumption. Green nodes (P2 and P4) indicate customers with generators that insert power at line voltages U_3 and U_5 into the feeder line, adding to the power drawn from the public grid at the HV level.

Figure 1.2 depicts the voltage profile that results from the load and insertion distribution and consequential power flow along the feeder line.

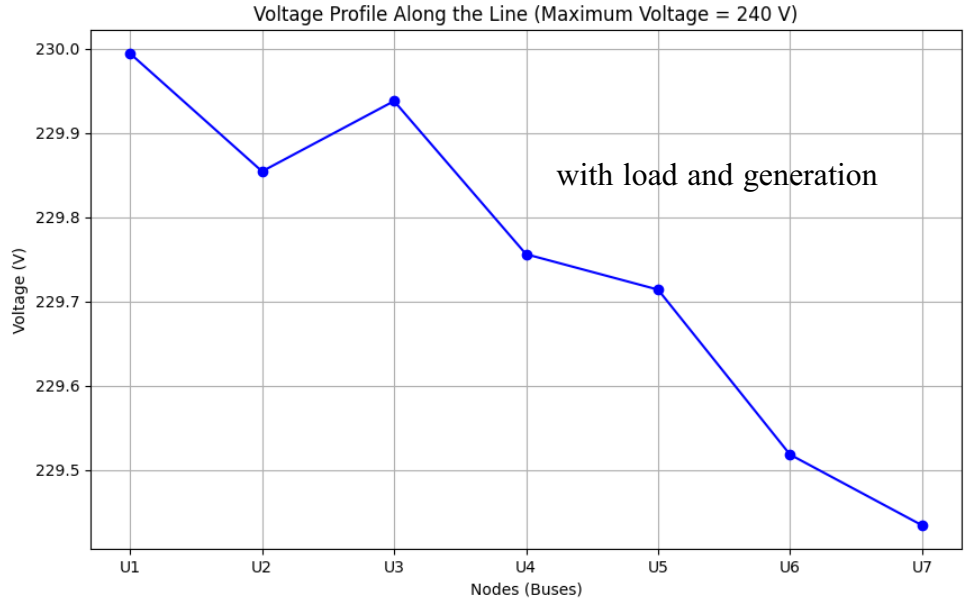


Figure 1.2 – Power flow analysis results for the low-voltage feeder line example with six customers connected in line as sketched in Figure 1.1

Starting from the low-voltage (LV) transformer, the voltage decreases gradually as the electrical power travels through the feeder lines due to the cable resistance, which is assumed to be 0.642 Ω/km. In consequence of the clients that locally generate and insert power, the power flow can be reversed, meaning a power flow toward the transformer can occur, as between U_2 and U_3, causing a voltage increases down the feeder line from node P2 to node P3.

The overall shape of the voltage distribution along lines reflects the dynamic balance between power consumption and local power generation, and reveals whether the voltage remains within operational limits.

1.1.2 Active energy management tools for RECs

In [26] the authors analyse twelve energy modelling tools (EMTs) applicable for RECs and conclude that the socio-economic, environmental, and spatial evaluations criteria required for a comprehensive renewable energy assessment, are not satisfied jointly by any of the evaluated tools. Some tools were found to be quite complete options for REC design, but we need a tool to simulate the energy management that intends to best utilise the REC assets. EMTs that do not consider any demand adjustments toward optimised self-consumption cannot entirely predict the potential benefits of RECs.

Sunshine is the evident signal to improve the self-consumption within a REC that is dominated by PV production: If energy hungry appliance are preferably switched on when the sun is shining, more community produced energy becomes shared instead of sold to some aggregator. However, the response is stochastic. No system can precisely predict when which appliance is switch on. In that respect, demand-supply balancing within RECs is comparable to price driven demand response schemes. Looking for tools to model demand response instead of REC modelling, yields better fitting tools to analyse the dynamics of actively managed RECs by adequate stochastic modelling.

1.1.3 General systems integration and smart home control tools

Two popular tools to actually implement energy management and other systems integration needs are ***Home Assistant*** and ***OpenRemote***. OpenRemote [27] is an open source Internet-of-Things (IoT) platform commonly used for professional automation of many devices. Therefore, it is particularly popular for systems integration. The platform supports many communication protocols and includes visualization features. OpenRemote is open source licensed under the *Affero General Public License version 3*.

Home Assistant [28] is a free and open-source home automation software. It serves as integration platform for smart homes, and enables smart control of smart home devices. The software emphasizes privacy by perfectly local control, i.e., is not cloud based, and requires no specific Internet-of-Things environment. It can be accessed through a web-based user interface from companion apps for Android and iOS, by voice commands via supported virtual assistants, e.g., Google *Assistant*, Apple *Siri*, Amazon *Alexa*, or using the built-in voice assistant *Assist*.

However, these nice software tools seem not applicable for simulation studies with modelled assets, and therefore, are no further considered henceforth.

1.1.4 ***Modelica*** – dynamic heating systems modelling

Open-source modelling has become popular in the scientific community and lead to a variety of openly available modelling languages and libraries. Many scientific works make use of the modelling language *Modelica*, which proves quality and wide applicability. It is an object-oriented, equation-based language developed to model complex systems that are described by coupled differential, algebraic, and discrete equations [29]. Building energy systems are such systems, and therefore, Modelica is perfectly suited to model these. It allows the engineer to model a complex system by constructing a virtual representation that is composed of individual components, e.g., pipes, pumps, vents, and heat exchange. The non-linear behaviour of each component can be defined using textbook equations. A comprehensive introduction to modelling building energy systems using Modelica is provided in [30]. A vast collection of modelling examples for a variety of dynamic systems is presented in [31].

The *Institute for Energy Efficient Buildings and Indoor Climate* at *RWTH Aachen University* has developed and published on GitHub [`https://github.com/RWTH-EBC/AixLib`] the *AixLib* Modelica library, which contains, among others, dynamic models for the built environment, i.e., rooms and heating systems [32]. It is based on the IBPSA (International Building Performance Simulation Association)

Modelica library [33], which provides basic models for the implementation of building and community energy control systems.

To simulate a system that is modelled using the Modelica language, a simulation environment is needed to solve the underlying equation system and run the simulation of the dynamic behaviour that results. *Dymola* (Dynamic Modelling Laboratory) [34] has been developed alongside the Modelica language and offers the best solver capabilities [35]. It enables multi-domain modelling, is open, customizable, and interoperable, supporting FMI (Functional Mock-up Interface), Python scripting, and Simulink integration. It is capable to perform real-time hardware-in-the-loop (HIL) simulation studies. The simulation environment became a commercial product and requires a license, but there is a free trial version with limited capacity to evaluate the usability of the product prior purchase [36]. Open source alternative are *JModelica*, a seemingly discontinued software platform to model, simulate, and analyse complex dynamic systems [37], and *OpenModelica* [38], which is still maintained and further developed, and can be downloaded from [`https://openmodelica.org/`].

1.1.5 Bi-level framework to analyse regional EV charging impact

Grigorev and colleagues explore in [39] how EVs influence regional traffic congestion and energy consumption. They apply an integrated modelling approach, as shown in Figure 1.3, to reveal potential shifts in urban mobility patterns. Arrivals are distributed according to an empiric histogram determined from measured traffic data, and normal distributed charging demand, which here peaks at 50 % of the EV's SoC. The other histograms, as well as the daily queue-length and power consumption, result from the simulation of the such defined distributed multi-server queueing system.

The tool they developed is a comprehensive simulation framework designed to assess the interconnected impacts of EV adoption on local traffic congestion and the regional energy infrastructure. It integrates a dynamic micro-scale traffic

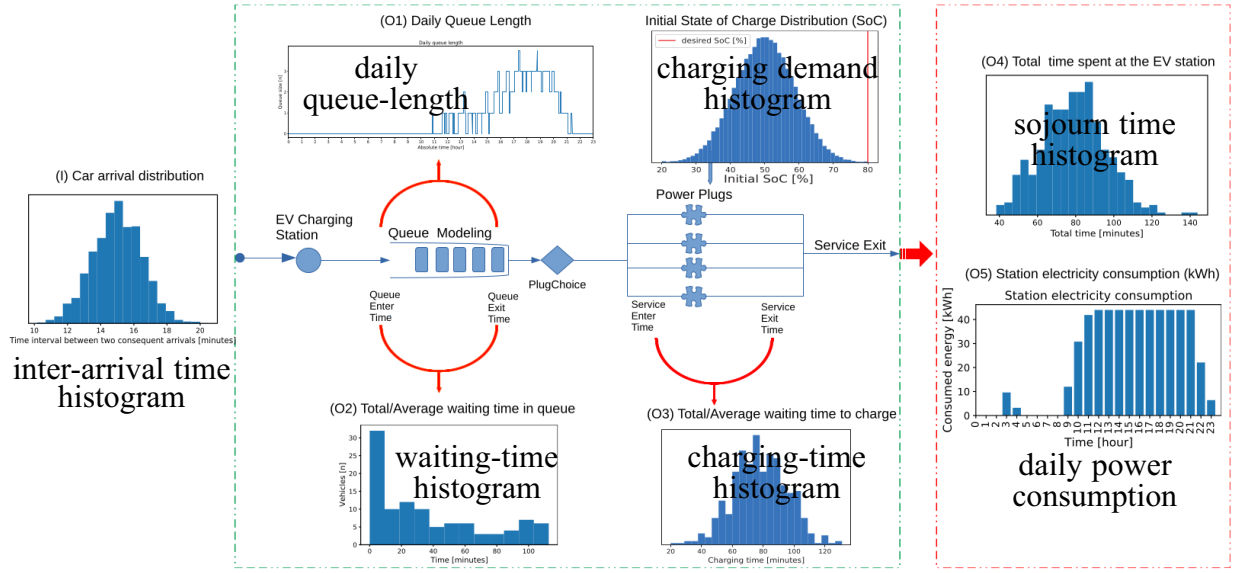


Figure 1.3 – EV charging representation as outlined in [39, Fig. 3]

simulation with a detailed queue modelling system for EV charging stations. The framework operates on two levels: traffic and queueing behaviour modelling (upper level) and energy consumption analysis (lower level), providing a holistic view of the challenges posed by increasing EV penetration.

The framework integrates real-time feedback loops between traffic congestion and energy consumption, enabling a dynamic representation of how one system influences the other. This capability is particularly valuable for identifying critical thresholds where small changes in one system (e.g., a surge in EV users during peak hours) can lead to cascading effects across the electricity grid.

Through its detailed and adaptable design, this tool serves as a powerful instrument for regional planning, helping stakeholders anticipate challenges, optimize resource allocation, and design strategies (policies) to support a sustainable EV integration in urban areas.

1.1.6 ***smartDCsim*** – fast and smart EV charging sites simulation

Most tools for EV charging assume charging at maximum speed. In the case of individual AC charging points this is reasonable up to 90 % SoC. Also

reasonable is a temporal charging management across a region to prevent electricity grid overloads due to many charging events that occur at the same time, as for example proposed in [39].

For an individual site that operates many charging points in behind a single electricity grid access point, for example at a public charging site as shown in Figure 1.4, dynamic limiting of the charging power is beneficial to prevent excessive peak power demands. The tool developed in the course of the bachelor work [40] is predestined to serve latter scenario. In addition, it also considers the dynamic charging power decline of DC charging, in case that is relevant.

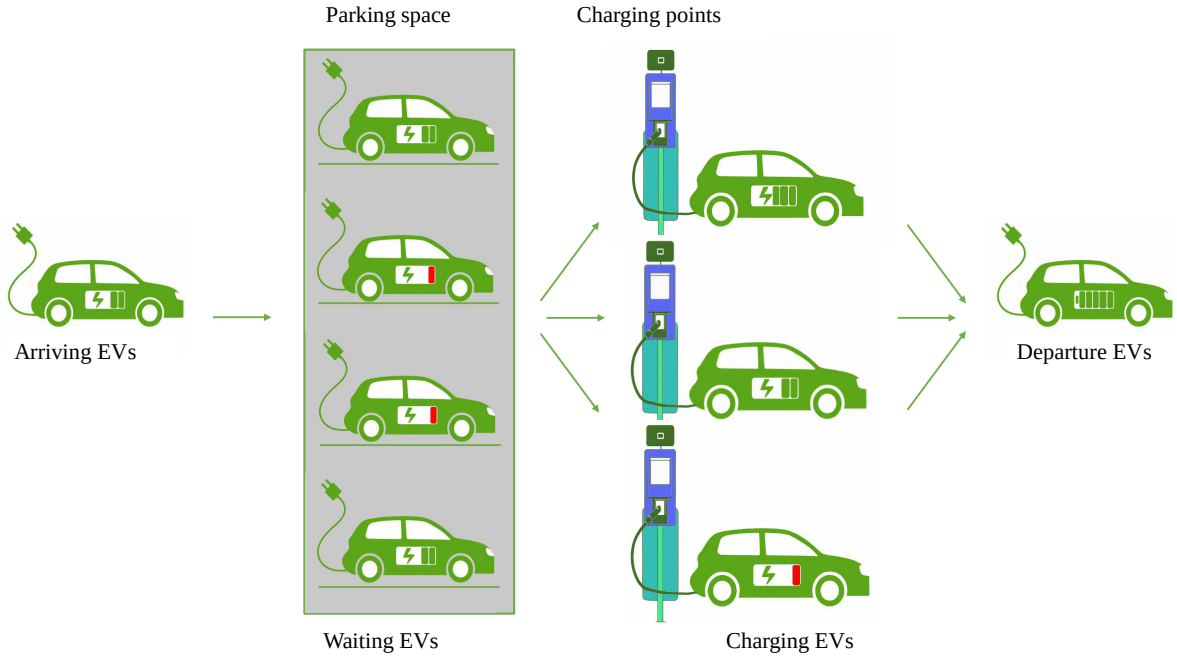


Figure 1.4 – Parallel charging at (virtual) charging site [40]

If the EV charging points owned and operated by a renewable energy community (REC) are virtually grouped and assumed to be behind a virtual community access point, the tool can be used likewise to optimise the usage of self-produced energy. In that case, the allowed power $P_{\text{max}}^{\text{site}}$ needs to become dynamically adjusted to the currently available self-provided community power $P_{\text{self}}^{\text{REC}}(t)$.With this adjustment, this tool serves the aim of the simulation framework to integrate the energy demand from the mobility sector into dynamic energy sector coupling.

1.1.7 Tools to model and analyse cross-sector integration

The ***Open Energy Modelling Framework*** (oemof) [41], provides a novel approach to energy system modelling, representation and analysis. The oemof project aims to be a loose organisational frame for tools in the wide field of (energy) system modelling. It offers an open-source toolbox to construct energy system models and is free to use.

The toolbox supports integrating various energy systems like renewable energy sources, storage, heat, and gas with electrical grids to simulate and analyse the interaction between sectors, for example how the heating demand impacts the electricity demand depending on the used heating technology. In addition, it is compatible with pandapower for electricity grid modelling.

Based on a generic graph-based description of energy systems, it is suited to model complex cross-sectoral systems and to incorporate various modelling approaches. This flexibility yields a multi-purpose modelling environment capable to analyse linked energy systems at scales ranging from urban to transnational. It can be used to optimise energy system configurations that balance electricity and heat generation in combination with renewable energy sources and storages.

Calliope is an open-source energy systems modelling framework that scales from local to international energy systems [42]. It provides high temporal and spatial resolution and clearly separates the framework (tools) from the models (data). The focus is on energy system planning. In an optional operation mode it enables the evaluation of a pre-defined energy system in varying operational and environmental conditions. Preliminary simulation results can be explored interactively while the simulation is in progress.

An energy system model for *Calliope* consists of several text files in YAML and CSV format. These files define different technologies, locations and asset specifications. From these input files, an optimisation problem is derived and solved. The results are provided as *xarray* datasets and converted into *Pandas*' data structures, which can be analysed with built-in tools or using Python data analysis libraries designed to handle *Pandas*.

Key features of *Calliope* are:

- State-of-the-art Python toolchain based on Pyomo, xarray, and Pandas
- Generic definition of production, storage and consumption units
- Definition of locations with different assets and loads
- Time series definition with arbitrary resolution
- Human and machine readable model specification in YAML format
- Support of high-performance computing (HPC) clusters
- Apache 2.0 open-source license (free to use)

The publications listed on www.callio.pe show that *Calliope* can effectively cover different energy sectors and their coupling.

1.1.8 General purpose tools utile for sector independent modelling

Julia is a dynamic, general-purpose, high-level programming language, designed to be fast and powerful. It is commonly used for machine learning, artificial intelligence, modelling and simulation, numerical analysis and data science in general. It links with other programming languages such as Python, Java, C, C++, R and Matlab/Simulink integration. A distinctive feature is its parametric polymorphism and that it relies on structs with generic methods/functions that are not tied to them, which enables versatile coding. By default, Julia programs are executed using a just-in-time compiler with support for ahead-of-time compilation and efficient garbage collection.

JuliaSim is a high-performance programming environment designed to merge traditional modelling and simulation with machine learning. JuliaSim can build accelerated surrogates from component-based models, such as those conforming to the FMI (Functional Mock-up Interface) standard, using Continuous-Time Echo State Networks (CTESN). The foundation is an acausal modelling language that can compose the trained surrogates as components within its staged compilation process. A complementary model library with differential-algebraic equations and pre-trained surrogates can be used to compose modelling systems

for design, optimization, and control applications. See "Composing modelling and Simulation with Machine Learning in Julia", arXiv:2105.05946.

MATLAB, literally *MATrix LABoratory*, and it's open source siblings ***GNU Octave*** and ***scilab***, offer a powerful multi-paradigm programming language and a numeric computing environment that provides effective interfacing with programs/modules in other languages. The additional *Simulink* package adds graphical multi-domain simulation and model-based design for dynamic and embedded systems, but only for MATLAB developed by MathWorks, and not for the open access siblings. A particular strength of either are the powerful matrix manipulation functions and the integrated visualisation features, which produce high quality graphical results (vector graphics).

1.1.9 Comparison and selection of modelling and analysis tools

To identify the best candidates that fit to the tasks ahead in research projects on joint optimisation of collaborative cross-sector energy systems, the tools outlined above need to be compared.

The relevant features are:

- power-/heat-flow analysis (effective/dynamic)
- managed energy-flow/-sharing
- stochastic demand modelling (electricity/heat/mobility)
- dynamic response to external signals (in-the-loop)
- controlled EV charging (peak-/cost-minimisation)
- transient analysis of energy efficiency/losses
- cross-sector balancing (demand optimisation)
- runtime data in-port and ex-port

Table 1.1 summarises the expected support of the required features by the different tools preliminary evaluated.

Comparison: Only principal support of common requirements is compared independent of the different energy sectors the tools are developed for. Every

Table 1.1 – Features supported by tools in the respective energy sector

feature → ↓ **tool**	power flow	energy flow	demand model	ext. signals	contr. charge	trans. analysis	demand opt.	data in-export
pandapower	+	-	-	-	-	+	-	+
Modelica	+	-	+	-	-	+	-	+
AixLib	-	+	+	+	-	+	-	+
openModelica	+	+	+	+	-	+	+	+
bi-levelEVsim	-	+	+	-	+	-	-	-
smartDCsim	+	+	+	-	+	-	+	-
oemof	-	-	-	+	-	-	+	+
Calliope	-	+	+	-	-	-	-	-

Note: The rating is based on the primary aims. Most tools can be extended by custom modules or external tools.

tool has it's unique strengths and some weaknesses are relevant only when put to actual use in a specific scenario. Thus, the best tools for a specific job are not necessarily the ones that serve the most features in Table 1.1.

That *pandapower* is a feasible and good choice to analyse electricity flows is evident. In its core, it is a power-flow solver with some software environment that prepares the in-put *pandas* the solver needs. Any software that can provide these in-put *pandas* and can use the power-flow result *pandas* that are returned, can integrate *pandapower* to analyse both, effective and complex electric power flows at any scale and on every grid level.

Modelica is in its core a programming language to specify meshes of complex dynamic behaviours, each describe by textbook equations, and linked together by the used variables. Dedicated solvers can solve the resultant dynamic system of equations. *OpenModelica* provides the software environment required to design modules, model demands, and do simulation studies. First, modules that model different systems need to be designed. Such modules are provided by the *AixLib* library for thermic systems, from simple heat distribution pipes, pumps and heat exchangers to complex heat demand models for different types of households and buildings.

The *bi-level EV simulation* used to model the impact of the electrification of individual mobility on the traffic and the electricity grid within a region

shows an interesting result but seems infeasible to be integrated with other tools. Possibly anything may be added, but it is not clear how. The same applies for the *smartDCsim* tool developed in the bachelor work. However, this source code we know line-by-line and thus, can adapt it to serve any other tool.

The *oemof* platform is an umbrella specification for energy system models and tools. Different teams develop models and tools for different energy sectors. The aim of the umbrella specification are exchangeable modules and a common ontology. To which extent modules can cooperate in a cross-sector optimisation approach is unclear.

Selection: Choosing *pandapower* to perform power-flow analysis and *pandas* for the data exchange appears logical. For heating system and demand modelling and analysis the *AixLibs* promise to be versatile, and *OpenModelica* should provide the necessary environment to design and evaluate heating systems. To model and analyse modern EV charging, the *smartDCsim* source code will be used to create a module that can be integrated in an overarching event based simulation environment, yet to be developed based on the *smartDCsim* simulation core. If a new tool is developed from scratch, *Julia* should be considered.

1.2 Analogous cross-sector simulation frameworks

Having compared and selected tools from different sectors to design a cross-sector simulation framework based on sector specific co-simulation, we finally compare this approach with existing frameworks that provide cross-sector capability by themselves. Table 1.2 summarises the assessment attempt.

The features considered for the cross-sector simulation assessment are:

- The potential sectors (electricity / heat / mobility / H_2 and e-fuels),
- self-optimisation, variable demand, distributed coordination support,
- and if the tool is open source and free to use for scientific work.

Calliope is primarily a tool for the design and dimensioning of energy grids considering cross-sector energy conversion options. Being based on a priori

Table 1.2 – Analogous simulation environment comparison

feature → ↓ **tool**	electricity	heat	mobility	H_2 e-fuels	var. demand	distr. coord.	self optim.	open source
Calliope	+	+	+	+	-	-	-	+
OpenModelica	+	+	+	+	+	+	-	+
JuliaSim	+	+	+	+	+	+	+	+
MATLAB	+	+	+	+	+	+	+	-
pandapower	+	-	-	-	-	-	-	+

Note: The rating is based on what can be integrated assumedly without major changes to the core of the tool.

assumed supply and demand predictions and the optimisation of a single parameter, commonly energy cost or CO_2 emissions, the information gained from simulation results is global only and difficult to relate to individual components, assumptions, or control strategies. Many simulation runs with changed parameters are required to identify the source of meagre optimisation results.

Modelica can in principle be used to model any dynamic system, including electric systems, and is in principle capable to model cross-sector integration. However, the existing libraries are focused on a specific sector each, and thus, coupling modules across sectors can be cumbersome due to the different modelling approaches, paradigms, and ontologies of different libraries. Thus, mixing models from different sectors to design and analyse cross-sector integration using ***OpenModelica*** might be an unresolvable task.

The dynamic systems modelling frameworks briefly discussed in Section 1.1.8, i.e., the programming language *Julia* and the simulation environment ***JuliaSim***, and the mathematical modelling and evaluation environment ***MATLAB*** and its free to use siblings *GNU Octave* and *scilab*. These are so generic that modelling a cross-sectorial system-of-systems becomes a programming task by itself. That can be very efficient and effective, but contradicts the intention to use established modelling from the different sectors.

1.3 Object of automation

The objects of physical automation are the energy producing and consuming assets and appliances operated at customer premises (behind-the-meter). These respond to signals received from the electricity grid (frequency/voltage), an overarching EMS (possibly managing a REC), or peer assets and appliances managed by a home or building automation system. Jointly and fully automatically, by default, these shall support best utilisation of renewable energy sources. The automation can be evaluated using the developed simulation framework, but cannot be achieved by simulation.

The minimal example scenario (object) for which automated energy flow management shall be simulated (to approve correct modelling and energy-flow management) is specified by the client as follows: Three buildings equipped with energy assets and appliances as detailed in Table 1.3 shall be assumed connected to the same feeder line such that their energy-flows become summed up.

Table 1.3 – The automated objects to be simulation (as specified by the client)

Parameter	private buildings	apartment/office buildings
feeder network access limit	5 to 20 kW	50 to 100 kW
heating/cooling power	up to 5 kW	20 to 50 kW
heating target temperature	23 ℃	22 ℃
cooling target temperature	25 ℃	24 ℃
peak PV production capacity	5 to 20 kWp	50 to 100 kWp
site EV charging power limit	5 to 20 kW	50 to 100 kW
number of charging points	1 to 4	5 to 30
number of assigned EVs	1 to 4	5 to 100

The actual configuration of the scenario finally demonstrated may be chosen such that representative results are achieved. However, the GUI needs to enable adding, varying, and removing of such specified and configured buildings.

The objects of automation within the simulation framework are (1) the automated default configuration of simulated systems and the entire scenario

based on the latest configuration, which eases the definition of similar components and simulation steps, (2) the automated execution of consecutive simulation runs, which allows the empiric analysis of sensitivities to parameter changes, (3) the automated results recording, processing and saving, (4) the automated statistical evaluation of results, yielding information on the trustworthiness of for example mean values, and (5) the automated scaling of generated visualisations, which sometimes can be irritating. An automated installation of the entire simulation framework, via batch script or Docker®, would be possible, but was not requested.

The objective of integrating established and highly performant sector specific simulation tools into a versatile simulation framework is the automated execution of co-simulation studies. The event based integration approach achieves asynchronous real-time coordination and linking of the simulated systems. This achieves a realistic and fully automated cooperation of the modelled systems (digital twins) as implemented by the evaluated coordination approach.

1.4 Conclusion of Section 1

In this master work, existing tools for different sectors are linked together by effectively parsing the output of one as input to the other and vice versa. Thereby, a cross-sectorial co-simulation environment is achieved based on established sector specific system modelling tools. The overarching simulation control is provided by an agnostic event simulation core and assignable data recording and visualisation options.

Selecting and integrating feasible tools state a core research question and the development task, respectively. Their achievement is documented in the following chapters step-by-step. Exemplary simulation studies conclude the work performed and show the applicability and performance of the developed cross-sector energy system simulation framework.

2 MODEL INTEGRATION AND CO-SIMULATION APPROACH

The fundamental design and technical basis of the developed co-simulation approach state the frame to link together the different system models and simulation steps performed by different tools. Consequentially, we first develop the system-of-systems architecture and the joint simulation core, the framework, before we integrate different (sub-)system models handled by the linked-in software tools.

2.1 Event-based Simulation

First, a common approach to time needs to be established. Evaluating the entire system-of-system in between two time-instances between which nothing changes is surplus. The result will remain the same until something changes.

Using event-based simulation, the system is evaluated only when some change, an *event*, occurred. But some processes occur continuously, for example the generation of electric power by a PV-system. If the power output is rather constant, the energy production is linear and can be calculated a priori for the entire time-interval during which the output power does not change. This linearisation is in reality never completely perfect but random noises are in general negligible if they are unbiased and small compared to the modelled parameter, in the example the PV output power.

Using event-based simulation, the simulation-time progresses non-linear, as shown in Figure 2.1. The current time t_i is always the occurrence time of the currently processed event, here event_{b_1}.

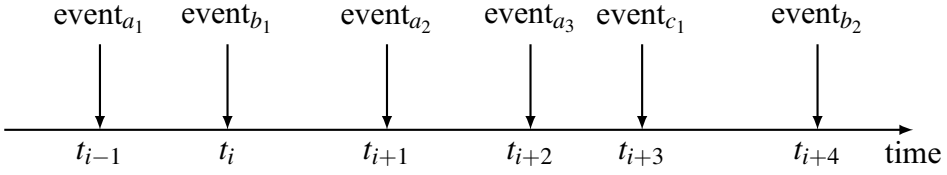


Figure 2.1 – Event-based simulation time

Whenever handling event$_{b_1}$ is finished, the simulation time jumps to the time of the next event in the event-stack and to time t_{i+1}. The future, the next event, is not known until the next event is picked for processing. In particular, the currently handled event can schedule a next event at the current time, which evidently becomes sorted to the head of the event-stack, pushing all previously scheduled events back, as sketched in Figure 2.2.

The simulation process is composed of:

- Event: Has execution time, event type, and links to the related client
- EventType: Determines the respective event handling
- EventHandler: Processes current event in *next event* → *get next* loop
- EventStack: Stores future events in order of occurrence

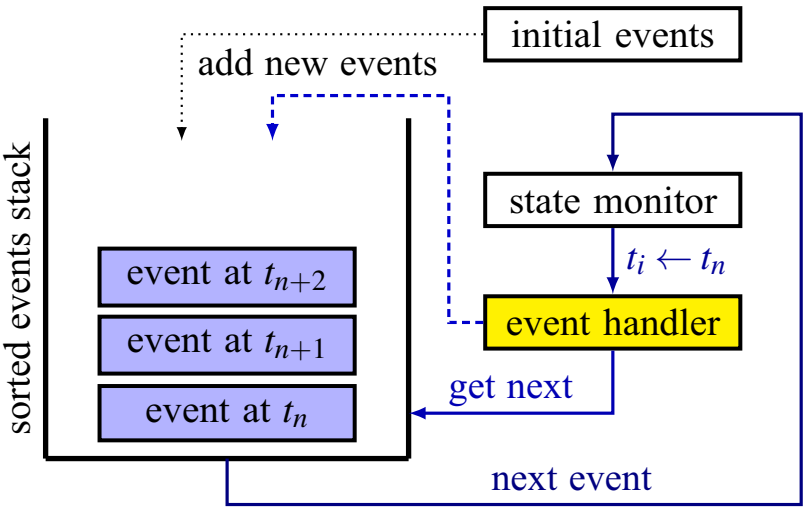


Figure 2.2 – Event-based simulation core [43, 44]

Event-based simulation starts with creating initial events for each client initialising the EventStack. To run the simulation, the EventHandler consecutively pulls events from the EventStack and triggers the processing of the pulled event by the referenced client system. Prior executing an event, the EventHandler is responsible to update the simulation time, and in case of multi-threaded event execution, that chronology and inter-event dependencies are preserved. The actual processing of Events is delegated to the (sub-)system that the referenced client belongs to.

To preserve the time that passed in between two successively handled events, it is paramount to calculate and store $t_{i+1} - t_i$ as the time that just passed, before the simulation time is updated to the execution time of the current Event. With some care, the current time can be consecutively re-set ($t_i = 0$) to avoid digital overflow issues and enable infinite simulation runs.

The pull-and-execute loop continues until the EventStack becomes empty, which terminates the simulation. To keep the simulation running, the handling of events needs to continually create new events that are pushed into the EventStack. All future events become sorted into the EventStack, which is a list of Events sorted by their execution time. After it has been handled by the EventHandler, the Event is discarded and the next Event is taken from the EventStack.

To end the simulation in a controlled manner, the creation of new events needs to be stopped by setting an EndSimulation flag that either all (sub-)systems need to obey or is implicitly enforced by the EventStack rejecting new events. The setting of flags may be triggered by a dedicated event type, such that also client systems can trigger a controlled termination of a simulation run.

To evaluate the behaviour of a system in respect to changes of the initial setting, a sequence of dedicated simulation runs is required, as shown in Figure 2.3.

The event-based simulation core involves (sub-)system models solely at the instants when events are scheduled. The time that passes in between can be infinitely short or very long. Thus, it operates on the premise that no alterations happen between events. This selective attention mechanism streamlines the analytical process not only computationally but also by sharpening the focus on the elements and changes that truly matter.

2.2 Cross-sector simulation framework composition

The central module of the simulation framework is the *Simulation control* module that also collects and records all generated data, as shown in Figure 2.4.

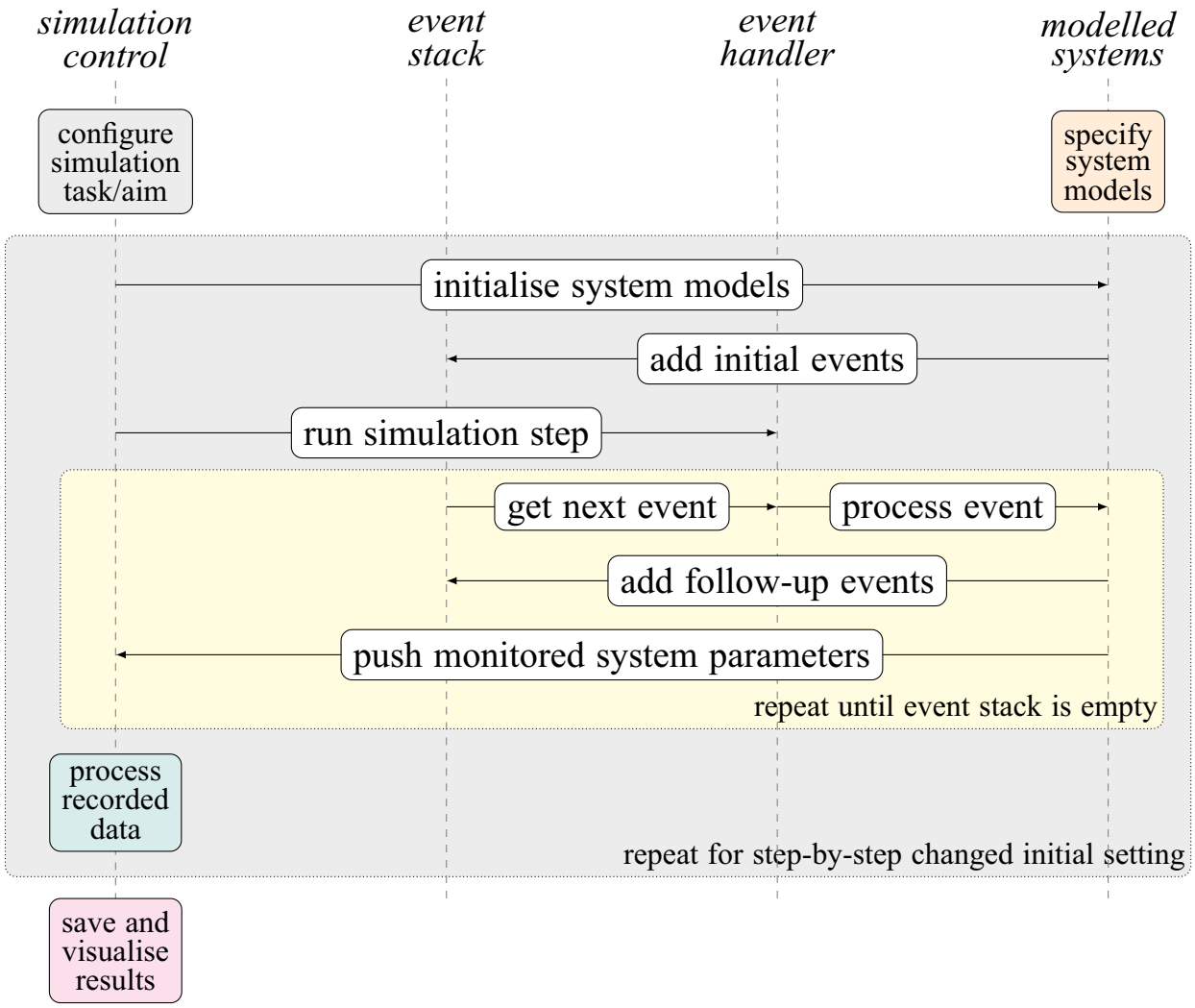


Figure 2.3 – UML sketch of a sequence of simulation runs where system settings are changed step-by-step to evaluate the cross-sector response to these changes

To perform that, it contains the interfaces to embed the external modelling tools and controls the simulation runs.

The external tools, *OpenModelica* to model heat energy systems, *pandapower* to model electricity systems, and *smartDCsim* to model EV charging demands and processes, handle the events that relate to them individually via the provided interface. If the processing of an event triggers the generation of a new event, this is pushed to the *Event-based Simulation Core* module, which schedules it and later triggers its execution.

A multitude of parameters is required to create an accurate model of every simulated system, as detailed in the following sections. These parameters need to

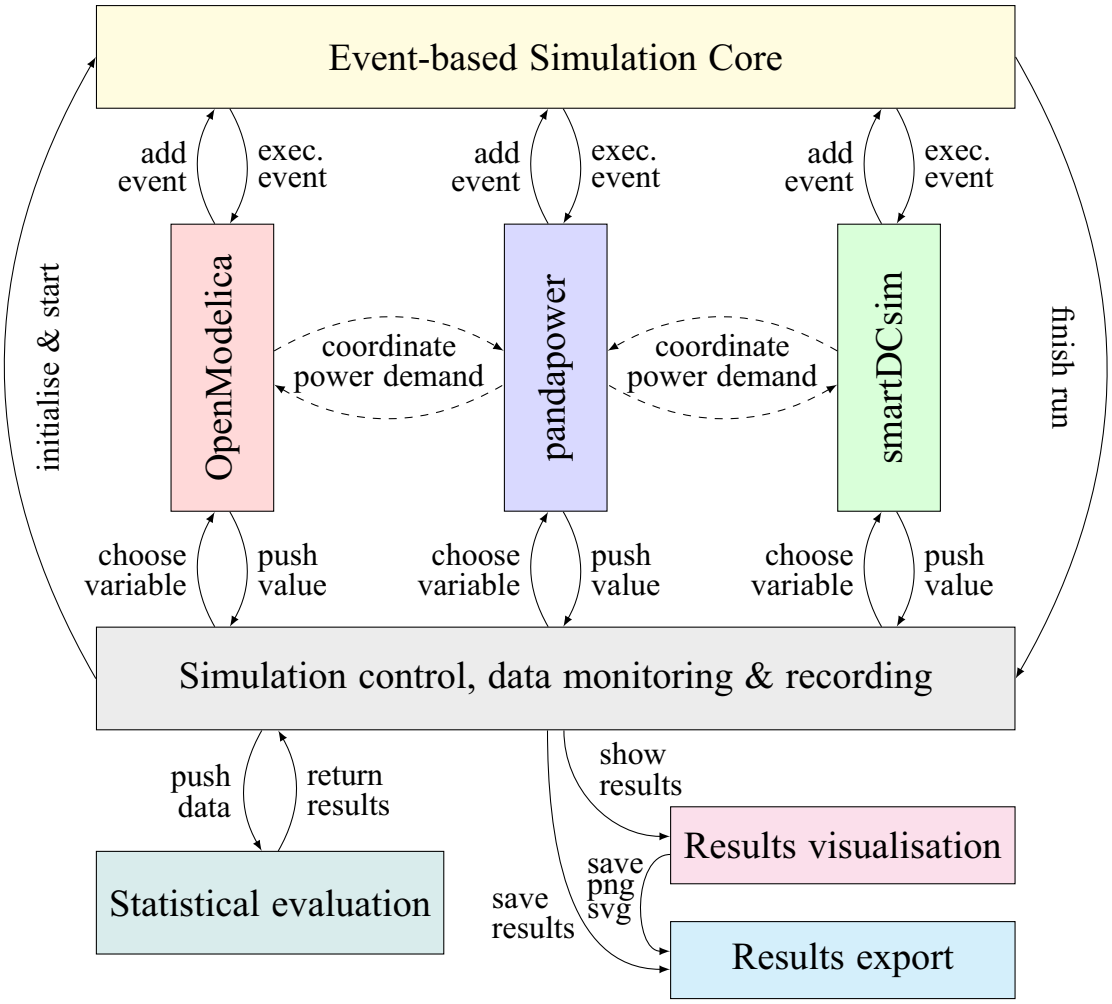


Figure 2.4 – Co-simulation framework modules architecture

be handed over to the according external tool, most conveniently via dedicated files, one per simulated system. These configuration files are loaded in the initialisation phase. Also the selection of the variables that shall be monitored and evaluated needs to be settled before the simulation starts. This step initialises the necessary data records and run-time visualisation options.

Whenever a simulation run is finished, or a sufficient amount of data is collected, the recoded data is analysed by the *Statistical evaluation* module. This is separated to potentially use an external tool as well. The same applies for the *Results visualisation* and *Result export* modules, which show and save the achieved results respectively.

The parameterisation of a simulation task shall be imported form an XML file. How this XML file is generated is irrelevant for the quality of the result.

Thus, Figure 2.4 includes no module that provides a graphical user interface (GUI) to configure the simulation. However, a self-explanatory GUI is a good feature to ease effective use of the simulation framework and thus, a recommended option.

2.3 Electricity Demand Modelling and Simulation

Electricity demand is commonly approximated by so called *standard load profiles* or *load patterns* [45]. These are derived from many years of consumption monitoring and according clustering of customers into classes with similar consumption pattern. This approximation works nicely for aggregates of thousands of customers, in particular for the supply-demand balancing on the energy market, where the independent deviations of individual customers from their standard profile become partly equalised by the laws of statistics (central limit theorem).

On the level of a single household or building, the actual match with the assumed load profile is hardly assessable. The randomness and resulting unpredictable variation of the demand is far too big to model a single household or building by an average load profile. The alternative is modelling every energy demanding appliance, from the dishwasher to the light bulb, and make an educated guess about the individual activation patterns of the different energy appliances. Such detailed models are nice engineering work and very welcome to test for example home and building energy management systems (HEMS/BEMS).

This detailed modelling alternative cannot be used to model many random customer for whom neither precise knowledge about the actually installed nor the diverse usage patterns exist. The variety of possible demand compositions that result in the same standard load pattern are far too vast. Therefore, and because for general applicable conclusions no detailed demand compositions as in [46] shall be assumed, we introduce heavy tailed variance to the standard load profile, here Lomax(2,1) [47] rescaling, as shown in Figure 2.5.

Thereby, an aggregate over $n \to \infty$ modelled customers results in the assumed load profile. Still, each modelled customer causes a demand volatility

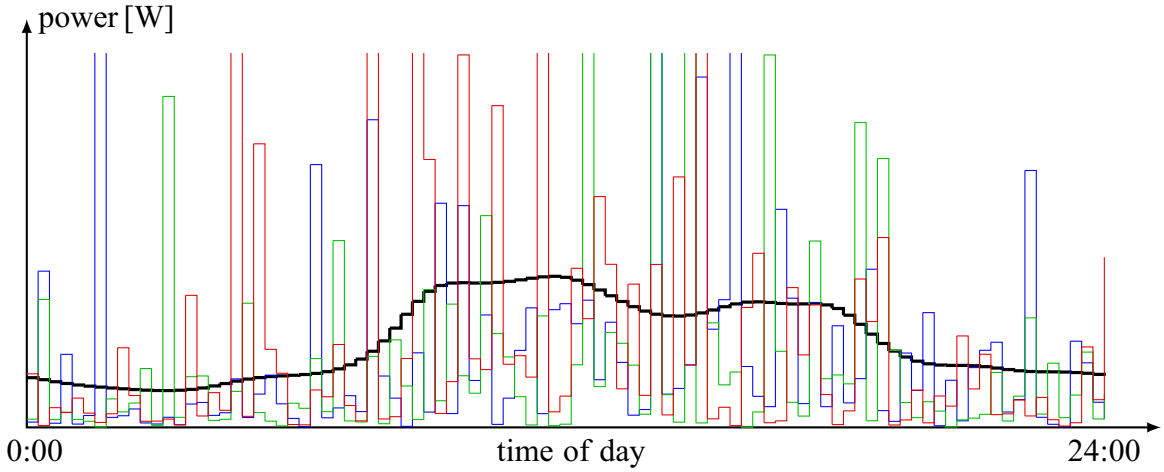


Figure 2.5 – Random load modelling: Heavy tailed scaling of H0 standard load profile mimics random consumption (red, green, blue being exemplary samples)

that resembles the expectable temporal demand variation of individual real customers. Clear load patterns evolve from merged customer demands only.

2.3.1 Modelling of feeder lines and customer demands with pandapower

To simulate a feeder line, first a feeder network needs to be designed in pandapower. To construct the simple example sketched in Figure 2.6, first a main substation bus with a voltage of 20 kV is created as unlimited power supply. A 20/0.4 kV transformer is connected and feeds the *start bus* with a voltage of 0.4 kV. The remaining feeder line is modelled as a chain of buses, where at each intersection a customer is connected. For simplicity and clarity of results, each line segment is modelled equally, with the same line length of 100 m and wire type NAYY 4x50 SE. With a maximum current of 142 A per phase and an electrical resistance of 0.642 Ohm/km, it can be used safely for power flows up to 100 kW.

Individual load profiles are generated for each customer to incorporate consumption patterns with realistic variation and volatility. The average base load of 0.4 kW is varied according to a 24-hour standard load profile and a Beta distributed variance in the range 0 to 5 with mean 1. A 24-hour load pattern suggested by ChatGPT [chatgpt.com] introduces the in average typical daily

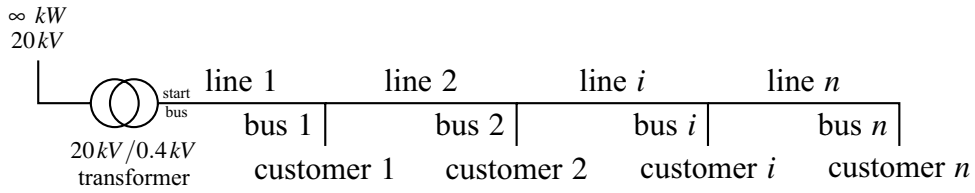


Figure 2.6 – Example feeder network

demand variation, whereas the random variation ensures diverse, individually unpredictable volatility of the individual customer loads.

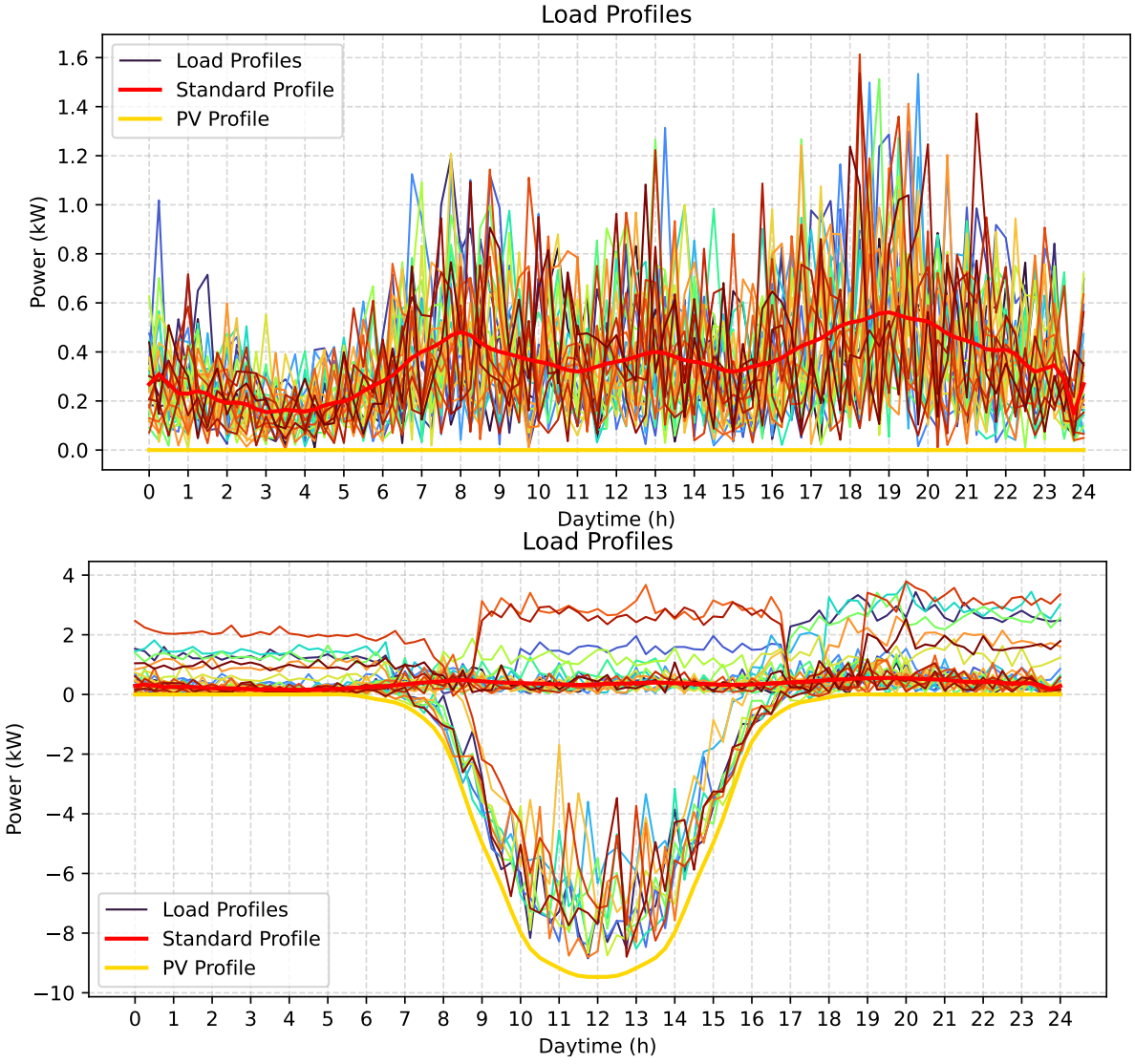
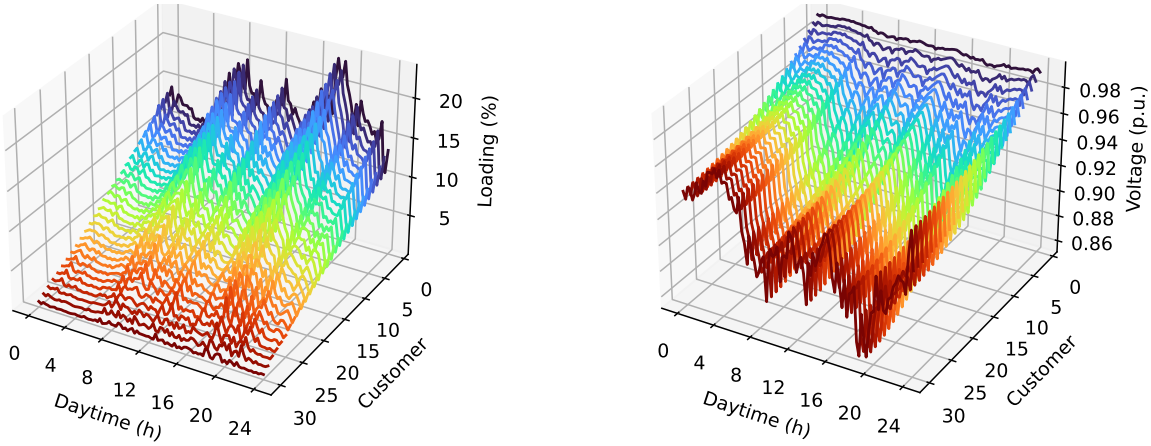


Figure 2.7 – Random customer load profiles w/o PV generation and EV charging

The generated power demands shown in Figure 2.7 for mean $0.4\,kW$ per customer shows the typical asymmetry of short high load peaks and more frequent loads below the standard load profile (red line) scaled to match the assumed

mean load. Whereas in Figure 2.5 the volatility of the peaks within 15 minute intervals is addressed, here the average load appears more smooth because most peak loads of households are of short duration (few minutes) and average out over long time interval. The output of PV systems at customer premises, here assumed for one third of the customers with a theoretic peak power of $10\,kWp$ each, causes negative load in case the produced power is not consumed internally within a site. Therefore, it shows up in Figure 2.7 as negative bump that stays within the boundary of the approximately sine-square shaped peak production limit of a PV system (yellow line). Variance is here introduced by the weather and obstacles causing temporary shading. On the load side, we see the impact of EV charging. With assumed $3\,kW$ these are clearly visible because they persist over many time intervals. EV charging is here assumed to occur at every second site with alternating charging times, modelling EV charging at work and at home.

In a 24 hours time-series simulation, the demands of each customer are updated at every 15-minutes time step based on their respective load profile. For the examples here, these updates are directly applied to the network model. Power flow analysis is performed step-by-step using pandapower's *runpp()* function. Line loading and bus voltage levels are recorded for and visualised as shown in Figure 2.8.



(a) Line segment loading over time

(b) Voltage at customer buses over time

Figure 2.8 – Calculated line loads and voltages at customer buses

The line loading percentage reflects how heavily the feeder line is stressed

as the demand fluctuates throughout the day. The voltage levels indicate how well the feeder network maintains acceptable voltage limits under varying load conditions, in particular for the customers at the far end. Here, for the chosen feeder design, the voltage drop is critically high and we recognise that the chosen cable is not up to the job if PV and EV charging becomes added. However, to show the effect of smart grid control, this misconfiguration helps to highlight the effect on the voltages along the feeder line.

The results are visualized in 3D to support a clear understanding of the feeder line behaviour. Line loading over time shows how peaks accumulate toward the transformer, stressing the energy supply. The voltage drops caused by these peak loads accumulate toward the far end of the feeder lines, causing potential trouble for voltage sensitive appliances used by customers connected far away from the transformer.

Displayed in 2D, as in Figure 2.9, the inverse correlation is clearly visible. The inversion of the power flow caused by PV insertion is only visible from the voltage levels. The power flows are displayed as absolute values, which is common because for the wire load the sign is not relevant ($P = I^2 \cdot R$).

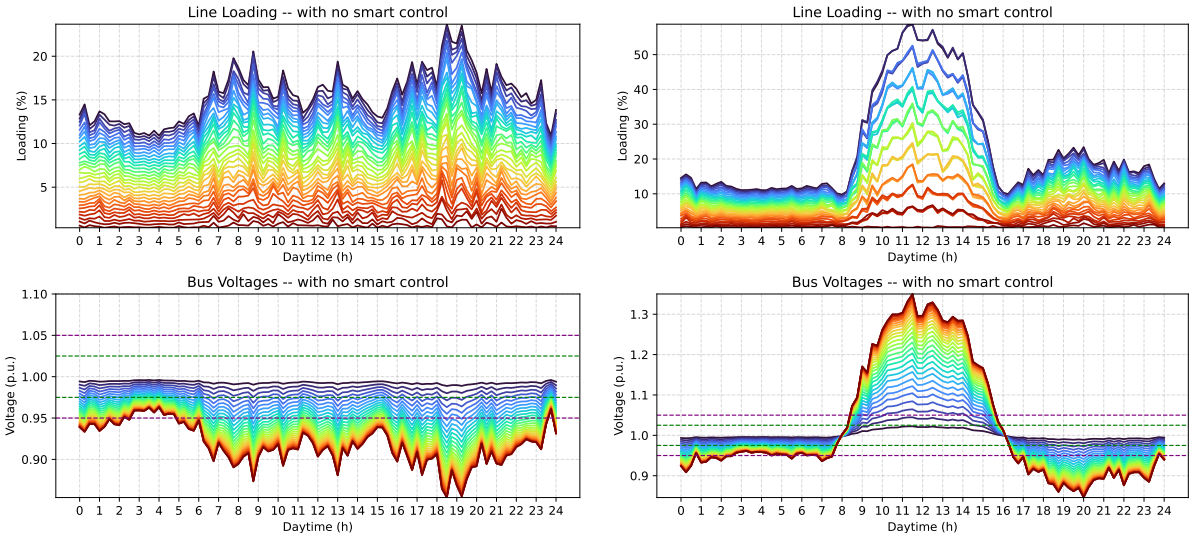


Figure 2.9 – Inverse correlation of line loads and line voltages at customer buses

What also can be seen, is that in practice connecting ten distributed $10\,kWp$

PV systems does not necessarily yield 100 % load on a wire rated to $100\,kW$. Even though here $10 \times 10\,kWp$ PV systems theoretically sum up to $100\,kWp$, the actual load on the feeder line is far from 100 %.

2.3.2 Electricity customer and environmental events

An electricity customer is a person that can always decide to interact with the appliances that consume electric energy. Human bodies do not respond well to consuming electricity. The customer can for example switch on the washing machine or a water cooker at any time. The conscious variation of the power demand needs to be modelled in case some grid signalling, for example varying energy price or a kind of traffic light signal, shall be tested. Customer responses to signals are never guaranteed. To provide a flexible scheme to specify customer interaction, the modelling includes:

- ΔP – power change volume (scalar or mean and distribution)
- Δt – power change duration (scalar or mean and distribution)
- p_Δ – power change probability (mean of normal distribution)
- $[t_i..t_j]$ – daytime interval when the specified power change can occur

The triggers, meaning the other systems that can trigger the potential response, need not to be known. The other systems that send signals (triggers) need to specify which customer response characteristic shall be assumed to correctly model a realistic response to their signals.

The weather is another external factor that can have an impact on power demands. Customers and appliances behave different under different weather conditions. Heating systems respond to the outside temperature and PV systems to the solar irradiation. Customers may change their daily routine depending on the weather, for example, wash cloths when the sun is shining because they wish to dry the cloths in the sun.

To use realistic weather data, an open-access weather data from Open-Meteo.com is integrated via public API [48]. For reproducibility, simulation

studies are commonly executed based on recorded weather data. The weather object integrated in the simulation control module provides the following set of parameters:

- T_{out} – outside temperature
- I_{sol} – solar irradiation
- p_{rain} – rain probability
- p_{cloud} – cloud cover
- V_{wind} – wind speed and direction
- $?_{out}$ – relative humidity
- $\cdots$ – many more (as provided by data source)

Every weather parameter can be used as trigger for a potential power flow change. PV systems respond to I_{sol} and heating systems to T_{out}, quite evidently. PV and heating systems linked to weather data needs not be modelled stochastically. However, the user has to make sure that the weather data is unbiased if universal conclusions and inferences are envisaged. If realism is irrelevant, the stochastic modelling is preferential. For example to analyse the behaviour of systems in extreme situations.

2.4 Heating System Modelling and Integration

OpenModelica is an open-source environment, available free of charge from the official website openmodelica.org, based on the *Modellica* [29] modelling language [38]. In this environment, sub-systems (modules) can be created from mathematical (textbook) equations, which than can be combined to entire systems (models) and linked together to form an entire system-of-systems. The interoperation of the entire model can than be simulated over a configured time-span. Graphical and a code-based views are available to edit the sub-system's composition into a system-of-systems.

The open-source model-library AixLib [32] provides a vast variety of sub-system modules that can be used to create and simulate complex HVAC

(Heating, Ventilation, and Air-Conditioning) systems with varying levels of detail. After installing OpenModelica Edit (OME), search for AixLib in *File → Manage Libraries → Install Libraries* and install it to get access to the available modules, models and examples. Latter help to understand the features of the different modules and how to configure them.

2.4.1 Modelling of heating systems and heat demand using OpenModelica

Heating and cooling is a shift-able load because both, the air volume and the circumventing walls serve as heat storage. These heat buffering capacities introduce heavy system non-linearity because the rate at which heat is absorbed and released is determined by the heat resistance of the surface. Thus, these heat capacities introduce RC-entities with complex behaviour.

Figure 2.10 shows a heating and cooling example, where the house is modelled by an air volume that represents the heat capacity that shall be kept within a tolerable temperature band. A wall separates the room from the outside with a heat resistance and also a heat capacity that stores absorbed heat and releases it if the indoor temperature falls below the wall temperature. Four windows facing in the four cardinal directions are required to consider the heat inserted by solar irradiation according to the provided weather data. They also summarise the heat losses that windows introduce.

Heating is simplified to a radiator and a pump that causes the circulation of constantly hot heating water. The pump is controlled on/off by the indoor air temperature. The cooling is accordingly simplified to a fan that blows the room air through an heat exchanger that is connected to a permanent coolAir source.

The annual performance of the heating and cooling system model presented in Figure 2.10 is shown in Figure 2.11. The purple curve shows the outside temperature from the weather data record 2009 in Kiev. The achieved indoor temperature is shown by the upper blue curve. The red curve shows the power demand of the hot water pump, which is not active during the summer months,

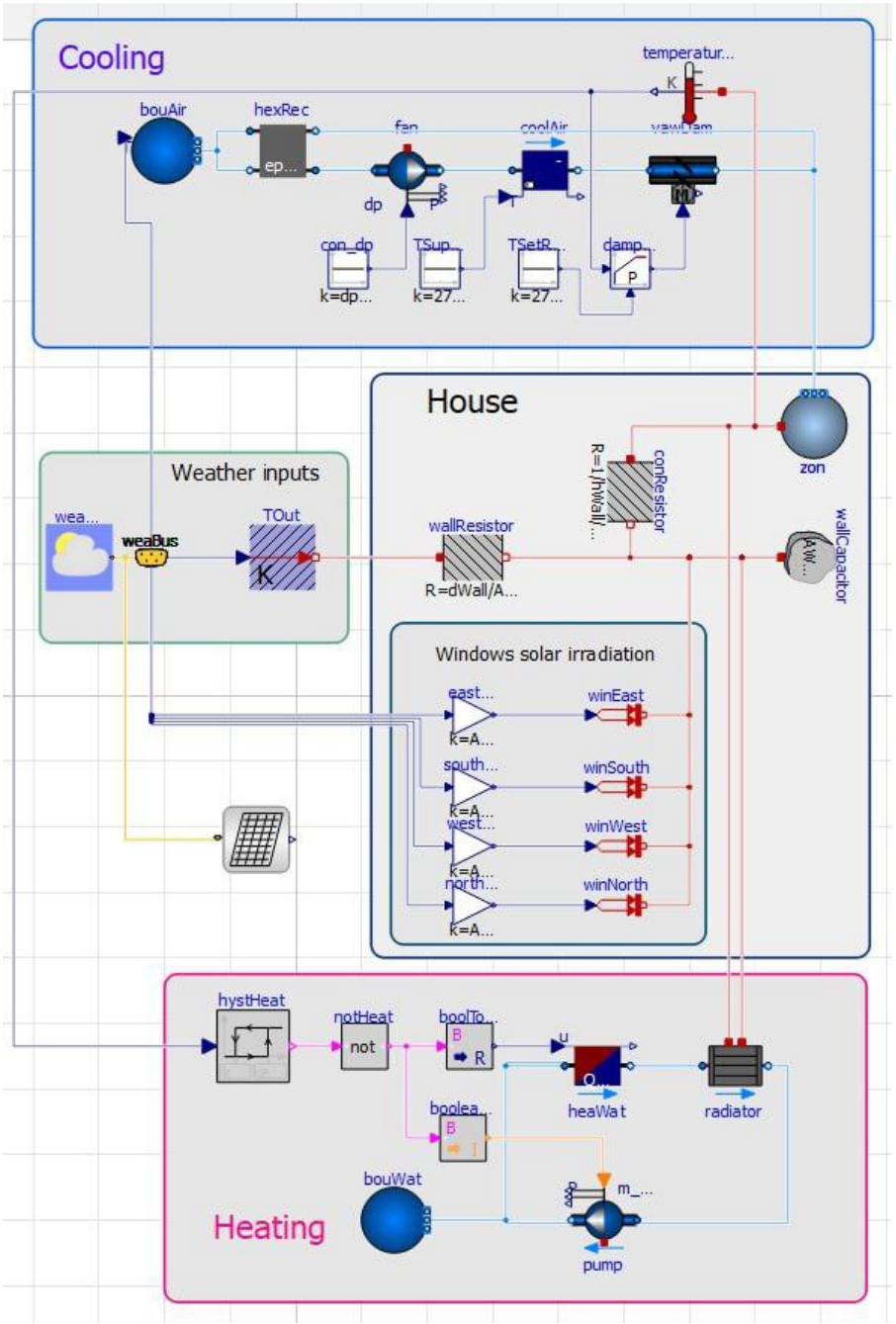


Figure 2.10 – Heating and cooling system model for a simplified house

where the other blue curve shows the power demand of the cooling fan. Finally, the green curve shows the power produced by the PV system.

Figure 2.12 shows the performance in more detail for different seasons. In January (a) the outside temperature is below zero and the heating system is working most of the time. To optimally use the self-produced PV power, the room temperature is automatically raised whenever the PV system provides extra power, which causes the recurring increases of the pump power demand and room

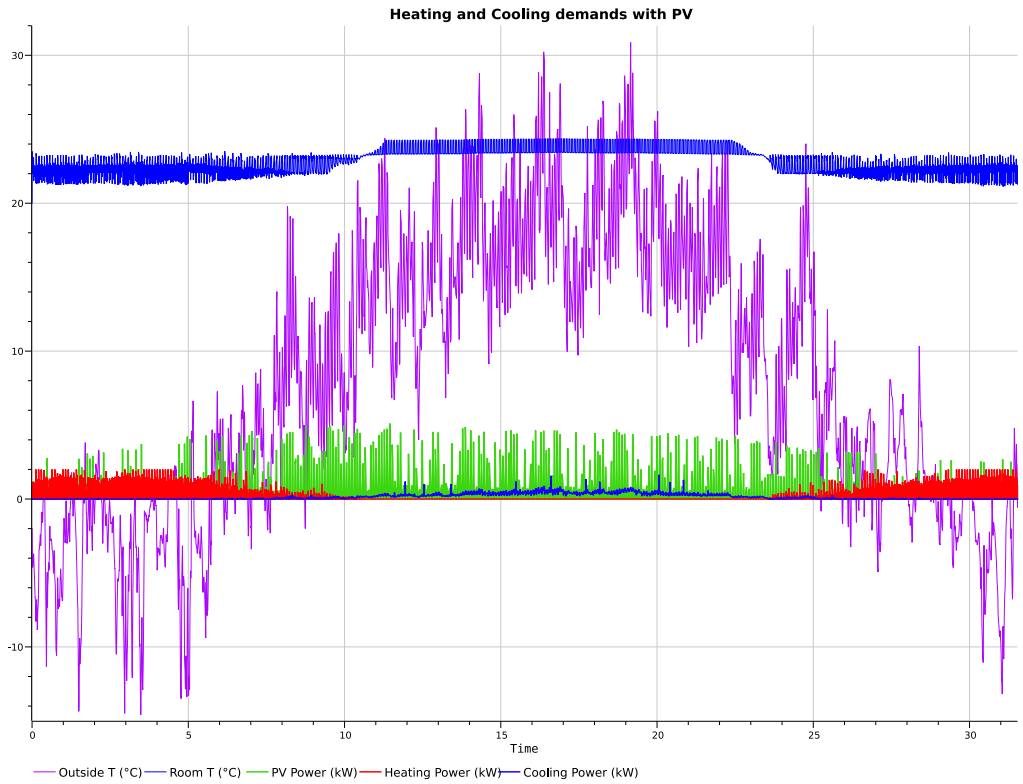


Figure 2.11 – Simplified heating and cooling across an entire year

temperature curve. In March (b) this effect is also present, but the overall power demand of the pump is less compared to January. In July (c) the cooling fan uses the self-produced PV power to automatically lower the indoor temperature, which causes the opposite pattern on a slightly higher average temperature level. In November (d) the heating system is again active and a behaviour similar to March but with considerably less PV production results.

2.4.2 Heat demand and control events

The heating process is complex because it involves delays and heat capacities. After activation, it takes some time until the heating system delivers the set power. Walls can store heat, but need long to heat up, and may be covered by isolating materials, e.g., furniture, tapestry, carpets. All this is covered by the parameterisation of the heated object and cannot be changed dynamically. What can change, is the outside temperature that defines the heat loss to the environment

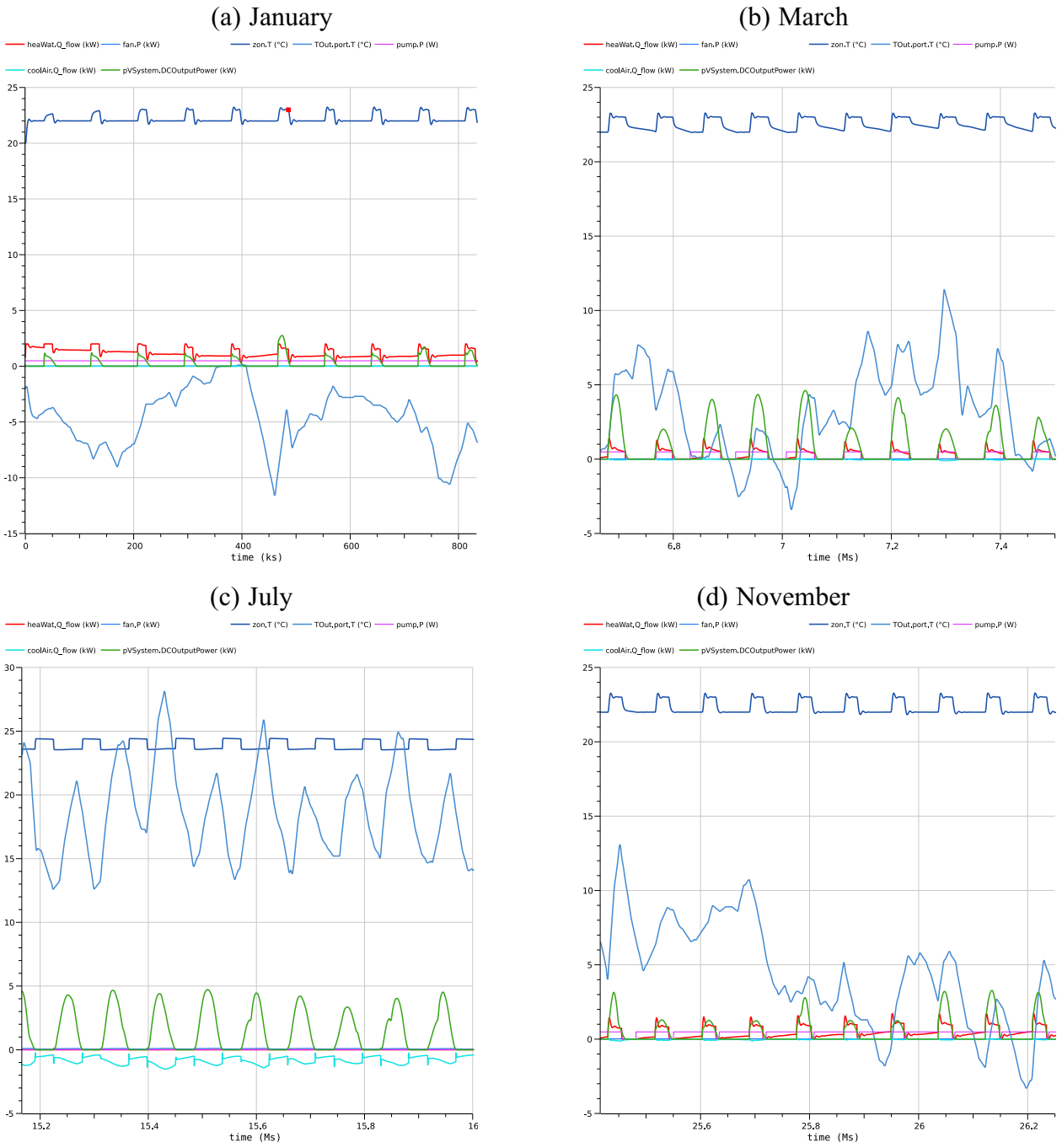


Figure 2.12 – Heating and cooling operation in four seasons

and the solar irradiation that heats the room through the windows. To compensate these dynamic changes, which are expressed by a varying heating energy demand, the parameterisation of the heating system can be adjusted dynamically to achieve for example Proportional-Integral-Derivative (PID) control.

For every heating system client, the following events can occur:

- *Set-Heating-Demand* to change the required heating energy
- *Set-Heating-Power* to change the delivered heating power

- *Set-Lower-Threshold* to change the activation temperature
- *Set-Upper-Threshold* to change the deactivation temperature
- *Switch-On-Heating* to activate the heating system
- *Switch-Off-Heating* to deactivate the heating system

Whenever an event causes a change, the heating system model is triggered: First, to calculate the current system state, and than to calculate the next required change and create the according event to be executed in the future. If that event already has been scheduled, it is replaced by the newly calculated. For example, if a Switch-On-Heating event already exists and a request to shift the upper threshold temperature arrives, the Switch-On-Heating event assigned to the affected heating system client becomes redone.

2.5 EV Charging Demand Modelling and Integration

Within the next decade, most households will become equipped with one or two AC wall-boxes, and office buildings with many such AC wall-boxes, eventually there will be one for nearly every parking lot. These millions of wall-boxes will be installed and controlled either by the owners or by energy providers that offer EV-charging-as-a-Service at reduced energy tariffs. The party that controls the charging can either benefit directly from the opportunities provided by a modern energy system that utilises and rewards flexible demand control, or can sell the charging control to an aggregator who trades the cumulated flexibility on diverse energy markets. Public charging sites, and places where EVs park for less long times, will provide fast DC charging, as evaluated in [40]. Demand control and participation in flexibility trading is in these cases rarely possible. In any case, the behaviour of EV charging infrastructures can be modelled as finite multi-server queueing systems, as explained in [43] and sketched in Figure 2.13.

A single wall-box with a single parking lot is the smallest possible finite multi-server queueing system, although with only one server and without a real

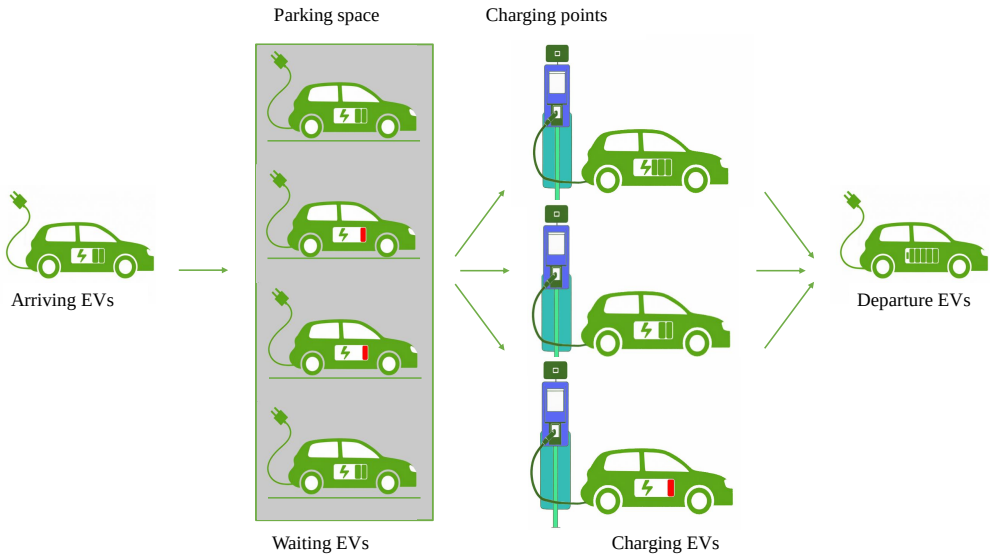


Figure 2.13 – EV charging infrastructure: finite multi-server queueing system [43]

queue. More wall-boxes (servers) and parking lots (queueing space) can be configured as needed to model a specific scenario.

In addition, shall the total power drawn from the local grid access point never exceed the contracted electricity grid access limit. Both, the queueing and the total power limiting, have been implemented in the *smartDCsim* tool developed in [43], which here is adapted to cover also slow AC charging and to enable the integration in the co-simulation framework.

2.5.1 Modelling of EV charging demand (re-)using smartDCsim

The arrivals and departures of EVs to and from a parking lot are differently distributed for different scenarios. Public charging is characterised by EV arrivals with varying likelihood across the day comparable to a load profile that can be derived from existing EV charging data in the region. The parking duration is wide spread, from less than an hour up to a full day or even more. Lacking information on what a random EV driver does while the EV is parked, we can model latter by a negative exponential distribution with according mean parking time. The arrival likelihood over the day correlates with the traffic density, as exemplarily sketched

in Figure 2.14.

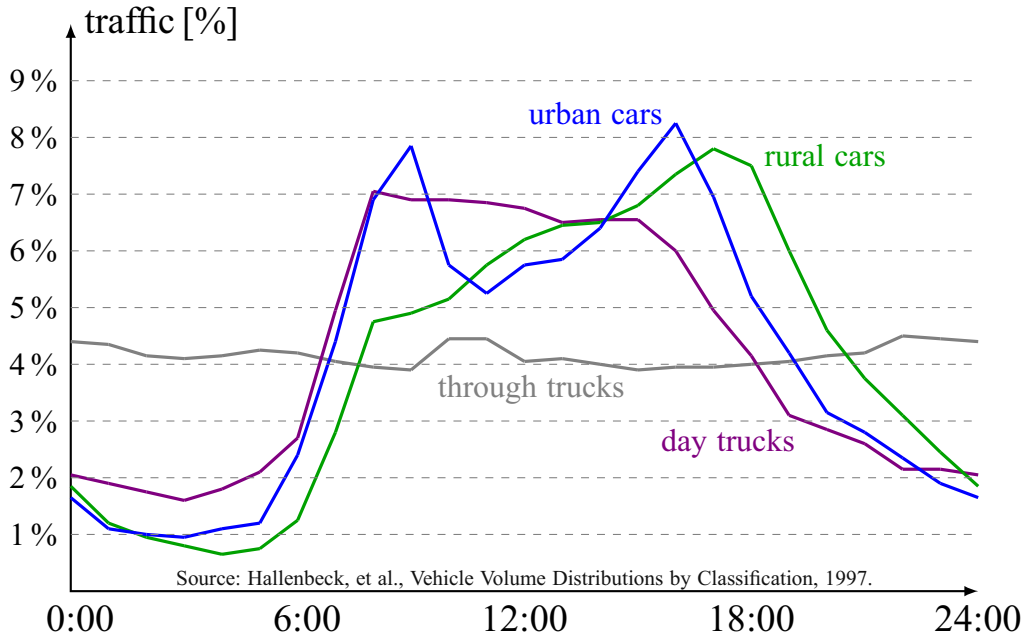


Figure 2.14 – Daily traffic pattern $\sim$ public charging load distribution

At home, for a private household, departures commonly occur in the morning and arrivals in the evening, and vice versa for office buildings where employees commute to work. No uniform pattern is possible, but for both scenarios most arrivals and departures are distributed in a more or less narrow time span, which can be approximated by an accordingly narrow distribution with a peak near the mean, e.g., Gamma distributed arrival and parking times. The departure time function D, characterising the time when an EV leaves the parking lot in relation to the lower bound of the arrival time, for example 0:00 *am*, results from

$$D = A \circledast T \tag{2.1}$$

where A and T are the arrival and the parking time distributions and $\circledast$ is the convolution operator. If we assume that the arrival and parking times are normal distributed around the expected time, we get

$$D = \bar{A} + N(\sigma_A^2) + \bar{T} + N(\sigma_T^2) = \bar{D} + N(\sigma_A^2 + \sigma_T^2) \tag{2.2}$$

where $N(\sigma_X^2)$ is the normal distribution around mean $E\{N(t)\} = 0$ with variance σ_X^2, and $\bar{X}$ is the expected time $E\{X(t)\}$ of the distribution X. In this case, the expected departure time $\bar{D}$ is the expected arrival time $\bar{A}$ plus the expected parking time $\bar{T}$, and the variance of the departure time σ_D^2 is the sum of the arrival variance σ_A^2 and the parking time variance σ_T^2 [49, Theorem 3.26 and 3.27].

The linearity of expectations, here $\bar{D} = \bar{A} + \bar{T}$, is always true, whereas (2.2) is true only if arrival and parking times are independent random processes. If for earlier than average arrivals the parking time tends to be longer than in average, the joint variance needs to be corrected by twice the covariance $cov\{A,T\} = E\{AT\} - E\{A\}E\{T\}$, which expresses the interdependence of the random processes A and T. In general, the variance of departures σ_D^2 results from

$$\sigma_D^2 = \sigma_A^2 + \sigma_T^2 - 2\,cov\{A,T\}. \tag{2.3}$$

For simplicity, we assume that EVs remain connected until the EV is driven away by the EV driver. This EV departure event defines the end of the charging process. Thus, T_{ch}^{EV} is a random sample from T, independent when and whether the EV becomes fully charged. When 100 % SoC is reached before the EV departs, the power demand drops to zero. In consequence, the charging power $P_{ch}^{EV}(t)$ is a dynamic parameter that is not constant throughout the random parking time T.

Also the initial SoC of independently arriving EVs is not constant and needs to be modelled by a distribution. Because the battery has finite capacity, the distribution is bounded to the interval 0 % to 100 % SoC. The *Beta* distribution provides this property and is therefore used. Depending on the scenario, a mean SoC of either 25 % is chosen for intentional charging events, when the EV is deliberately brought to the charging possibility, and 50 % for opportunity charging, where the charging opportunity is not sought for but coincidently available. The parameters α and β of the *Beta* distribution $B(\alpha,\beta)$ can be calculated from

mean μ and variance ν as follows:

$$\alpha = \mu \left(\frac{\mu(1-\mu)}{\nu} - 1 \right) \tag{2.4}$$

$$\beta = (1-\mu) \left(\frac{\mu(1-\mu)}{\nu} - 1 \right) \tag{2.5}$$

$$\text{for all} \quad \nu < \mu(1-\mu) \tag{2.6}$$

Figure 2.15 shows different distributions used to model (a) the time passing in between subsequent arrivals, the inter-arrival time T_A, (b) the parking duration that upper bounds the charging time T_{ch}, and (c) the SoC of arriving EVs, which yields the maximum charging capacity

$$E_{\text{ch}} = (1 - SoC) \cdot E_{\text{EV}}$$

where E_{EV} expresses the EV's usable battery capacity in kWh.

Using this modelling of the random time intervals and charging requirement, every arrival and departure of an EV defines a change event. These events yield the basis to simulate and analyse the stochastic performance and behaviour of charging sites, as presented in [43]. For the co-simulation, these events are the ones that trigger the evaluation of the integrated charging site module, and the flow-chart shown in Figure 2.16 differs only in the initial decision, where a flag set by the simulation control module ends the generation of subsequent events and not the event counter, as in [43]. The smart charging adjustment of all charging powers P_{ch}^{EV} to not exceed the total charging site power limit P_{max}^{CS} is here also included.

If an arriving client can be served instantly at its time of arrival, meaning that currently less clients are present than can be served in parallel, the departure event for this client can be generated with the execution time being the arrival time plus the random service time specified for the EV type. If all servers are occupied, a queueing event accommodates the arriving client in the waiting space.

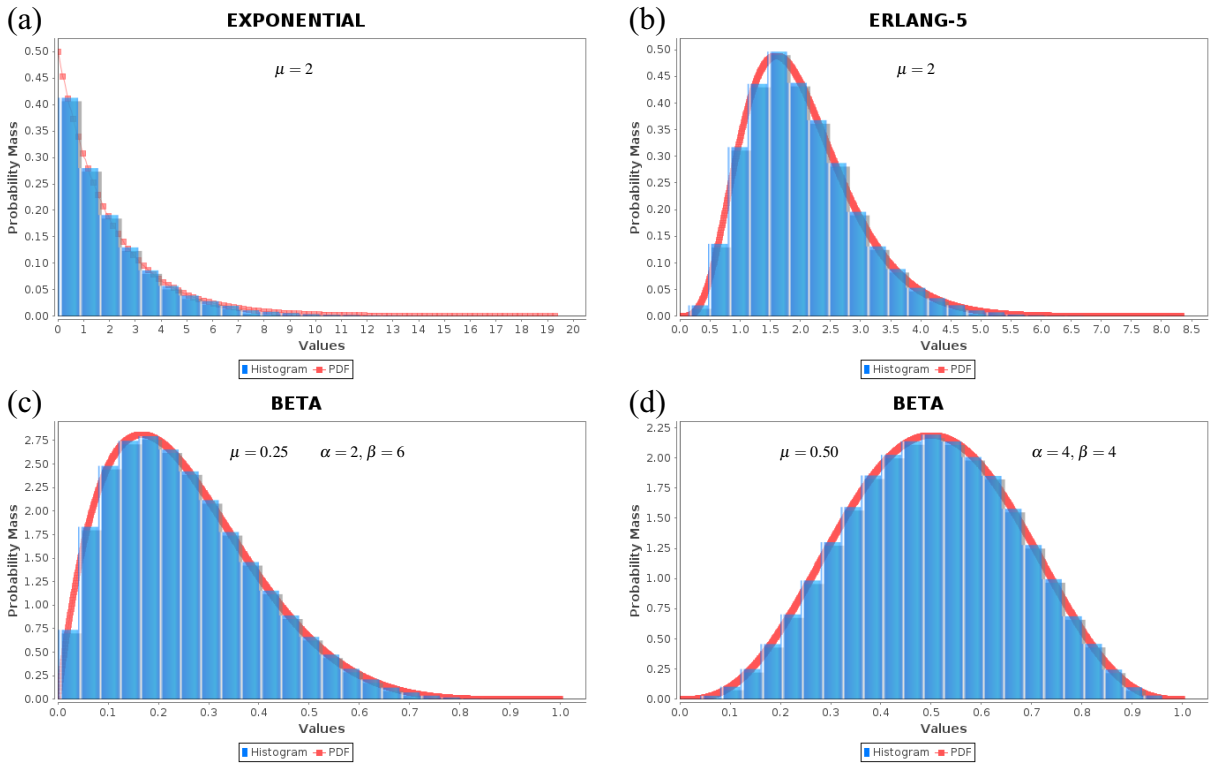


Figure 2.15 – Distributions: (a) negative exponential distributed EV inter-arrival times, (b) Erlang-5 distributed parking times, (c) and (d) Beta distributed SoC

As long as there is space in the queue, the system can manage eventual service after some delay. If the queue is full, a blocking event is generated and counted. In this case, both processing and waiting capacities are maxed out, and no further clients can be accommodated until the next departure.

A departure event indicates the conclusion of one client's charging process, and also the readiness of the system to serve the next client. The flow-chart shown in Figure 2.17 details the handling of departures. To assure that a sufficient SoC is achieved prior actual departure, checking the SoC and delaying the departure is added to the process implemented in [43] to mimic EV drivers' behaviour.

When the current client leaves, the times the client spent waiting and being served become recorded, and the now useless data object is discarded. If some other client is currently waiting in the queue, it is picked from the queue and a new departure event is created. Thus, service fluidly transitions from one client to the next. If no client is waiting in the queue, the server that finished the departing client becomes idle, ready to serve the next arrival whenever that occurs.

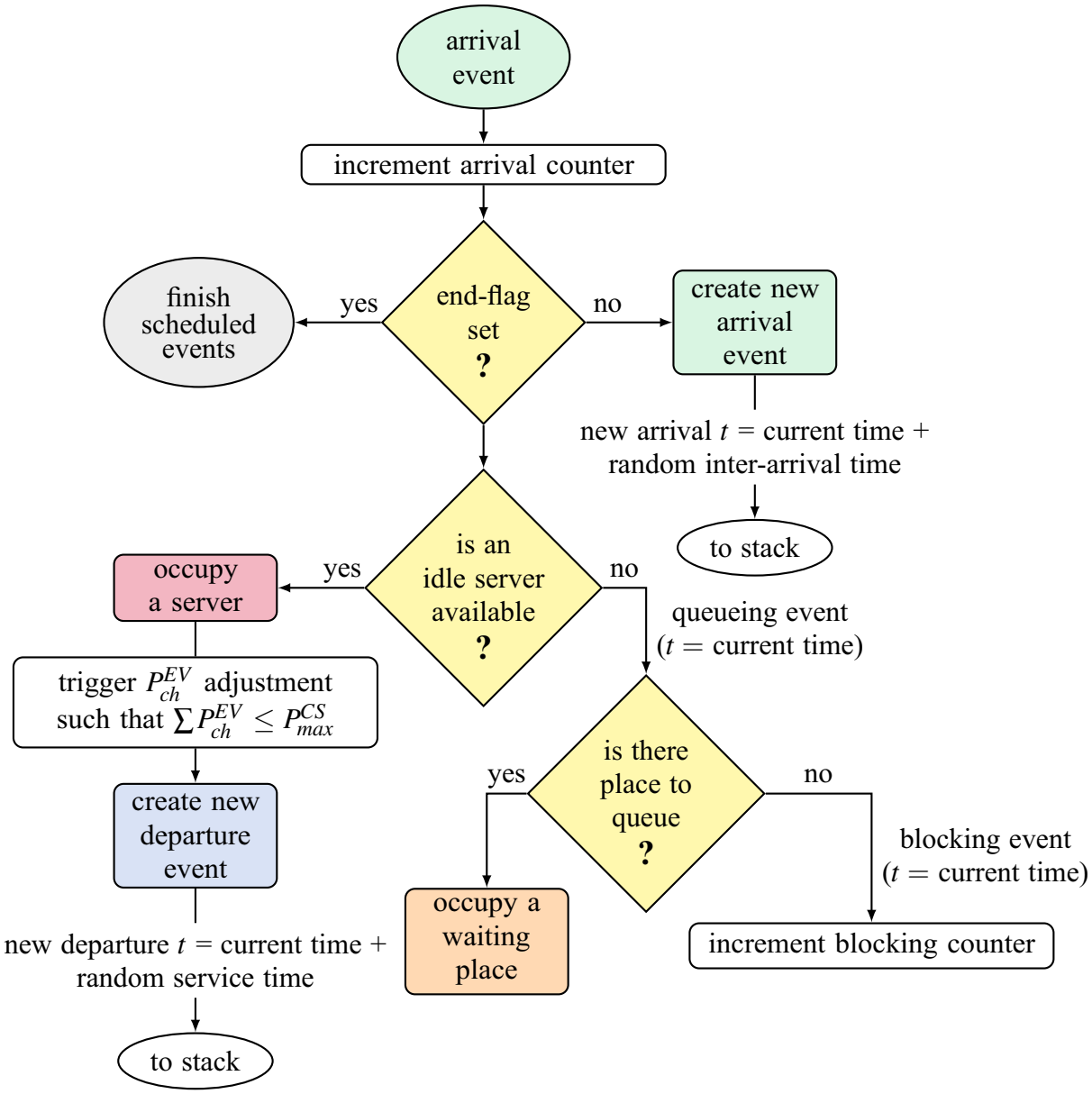


Figure 2.16 – EV arrival event flow chart [43, adapted]

2.5.2 EV charging demand and charging control events

For cross-sector coupling and inter-operation, the **dynamic site power limit** adjustment is needed. This event can be triggered by external modules only and indicates that the site power limit shall be set anew to the value or percentage handed over. This event triggers the chain of actions displayed in Figure 2.18.

The total power setting can occur in two different ways: Absolute, if a new P_{max}^{CS} or a percentage of the physical limit is requested, or relative to the current

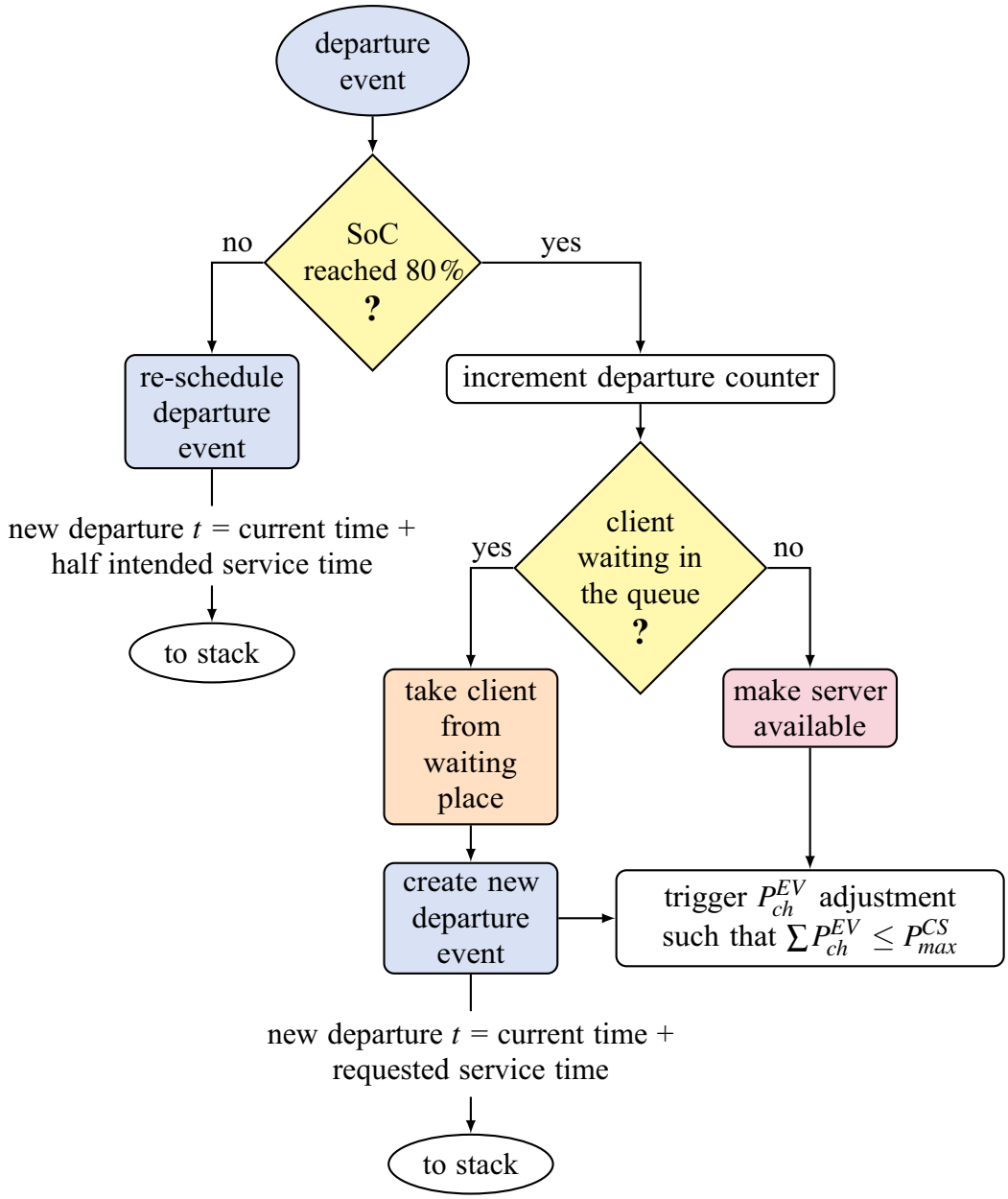


Figure 2.17 – EV departure event flow chart [43, adapted]

P_{max}^{CS}, if a ΔP is requested.

A new absolute P_{max}^{CS} or absolute percentage is commonly used to curtail the maximally possible power. If a percentage is requested, the new P_{max}^{CS} is the according fraction of the site's physical limit, independent which P_{max}^{CS} has been set before. A relative ΔP or relative percentage is commonly requested if flexibility shall be provided to support grid stabilisation. In that case, the change needs to be applied relative to the current P_{max}^{CS}. If that is not feasible, meaning the resultant new P_{max}^{CS} becomes smaller then zero or exceeds the physical limit, the

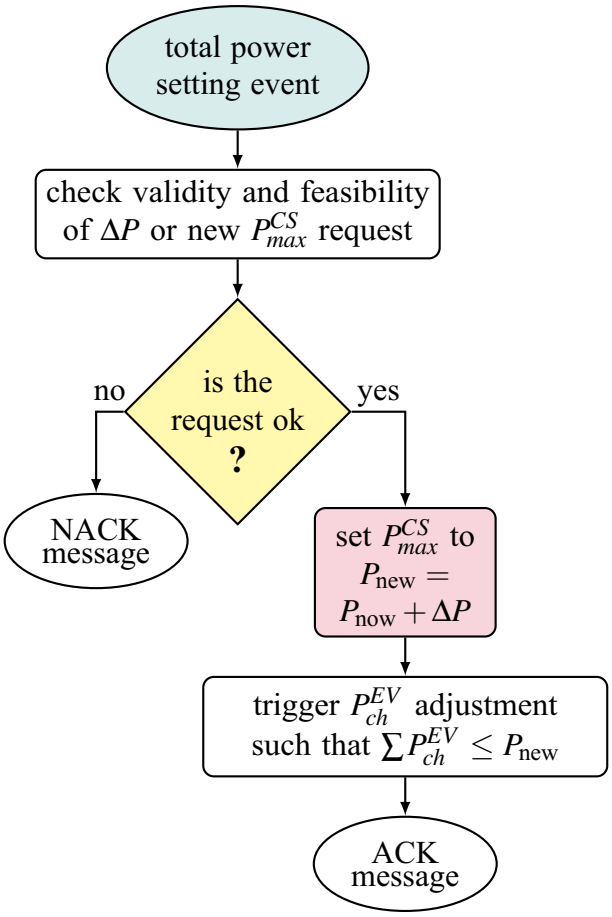


Figure 2.18 – Site power limit $P_{max}^{\text{CS}}(t)$ setting event flow chart

request needs to be rejected and a not acknowledged (NACK) message returned. In all other cased, if the request is eligible and executable, the request shall be acknowledged (ACK).

Whenever the P_{max}^{CS} is changed all the currently progressing charging processes need to be re-evaluated and adjusted to meet the requested new P_{max}^{CS} as good as possible and never exceeding it.

2.6 Conclusion of Section 2

The evaluated tools, *pandapower*, *OpenModelica* and *smartDCsim*, are adequate to perform the intended tasks to evaluate smart power control algorithms, as for example sketched above, that couple the different sectors and make them inter-operate, meaning digitalise them to support a modern integrated energy

system. The intended simulation task sketched in Figure 2.19 appears solvable with these tools. Their integration into an event driven simulation environment is however challenging.

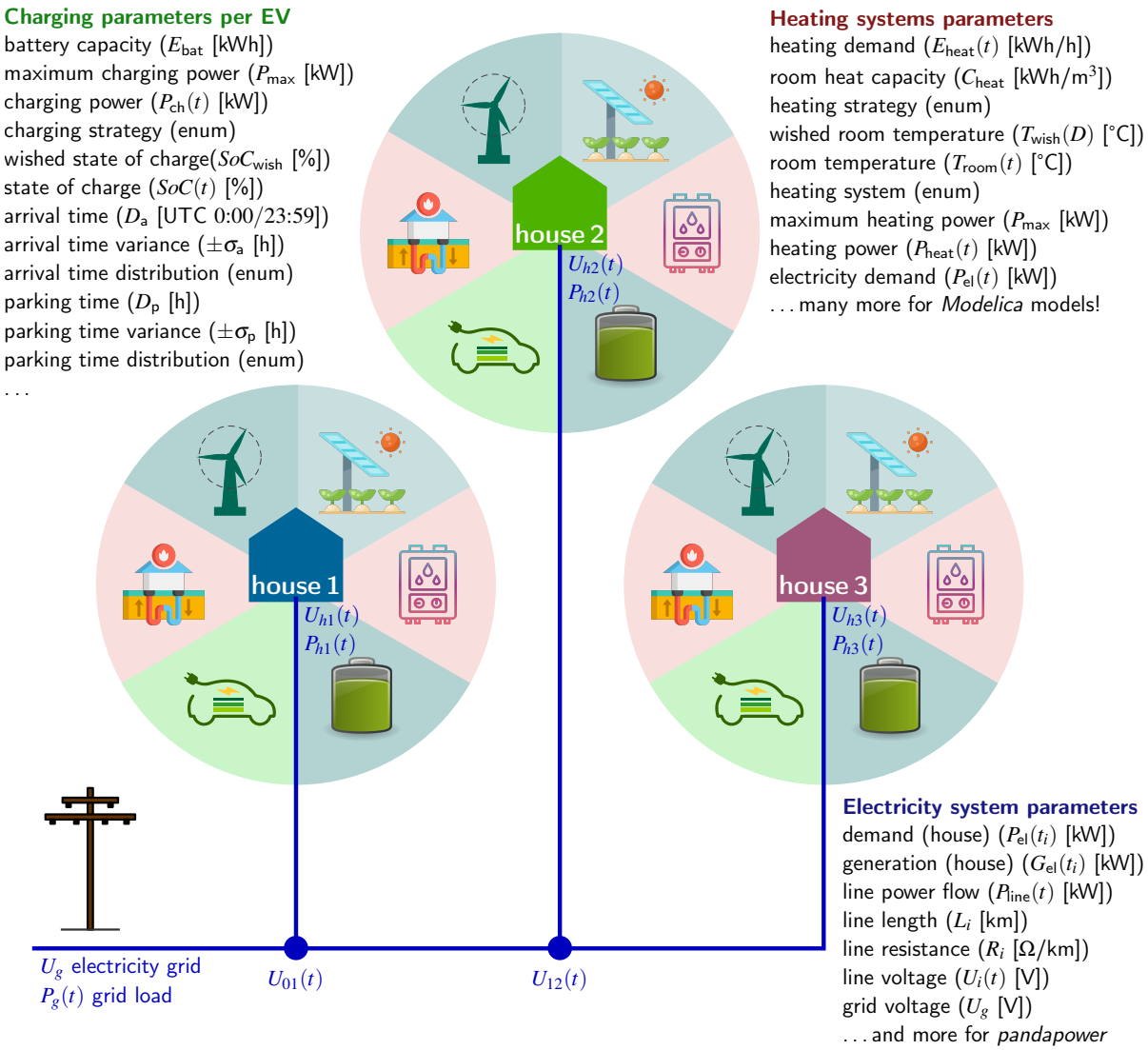


Figure 2.19 – Real system layer parameters (three houses scenario)

The easiest to integrate is smartDCsim because it is designed to use an event based simulation core that executes the simulation, whereas pandapower and OpenModelica are clocked simulation tools based on a constant clock interval. To integrate the latter, the simulation of the subsystems needs to be performed for the time spans between scheduled events that affect the power flows. In case power changing events become scheduled in between existing events, the simulation of the affected time span needs to be repeated to consider the new event, which yet

appears feasible and doable but poses the risk of looping.

To perform the simulation of the ’micro’ time spans in between power changing events, we need to develop individual simulation control modules for each integrated tool, which reside in the simulation control block. These modules, designed and described in Chapter 3, handle the events managed by the unifying simulation core and trigger the simulation of the according time spans whenever needed. They shall also analyse the simulation result to create new events in case the simulation result reveals that an adjustment of the power flow is needed in accordance to some smart control algorithm that shall be evaluated.

Each tool simulates its part, the power flows on electricity grids, the energy demand and performance of heating and cooling systems, and the power usage for smart EV charging, respectively. The linking is performed by the event core and the individual simulation control modules that manage and execute the integration of the individual simulation tools, as outlined in the next chapter.

3 SOFTWARE DEVELOPMENT

The software development process is critical for an effective and efficient modelling and consequential analysis. As we transition from the theoretical basics and practical applications outlined in the previous sections and summarised in Figure 2.19, this Section 3 delves into the technical specifics of the framework's back-end components, the developed interfaces (APIs), the integration of existing simulation tools (wrappers), and the statistical analysis and visualisation integrated in the simulation framework's front-end. Finally, we explore potential extensions and adaptations, laying the groundwork for future development, approving the scalability and extendability of the developed framework for continued adaptation.

3.1 Layered framework architecture

Starting from Figure 2.19, which outlines an exemplary real scenario, we develop the layered architecture of the simulation framework. The simulation task outlined in Chapter 1 and detailed in Chapter 2 is extended into a layered structure of different modules shown in Figure 3.1. Whereas the definition occurred top-down, as detailed below, the implementation happened more bottom up, starting with the implementation of a robust simulation loop in the back-end, followed by the wrappers that control and thereby integrate the different simulation tools. On top, user interface and the statistical evaluation, visualisation and saving of results, provide the front-end of the simulation framework.

Simulation Framework Layer: Configuration, Execution, Results

On this layer, the user interacts with the simulation framework's front-end. Using the GUI, pre-defined scenarios can be loaded and their simulation triggered. The execution is controlled by the Simulation Control module, which is the main

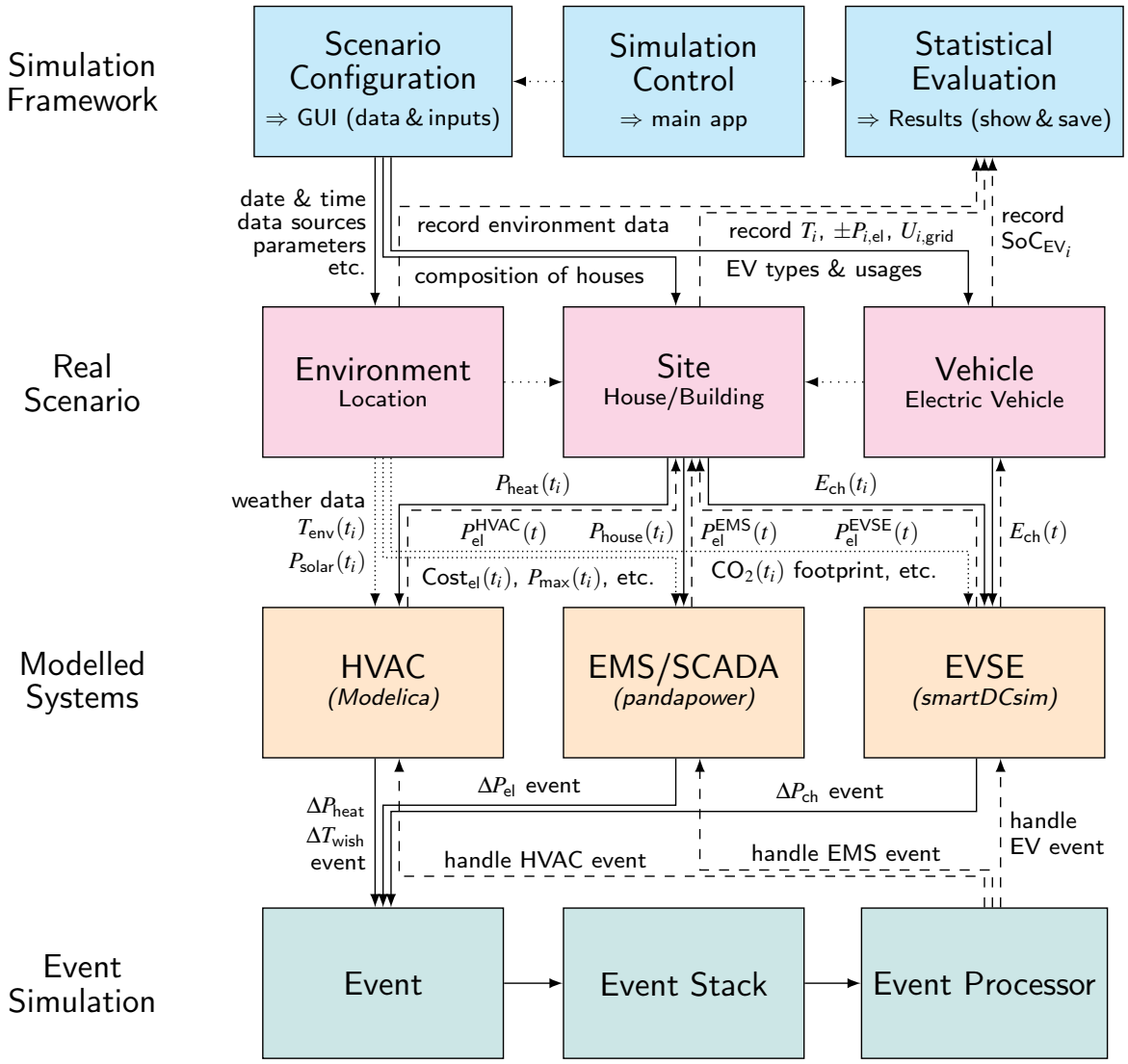


Figure 3.1 – Simulation framework layers from user interaction and simulation control to configured scenario and modelled systems down to the simulation core

application of the framework's back-end that is directly linked to the front-end. Preliminary and final results become visualized to provide instantaneous feedback. To compose scenarios, an optional Scenario Builder tool might be provided.

The GUI briefly sketches the selected scenario so the user is assured to have selected the correct scenario. Clicking the "Start Simulation" button in the front-end, initializes the required modules in the back-end, which pushes all initial events into the Event Stack. After a successful initialisation, the Event Processor starts to execute events until the Event Stack is empty or a configured simulation duration is reached.

In between simulation steps, which cover a specified time duration (hours, days, months) with a given monitoring granularity (seconds, minutes, hours), selected environment parameters can be changed to initialise a slightly different simulation run of the same scenario, or to repeat the simulation *n* times with newly randomised volatile parameters for a more detailed statistical evaluation. Adjustment of set parameters can be done manually, using the GUI, or automatically, controlled by a parameter sequence matrix that holds the changing parameters in sequential order to perform consecutive simulation steps.

Monitored time series and statistical metrics achieved per step are displayed in the graphs provided in the front-end. Thereby, the impact of the performed parameter change, or the statistical averaging, becomes conveniently displayed step-by-step. Which parameters shall change and which results shall be displayed, is configured as part of the scenario specification. In the end, when all steps have been completed, all results, being the sequences of achieved time series and metrics and their visualisation, can be saved individually, as comma separated values (CSV) and scalable vector graphics (SVG), as well as jointly, in a dedicated folder including also all input parameters for consistent documentation and archiving, for example by zipping the dedicated folder whenever the results appear worthy. The default file names shall be composed of the scenario name, the contained result or metric, and the current system date and time. Latter is added by default to prevent unintended overwriting of results.

Real Scenario: Environment, Houses/Buildings/Sites, EVs

The modules on the second layer describe the scenario that is simulated as it may exist in real. In the centre is the **Site** module, typically representing a house or building, but it may as well specify a single flat or an entire district. Commonly, a simulation scenario consist of several differently composed sites. The individual composition of a site specifies different modules from the layer below:

- a general electricity demand model (load profile and variance),

- models for electricity generation and storage, for example, a PV system and battery electric storage systems,
- a model for the electric vehicle supply equipment (EVSE) used to charge occasionally present EVs, and
- a model for the heating and cooling demand including the used HVAC systems to fulfil the demand.

The Site object itself provides static information about the site:

- the geographic location (to select the correct weather data) and
- the electricity grid access point (node ID in the modelled grid).

The provided information is used to select and configure the system models used to simulate the different appliances and assets, and the grid load put on the local supply line of the feeder network. The configurable parameters for the different systems related to a site are summarised later on in Section 3.3.1 in Table 3.1.

Every Site resides in some **Environment**, which specifies the external influences that affect the modelled systems of a Site object. The most relevant environmental impact factors specified by this module are:

- weather data (recorded data from a file) and
- electricity grid constraints (state of the grid and thresholds to respect).

Commonly, one Environment module specifies the conditions of several locations, typically along an electricity supply line or covering the area supplied by a low voltage transformer. The weather data for such a confined area can be assumed identical for all covered sites, whereas the electricity supply conditions depend on the position of the site along the feeder network modelled by *pandapower*.

The **Vehicle** module on the other side represents electric vehicles that may be charged when present at a Site. The specification of a Vehicle object includes:

- Site-ID to which the modelled EV belongs, it's 'home' EVSE,
- mean and variance (distribution) of the EV's arrival time $t^{\mathrm{EV}}_{\mathrm{arrive}}$,
- mean and variance (distribution) of the parking duration $d^{\mathrm{EV}}_{\mathrm{park}}$,
- mean and variance (distribution) of the charging demand $E^{\mathrm{EV}}_{\mathrm{wish}}$,
- the EV's charging curve $P^{\mathrm{EV}}_{\mathrm{max}}(SoC)$ or a guessed approximation.

These parameters are also included in Table 3.1 in Section 3.3.1. In principle, this module can be used for any energy demanding application that is only temporarily

present. Here, we use it exclusively for EVs. Note also, that the charging point (CP/EVSE) is a component of the Site object, because it is always there. In contrast thereto, is the EV mobile and hence not always connected to the site. Theoretically, and only at different times, might an EV use the CP/EVSE of different sites. Here, we assume that EVs are bound to a particular site (myHome) and become charged at this site only.

Modelled Systems: EMS/SCADA, HVACs, EVSEs

Specialised tools become integrated in the simulation framework to model the different energy consuming appliances from the electricity, the heating and cooling, and the mobility sector. In particular, *pandapower* [19], *Open Modelica* [38], and *smartDCsim* [43] are integrated in the initial version. These tools provide the modelling of the real systems that are specified in the layer above. If these system models would be controlled by events triggered by a real system, instead of the events handled in the simulation core below, we would call them *digital twins*.

The integration is performed by either using APIs or command-line execution requests, latter executing the called system model request-by-request, handled by a wrapper instance. The required model configurations are automatically created and updated in between calls. Variable adjustments are performed in the Simulation Control module, where also the output of the tools is monitored and processed before it and derived variables are made available for display in the front-end. Based on the interim simulation results received from the different tools, new Events are created as required by the smart power management algorithms designed and intended to be evaluated, for the same system model as well as for adjacent systems. Thereby, the different simulation tools become integrated into a multi-energy or multi-vector simulation framework.

The physical system that links all sites, is the local electricity grid (feeder network), consisting of the low voltage transformer and the feeder lines that connect the sites. This is commonly managed by a so called System Control And

Data Acquisition (**SCADA**) system, whereas the loads and sources within a site (behind-the-meter) are managed by an Energy Management System (**EMS**). The configuration and state of the grid and each site are reflected by a data structure, a *panda*. This is the basis for the simulation of the power flows using *pandapower*.

Changes in the panda are caused by Events that trigger a recalculation of the power flows for a specific time period. The power-flow result, in particular the voltage levels on the feeder network, can trigger follow-up events that change parameters according to the control mechanisms (algorithms) implemented by linking the SCADA system with the sites' EMS to maintain and manage the power flows to and from sites. These control actions are in general threshold based responses. Thus, the follow-up events' execution time is the time at which a pre-configured threshold is exceeded.

Heating, Ventilation, and Air Conditioning (**HVAC**) systems are among the largest energy consumers. Their electrification improves energy efficiency, but also causes locally synchronised demands due to the high dependence on local weather conditions. Heat is buffered in every matter, and therefore, every volume of air confined in a room, every wall, and every piece of furniture has a specific heat capacity that causes nonlinearity. In addition, show surfaces (contact areas) and bodies (volumes) heat reflection and conduction properties. *Open Modelica* sets up and solves the resultant system of differential equations, very eloquently.

Scenarios can be created in the front-end or loaded from predefined configuration files (see Appendix C Listing 1). If the impact of a specific configuration parameter shall be analysed, step-by-step scenario adjustment can be configured (*parameter sequence matrix*), such that the simulation is repeated and the impact of the changes become visualised. Also in the GUI of the front-end, the user specifies the parameters that shall be monitored and visualised at the end of every simulation step.

While executing a simulation step, interim system states can be saved by the Simulation Control module, such that these can be reloaded to continue the HVAC system simulation with adjusted parameters from an exact point in time on, considering also potentially ongoing transitive effects.

Event Simulation: Events, Event Stack, Event Processor

The event simulation core is the heart of the simulation framework. All the changes that occur somewhere in the simulated scenario originate from an event in the event simulation core. It is the centre of all action and sets the global simulation time to the execution time of the currently handled event. It shall be maximally simple and robust to work efficient and independent of the Event execution performed somewhere outside of the simulation core.

Accordingly, specifies the basic **Event** class tiny data objects (Events) that hold essential information only:

- *Execution Time* — scheduled time of the change execution
- *Change Amount* — size of the change that becomes executed
- *Home Module* — module that needs to execute the change
- *Event Client* — object that is affected by the change
- *Event Type* — kind of change that becomes executed

The parameters may be extended as needed by different systems modelled on the layer above. Therefore, siblings derived from the basic Event class shall be specified by the according modules to server their needs.

The **Event Stack** is a sorted list that stores all Events that yet have been created but not executed. Stored Events are sorted by their Execution Time t_j. New Events can be created and pushed into the Event Stack with any Execution Time $t_j \geq t_i$, where t_i is the current simulation time equal the Execution Time of the currently executed Event. If $t_j = t_i$, the new Event will be executed instantly after the currently executed Event.[1] If multiple Events with $t_j = t_i$ are created, they are executed in the order determined by the sorting algorithm of the list element used to create the Event Stack.

The **Event Processor** picks the next event from the Event Stack the moment the previous Event's change has been executed. First of all, it calculates the time that passed since the last Event δt and than sets the simulation time t_i to the

[1] An alternative approach to the implemented first-in-first-out regime would be pause-and-resume execution of Events in case new Events to be executed instantly are triggered, which would yield a last-in-first-out regime.

Execution Time $t_j(nextEvent)$ of the newly received Event:

$$\delta t = t_j(nextEvent) - t_i \tag{3.1}$$

$$t_i = t_j(nextEvent) \tag{3.2}$$

To be commonly accessible, t_i and δt are maintained in the Simulation Control module but calculated and set by the Simulation Core.

After the simulation time has been updated, the newly received Event is handed over to the module identified by the *Home Module* pointer to become executed. And after the Event has been executed, the executing module (*Home Module*) shall trigger the Event Processor to get the next Event from the Event Stack. This loop continues until the Event Stack becomes empty, which indicates the end of a simulation step.

3.2 Timing, calculation progression and pre-calculation horizons

The Heating, Ventilation, and Air Conditioning (**HVAC**) systems modelled by Modelica are simulated for ten day intervals irrespective of the time period between scheduled events. Whenever parameters of a HVAC system become changed or the next scheduled event occurs beyond the pre-calculated ten day interval, the simulation in Modelica is re-executed for again ten days, starting 24 hours prior the new change event or the end of the previous ten day interval. Parameter changes due to new change events are added and scheduled to happen at the correct time considering the 24 hours lead time.

Calculation parameters saved from the previous simulation run are used to seamlessly continue the simulation of the non-linear system. This, and the 24 hours lead time, are necessary to mitigate the initial ramp-up of the highly capacitive heating and cooling of the different heat capacities (masses) involved. These first 24 hours become stripped from the simulation result whenever Modelica returns results to the Simulation Control module.

The calculated power demand of the HVAC system across the simulated time period is used by the Simulation Control module to adjust the HVAC power variables in the according *panda*, which triggers the re-calculation of all power flows across the simulated feeder network.

Every site's Electric Vehicle Supply Equipment (**EVSE**) infrastructure is individually modelled by an independent *smartDCsim* instance to model the local smart EV charging as it is performed per site by the present EVSEs with one or more charging points. Here we use APIs to control the modelling, being the stepping from one scheduled Event to the next. JSON configuration files are used to initialise each EVSE infrastructure model, including the parameters of the EVs that are assigned to a site's EVSE (myHome).

Whenever an EVSE model related Event is handed over to the according smartDCsim instance, the charging power and SoC of all EVs currently connected are calculate. Next, follow-up Events related to the Site, the modelled EVSE, or the charged EVs, are created and executed as needed.

The resultant site power demand of the EVSEs currently in use, is returned to the Simulation Control module, where it changes the EVSEs power variable in the according *panda*. This change then triggers the re-calculation of all electricity power flows across the modelled feeder network. This can again trigger a charging power adjustment Event, which is instantly executed by the EVSE to keep all power flows within save limits at all times.

3.3 Modules, objects and classes

In this section, we delve into the modules and objects composition and the interplay between the different classes and objects of the simulation control module and the event simulation core, before we immerse into the tailored APIs (application programming interfaces) and message processing required to integrate the different simulation tools used to determine electric power flows, EV charging performance, and heating system demands.

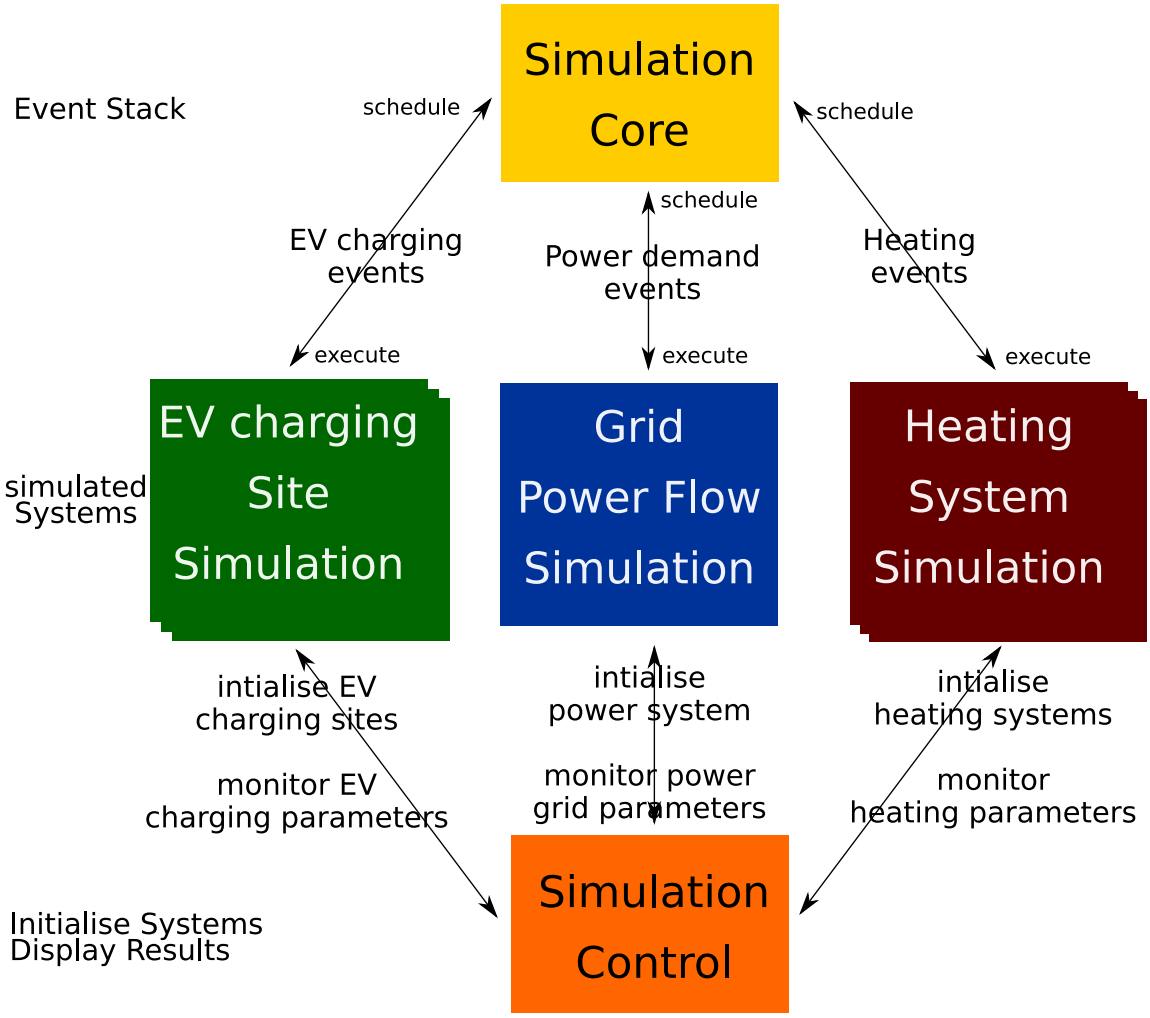


Figure 3.2 – Modules architecture linked by APIs and exchanged messages

Every square in Figure 3.2 is an independent instance of a module. Each module could be done in a different programming language. Simulation Control, Simulation Core, and the Grid Power Flow Simulation exist only once, whereas any number of EV Charging Site Simulation and Heating System Simulation modules can exist in parallel. For ease of implementation, the Simulation Core can be embedded in the Simulation Control, but in general, there shall be no direct link between them, as indicated by the separation shown in the figure.

Like other modules, can Simulation Control push Events into the Simulation Core. For example, initial events for each simulated system, parameter monitoring events, pre-scheduled parameter change events, and the end simulation event.

3.3.1 Initialisation of a simulation run

The simulation of a scenario is based on pre-configured files. Offering a Scenario Builder GUI to individually edit these files in a consistent and intuitive manner would be great but causes a lot of routine programming effort and thus, is considered optional. Plenty of tools exist to edit JSON files and alike.

First and most central, specifies the **simulation configuration** the

- scenario, by listing the sites and the systems these contain,
- predetermined parameter changes (for example the weather),
- parameters to monitor, display, and compare (step-by-step),
- sequence of configuration changes that shall occur step-by-step.

For example, a scenario can be composed of three sites (houses), one-by-one connected to the same electricity feeder network with access power limits being $6\,kW$, $30\,kW$, and $100\,kW$ for the sites 1 to 3 respectively. **Site 1** contains a $12\,kWp$ PV system with attached $12\,kWh/3\,kW$ battery storage, a single $5.4\,kW$ EV charging station for the owner's EV, and a $4.2\,kW$ heat-pump extracting heat from the outdoor air. The EV is used to commute to/from work every morning/evening. The wished indoor temperature is set to $22\,°C$ from 15:00 to 21:00 and $20\,°C$ else. **Site 2** contains a $45\,kWp$ PV system but no electric heating system. It is a small office building providing 10 EV charging points for 6 employees and the remaining 4 for guests/clients/visitors. **Site 3** is a multi-storey apartment building with 24 flats, a $120\,kWp$ roof PV and a parking area that supports charging of 24 EVs, one per flat, with pre-assigned charging points pre flat (no choice of charging point). Heating is based on a central $50\,kW$ heat-pump using geothermal heat extraction. The average indoor temperature setting is $23\,°C$, Beta distributed in the range $18\,°C$ to $25\,°C$ and in average constant across day and night.

The parameters listed in Table 3.1 are either loaded from file (see Appendix C Listing 1) or configured in the GUI, as sketched in Figure 3.3.

The definition of the different parameters and variables summarised in Table 3.1 is provided in the according sections on the different systems and their modelling. More parameters than those listed are in general required to correctly

Table 3.1 – Configuration parameters of modelled systems (Figure 2.19)

System type	electric power flows	heating system	EV charging
Parameters	**grid parameters** grid topology $[n \times n]$ $L_{ij}[km]$, $R_{ij}[\Omega/km]$ $U_i(t)[V]$, $U_0^{\mathrm{LV}}[V]$ $\delta E_{ij}(t)[kWh/h]$ **site parameters** location $[long.\ \&\ lat.]$ std. load profile $[enum]$ $P_{\mathrm{load}}^{\mathrm{el}}(t)[kW]$, $\pm\sigma_l[kW]$ *load* distr. $[enum]$ $E_{\mathrm{bat}}[kWh]$, $SoC_{\mathrm{bat}}(t)[\%]$ $P_{\mathrm{max}}^{\mathrm{bat}}[kW]$, $P_{\mathrm{bat}}(t)[kW]$ **PV parameters** $P_{\mathrm{peak}}^{\mathrm{PV}}[kWp]$, $P_{\mathrm{gen}}^{\mathrm{PV}}(t)[kW]$ $azimuth[^\circ]$, $slope[^\circ]$	**site parameters** $T_{\mathrm{out}}(t)[^\circ\mathrm{C}]$, $T_{\mathrm{in}}(t)[^\circ\mathrm{C}]$ $T_{\mathrm{wish}}^{\mathrm{in}}(t)[^\circ\mathrm{C}]$ $\delta E_{\mathrm{heat}}(T_{\mathrm{out}} - T_{\mathrm{in}})[kW]$ $\delta E_{\mathrm{sol}}(I_{\mathrm{sol}}, t)[kW]$ $C_{\mathrm{heat}}^{\mathrm{site}}$, $C_{\mathrm{heat}}^{\mathrm{tank}}[kWh/^\circ\mathrm{C}]$ $P_{\mathrm{tank}}^{\mathrm{el.}}[kW]$, $T_{\mathrm{tank}}(t)[^\circ\mathrm{C}]$ heating strat. $[enum]$ **HVAC parameters** heating system $[enum]$ *heat* source $[enum]$ $C_{\mathrm{heat}}^{\mathrm{HVAC}}[kWh/^\circ\mathrm{C}]$ $P_{\mathrm{HVAC}}^{\mathrm{el./heat}}(t)$, $P_{\mathrm{max}}^{\mathrm{el./heat}}[kW]$ +many model param.	**EVSE parameters** $n(CP)[n]$, $P_{\mathrm{CS}}(t)[kW]$ $P_{\mathrm{max}}^{\mathrm{CP}_i}[kW]$, $P_{\mathrm{max}}^{\mathrm{CS}}[kW]$ smart charging $[enum]$ **EV parameters** $P_{\mathrm{ch}}^{\mathrm{EV}}(t)[kW]$, $E_{\mathrm{bat}}^{\mathrm{EV}}[kWh]$ $P_{\mathrm{max}}^{\mathrm{EV}}(SoC)[kW]$ $SoC(t)$, $SoC_{\mathrm{wish}}(t)[\%]$ $D_{\mathrm{mean}}^{\mathrm{arrival}}[UTC]$, $\pm\sigma_{\mathrm{a}}[h]$ arrival distr. $[enum]$ $D_{\mathrm{mean}}^{\mathrm{park}}[h]$, $\pm\sigma_{\mathrm{p}}[h]$ parking distr. $[enum]$ $D_{\mathrm{mean}}^{\mathrm{SoC}}[\%]$, $\pm\sigma_{\mathrm{SoC}}[\%]$ SoC distr. $[enum]$

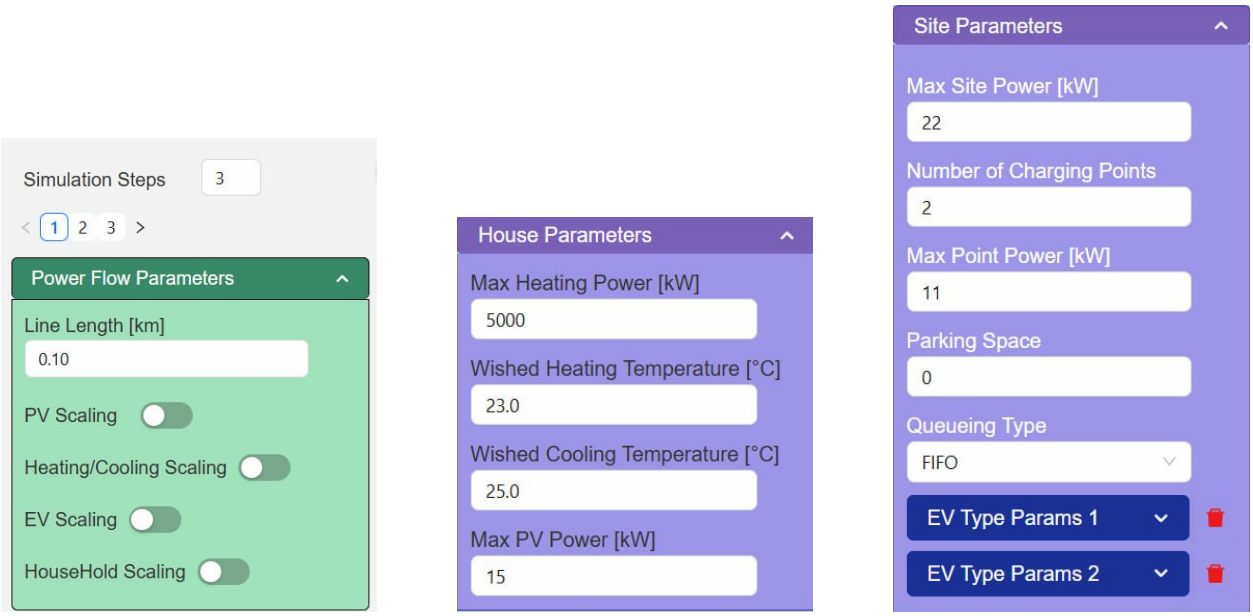


Figure 3.3 – Site-by-site module-by-module scenario configuration in the GUI

model a particular system. For example, the minimal thermal mass flow needed by a heat pump before it can become operational. Such system specific parameters are specified in the according model configuration file and not considered changeable or available for monitoring.

Weather data shall be provided as time series in the HVAC module because it is relevant for heating and cooling, but also for PV systems. If we need weather

data for modelling the temperature dependence of the EV charging process in the EV charging module, the simulation control module shall obtain this data from the HVAC system to integrate it.

3.3.2 Run a simulation

Before a simulation run can be executed, all involved modules need to be started. The first module started, is the Simulation Control module, being the main app, and the Simulation Core module. Simulation parameters are set via GUI and then transformed into the configuration file.

One-by-one, the configured modules are started, with the parameters specified in the configuration file or in a specific GUI section, module-by-module. Next the parameter monitors are created and assigned to the parameters of the independently modelled systems. Once all modules have been started and all initial events are in the Event Stack, the simulation starts. It continues running until the End Simulation Event stops the generation of new Events, such that eventually the Event Stack becomes empty and the simulation ends.

The gathered sample lists are statistically analysed and the result of this analysis is stored in the according result lists. Once that is done, obsolete sample lists become cleared. See subsection 3.3.3 for details. In the case that a sequence of simulation steps with changed parameters is specified, one-by-one the simulation modules become re-initialised with the changed parameters and all others become reset. New initial events are pushed into the Event Stack to start the execution of the next simulation step.

Every time a step is finished the visualisation is updated to show the progress. The complete simulation run ends when all steps have been performed. The GUI remains open and offers the user a button to jointly save all results. If the user tries to close the main window, or attempts to select a different simulation configuration file, the user is asked whether to store the current results or to discard them prior loading a different scenario.

3.3.3 Parameter Monitoring

Simulating interrelated energy flows offers a multitude of parameters that can be monitored during a simulation run. Not all are relevant for all studies, and every monitored parameter causes computation effort. To keep the effort reasonable, and to maximise the applicability of the simulation framework, the list of parameters to monitor shall be part of the customised configuration file.

Simulation Control creates a separate module for monitoring, the Simulation Monitor. This reduces dependencies and enhances the overall architecture by ensuring a clear separation of concerns. As a result, the system becomes easier to maintain and extend, since each module can be developed and updated independently without affecting others.

Simulation Monitor instantiates a dedicated monitor for each modelled system, such as feeder network, heating system, PV system, charging site, and assigned electric vehicle. Each monitor maintains its own set of parameters to track, enabling focused monitoring for the selected system parameters.

For each monitored parameter, it provides (a) Sample Records (Traces), which are created in each simulation step, and (b) a Result Record, where in each simulation step, one complete row is added. A panda may for example contains the Traces (time series) together with a Statistics Record calculated per step, as sketched in Figure 3.4.

To exclude the transient system behaviour at the start of a simulation step and at the end, the first Monitoring Event shall be scheduled some time after the simulation start. This shall be specified in the general simulation parameters section of the configuration file. At the end, the moment the simulation end flag is set, no more samples shall be recorded.

$t_i < t_{min}$	$t_{min} < t_i < t_{end}$	$t_i > t_{end}$
no samples	Monitoring Event records samples	no samples

At the end of every simulation step, the recorded Sample Trace is used

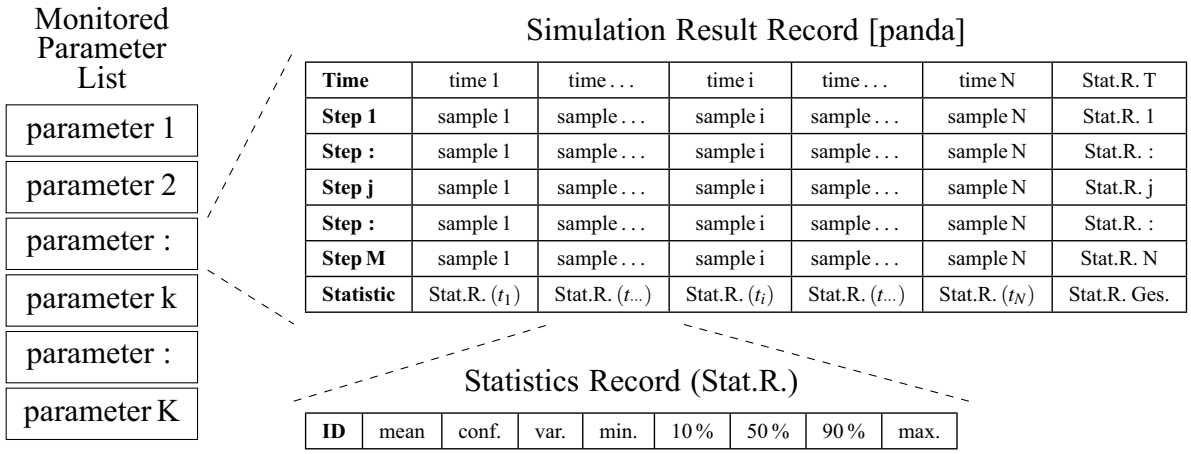


Figure 3.4 – Parameter List, Sample Records, Statistics Records, Result Record

to calculate the Statistics Record across the recorded trace, which is added to the end of the row just created in the according Result Record. Each Statistics Record includes the mean value and confidence interval, the variance or standard deviation, minimum and maximum value, as well as the 10 %, the 50 % (median), and the 90 % percentile, as sketched in the according inset in Figure 3.4.

When all simulation steps have been finished, the statistics per time step are calculated and added to the Result Record as final row. These are of particular relevance in case the configuration is not change from step to step, meaning in the case that a sequence of randomly generated sample traces are required to find time dependent statistics.

In the case that the GUI is used, another list of parameters specifies which parameters shall be visualised in the course of a simulation run, as shown in Figure 3.5. The presented screenshot shows for example the results for the first site: Indoor temperature, heating power demand, PV production, and the supply voltage at the grid connection point. Five simulation steps are performed with different algorithms enabled step by step. Because this site is the closest to the transformer, the algorithms are not triggered at this location. Neither is the PV curtailed nor the heating reduced. The variation in the upper heating related figures is due to minor fluctuations and the auto-scaling applied by the graphical tool creating the 3D-graphs. However, the activation at other sites is visible in the varying grid voltage at the connection point.

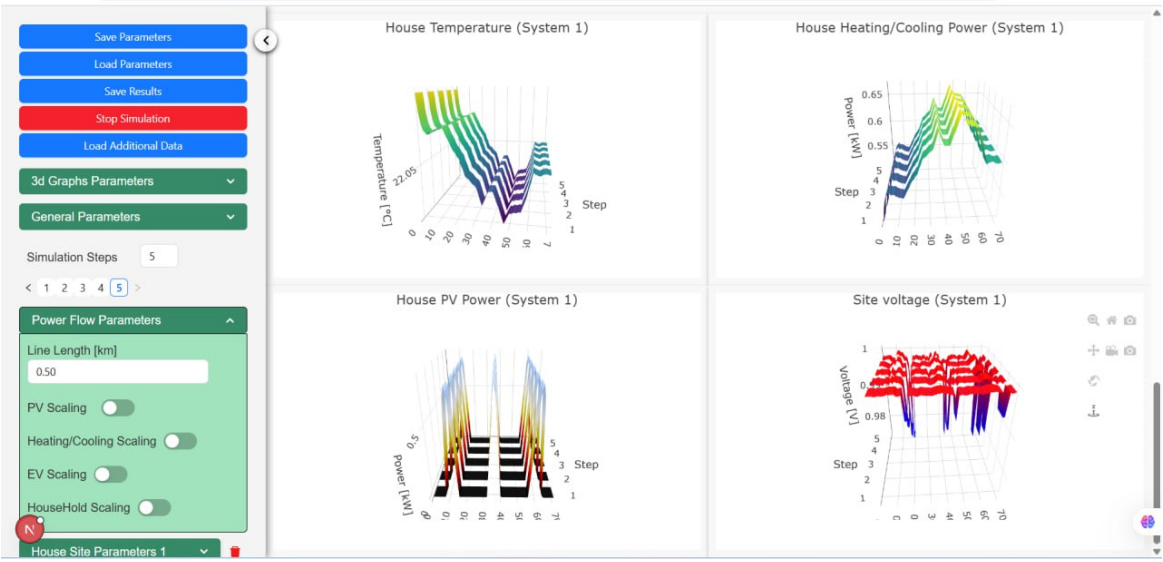


Figure 3.5 – Progress monitoring by parameter visualisation in GUI

The initial visualisation in the GUI is refreshed whenever a simulation step ended. Thereby, the simulation progress is presented to the operator step-by-step. Misconfiguration and other problems can be detected early without the need to wait until all simulation steps are finished. The possibilities for premature termination are described next.

3.4 Interfaces to linked simulation tools

The exploration in section 3.3 provides insights into managing diverse functionalities of the simulation framework itself. In this section, the integration of existing systems modelling tools is addressed. The focus is on the interfaces provided by the integrated tools and how these are used to perform the simulation.

3.4.1 Electric power flow simulation

The *pandapower* simulator is a toolbox, written in Python, provided open source and free to use, available at www.pandapower.org. It combines the data

analysis library *pandas* and the power flow solver *PYPOWER* to analyse electricity grids. It uses *pandas*, a powerful Python library for relational and labelled data, to configure and execute calculation and simulation tasks.

A *panda* Data Frame is a data format that allows to store data in a two dimensional structure, comparable to a matrix, but with freely named (indexed) rows and columns, which can hold any kind of data, for example a mixture of values and strings. An example is shown in Figure 3.6. Note that in general indices

INDICES	*column 1*	*column 2*	. . .	*column n*
time	UTC 1	UTC 2	. . .	UTC n
red	string 1	string 2	. . .	string n
red	value 1	value 2	. . .	value n
. . .	. . .	. . .	. . .	. . .
row m	duple 1	duple 2	. . .	duple n

Figure 3.6 – Panda: data file structure

need not be unique. However, using the row index *red* to extract date from the panda example shown in Figure 3.6, returns a data frame containing both rows indexed with *red*, which in some applications can be a welcome feature. The same applies for columns.

To simplify initializing the pandapower module and organize communication with the python program, the Panda Power Wrapper is embedded in the Simulation Control module. It is responsible for starting the Python script and providing the necessary parameters, such as the number of customer sites, the length of the power lines, and specific site details, in particular the varying power demand. These parameters are passed to the Python script as command-line arguments when it is launched.

The part of the code shown in Figure 3.7 is responsible for building the electrical feeder network that will be simulated. The process begins by setting up the main substation, which is assumed to provide infinite power. This is the starting point of the feeder network. It then connects the substation with a transformer. For each customer site, a new connection point is added and a feeder line with

```python
def create_feeder_network(customers_number, line_length, site_parameters):
    net = pp.create_empty_network()

    main_bus = pp.create_bus(net, vn_kv=20.0, name="Main Substation")
    feeder_start = pp.create_bus(net, vn_kv=0.4, name="Feeder Start")

    pp.create_transformer(net, main_bus, feeder_start, std_type="0.4 MVA 20/0.4 kV")
    pp.create_ext_grid(net, main_bus)

    last_bus = feeder_start
    customer_buses = []
    for i in range(customers_number):
        new_bus = pp.create_bus(net, vn_kv=0.4, name=f"Customer {i+1}")
        customer_buses.append(new_bus)

        site_param = json.loads(site_parameters)[i]
        pp.create_line(net, last_bus, new_bus, length_km=line_length, std_type="NAYY 4x50 SE",
                       name=f"Line to Customer {i+1}")

        access_limit_mw = site_param.get("accessLimitKW", 100.0) * 0.001  # Convert kW to MW
        pp.create_load(net, new_bus, p_mw=0.001, q_mvar=0.0005, max_p_mw=access_limit_mw,
                       name=f"Load {i+1}")
        last_bus = new_bus
    return net, customer_buses
```

Figure 3.7 – Panda: creating of feeder network

specified length, which connects the new connection point to the previous. It also sets the power limits and other characteristics for each site based on the provided parameters. When this feeder network set-up process is finished, it returns a complete network model and a list of all customer sites, ready for simulation.

The next part of the code, shown in Figure 3.8 is responsible for updating grid power flow parameters. When the Java application wants to update the power demand of a customer site, it sends a message containing the new demand values. The Python function receives this message, extracts the list of power values, and updates the simulation model.After updating the model, the script runs a calculation to simulate how these changes affect the network, particularly the voltage at each customer site. It then collects the voltage results for all sites and sends this information back to the Panda Power Wrapper that forwards these values to the Simulation Control module.

Based on the parameters received from *pandapower*, Events relevant for maintaining safe and smart power flows may be triggered. These most likely affect also other systems. For example, if the available power for EV charging shall be changed, an Event to adjust the available site power parameter of the affected

```python
def process_command(cmd):
    site_powers = cmd.get("site_powers", [])
    print(f"Received site powers: {site_powers}")

    for i, bus_id in enumerate(customer_buses):
        load_idx = net.load[net.load['bus'] == bus_id].index[0]
        net.load.at[load_idx, 'p_mw'] = 0.001 * site_powers[i]
        print(f"Updated load for {bus_id}: {net.load.at[load_idx, 'p_mw']} MW")

    pp.runpp(net)

    voltages = []
    for bus in customer_buses:
        voltages.append(net.res_bus.vm_pu[bus])

    return {
        'voltages': voltages,
    }
```

Figure 3.8 – Panda: update feeder line parameters

smartDCsim instance is pushed to the Event Stack. To be executed instantly, the execution time of this Event is set to the current time, being the execution time of the Event that triggered the re-calculation of power flows.

Power flows and site connection point voltages from the most recent *pandapower* execution are also stored with the sites, enabling smart power flow management algorithms and the monitoring system to use and track *pandapower* parameters in the same manner as those from other systems for power flow management and statistical evaluation respectively.

3.4.2 EV charging site simulation

The smartDCsim tool is implemented as a Java program, like Simulation Control, which eliminates the need for an external API to facilitate their interaction. However, they interact with each other by a predefined interface.

Simulation Control maintains a configuration list for each site and provides this configuration during the site's creation and throughout each simulation run,

executed as shown in Figure 3.9. It also supplies a reference to itself, allowing the charging site to interact with the event simulation.

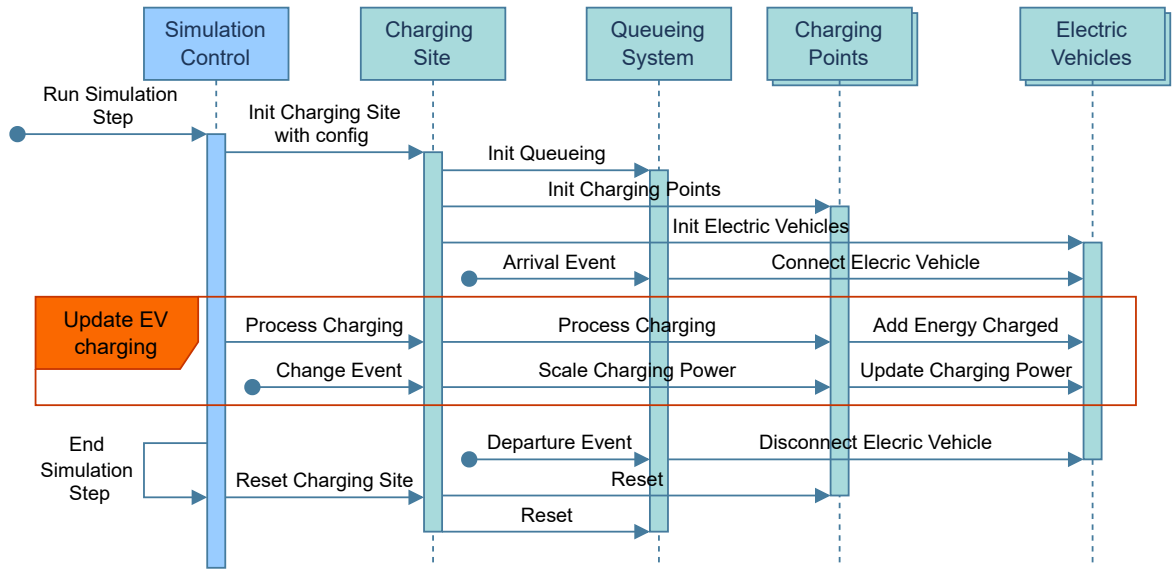


Figure 3.9 – UML of EV Charging at a Charging Site

Each instance of a site has its own EV Queueing System, which provides the logic for handling dynamic queueing and resource allocation to manage the local EV charging. The Queueing System uses reference to Simulation Control to schedule and process events, ensuring that all queueing and charging operations are integrated into the simulation's event-driven workflow.

The main events that can be scheduled are: arrivals, departures, queueing and blocking, as well as available power change. These scheduled events are added to the simulation's event stack or processed immediately, ensuring that all changes of client and server states are reflected in the global simulation timeline. This design allows for centralized management of simulation events and consistent state updates across all charging sites. In addition exposes the Charging Site object its internal configuration and power metrics, allowing other components to query and manage the site's state, charging process, and monitoring integration.

One of the most important power metrics of the Charging Site is the total power consumption $P_{\text{site}}^{\text{EVs}}$. Simulation Control uses the Get Total Power method for calculating and returning the total power currently being drawn by all charging

points at the site. This metric is essentially required for the electricity power flow calculations. It is also used by the Simulation Monitor for statistical evaluation.

The Charging Site object provides the possibility to scale the available charging power up or down, within configured minimum and maximum limits. This is used to adjust power flows, for example in case the electricity power flow calculation, in particular the voltage at the site's feeder network connection, advises a charging power adjustment, which is than triggered by a Charging Power Change Event. Likewise, a Charging Power Change Event could be triggered by a dynamic electricity price or the current CO_2 footprint of electricity from the grid, depending on the smart power management algorithm that shall be evaluated.

The Charging Site also provides an interface for initiating the charging process of electric vehicles. Central to this functionality is the processCharging method within the ChargingPoint class, which governs the progression of charging for the vehicle currently connected to a given charging point. If a vehicle is present, the method first updates the vehicle's charged energy, incrementing the charged energy (state of charge – SoC) based on the elapsed simulation time and the current charging power. Subsequently, it recalculates and adjusts the charging power in accordance with site constraints, in particular the Available Charging Power, the vehicle status, and other dynamic parameters, for example local PV production. In the absence of a connected vehicle, the method remains inactive.

Finally, to support accurate, independent, and repeatable simulation runs, the Charging Site includes a comprehensive reset mechanism. The reset() method is responsible for restoring the state of the entire charging site to its initial configuration at the beginning of each simulation run within the same or a changed scenario. Specifically, this method resets all charging points, clearing any ongoing charging sessions or power allocations, and reinitializes all electric vehicles by removing accumulated charging data and queueing states. Additionally, it resets the queueing system, clearing all client and server states, queues, and counters, and restores the dynamic power limit to the maximum site power, as defined for the configuration to be analysed in the current simulation run.

3.4.3 Heating system simulation

Integrating *Modelica* is a little more tricky. To calculate the behaviour of a heating system in case a parameter is dynamically changed, either all changes need to be known in advance and configured in a table, or the calculation needs to be redone with changed configuration.

The former is not applicable in general because some changes are dynamically triggered by other modelled systems (smart algorithms) and thus, not known in advance. However, weather data and user behaviour, for example varying heat demand across different days, can be handled using tables because these are part of the static simulation configuration, i.e., known when the model is initialised.

A particular challenge are dynamic latencies introduced by heat capacities and delayed heating responses. The current state of the modelled heating system at the time of the change needs to be used as initial state for recalculating.

The default prediction horizon shall be 10 days. If no change event occurs until the prediction horizon is reached, a continuation event triggers the calculation of the next 10 days. How the different modules work together to achieve a smooth simulation considering dynamically triggered parameter changes of heating and cooling, as well as PV production (which is also weather dependent), is shown in Figure 3.10 and explained next.

In the beginning, Simulation Control initializes Heating Control using the parameters of the Modelica simulation set-up configuration. Each Heating Control instance receives the parameters of the modelled building (house) and the PV configuration. It sets up the Modelica simulation wrapper, and initializes all internal state variables for temperature, PV scaling, and simulation results storage.

Heating Control uses the Modelica Python Wrapper for the communication between the Java simulation framework and the external Modelica-based Python simulation. It encapsulates the logic for configuring, launching, and communicating with the Python process that executes the Modelica simulation runs, passing relevant house parameter changes and execution options.

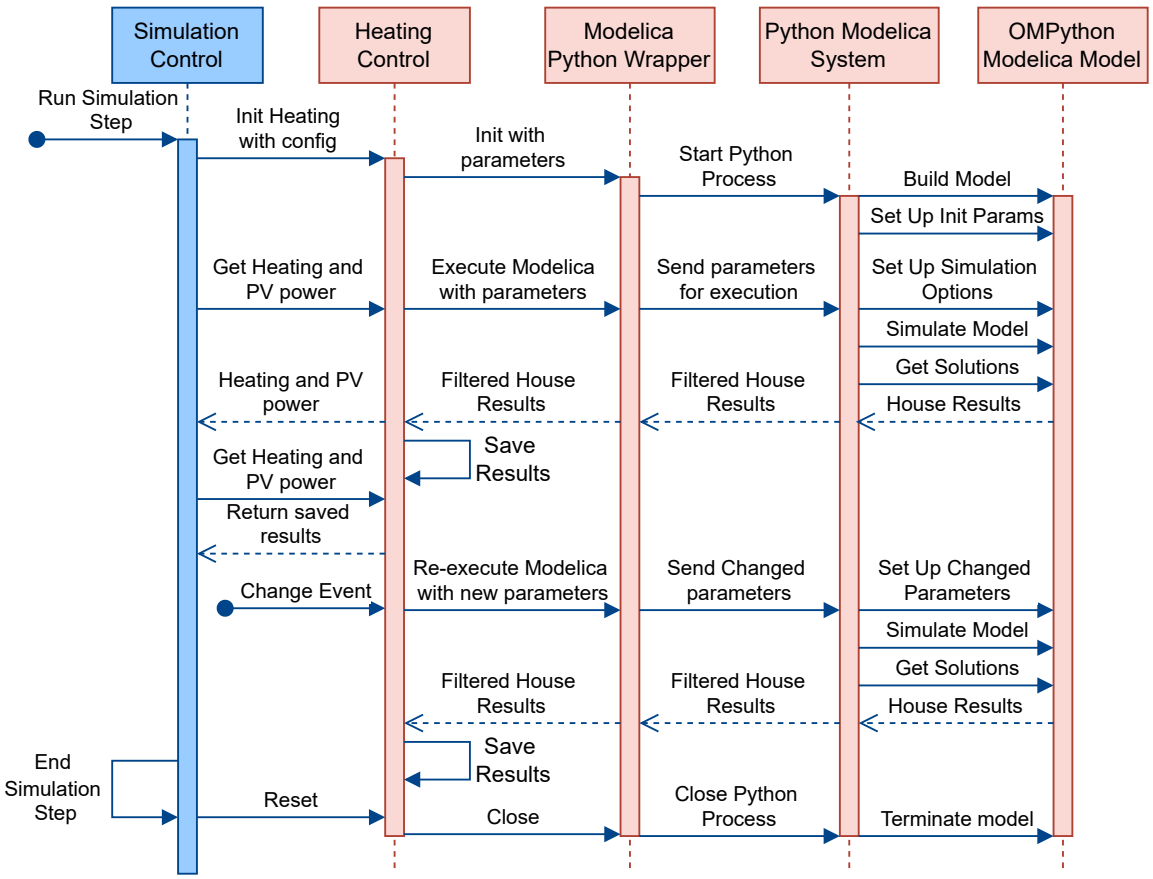


Figure 3.10 – UML for integrated Heating and Cooling system simulation per Site

The purpose of Python Modelica System is to provide an interface for simulating a Modelica-based system model using OMPython and OpenModelica. It loads and builds a Modelica model, sets simulation parameters such as heating/cooling temperatures and power, and processes simulation commands received in JSON format. When requested, it runs the simulation, extracts relevant results (i.e., temperatures and power consumption), and returns them in JSON format.

The *ModelicaSystem* library, provided by OMPython, serves as a Python interface for interacting with Modelica models and the OpenModelica simulation environment. It enables programmatic loading, parameterisation, compilation, and simulation of Modelica models directly from Python scripts. This facilitates automated workflows, integration with data analysis tools, and flexible scenario management, as needed for example in scientific research. ModelicaSystem is particularly valuable for coupling advanced physical system modelling with Python-based control, optimisation, and post-processing, supporting reproducible

and extensible simulation studies.

Heating Control stores the latest results received from Modelica. Simulation Control can request heating and PV power values for a specific simulation time. If the requested time falls within the stored time range, Heating Control returns the corresponding results. Is the requested time beyond the currently stored time range, Heating Control runs the simulation for the next 10 days until it can provide the requested simulation results.

Simulation Control uses the received heating and PV power, together with site power, for power flow processing, i.e., the implementation of smart power flow management algorithms. Thereto, *PowerFlow* receives local voltage values, and if these values exceed boundaries specified by a smart algorithm, a resultant change event is triggered for the heating, cooling, or PV system.

When *HeatingControl* receives a change event, it re-executes Modelica with accordingly adjusted parameters. The results automatically overwrite the overlapping part from previous Modelica runs. The next time PowerFlow is executed, it use the heating and PV power parameters updated according to the processed change event. This may occur recursively in case of successive changes.

When a simulation step is completed, the recorded results for all parameters reflect how the system performed and how the power flows changed according to both, the changes configured a priory, e.g., weather data, and the dynamically triggered changes caused by the smart power flow management algorithms implemented and tested. Figure 3.11 shows exemplary results as they are provided by the graphical user interface if configured to be displayed.

Here, the feeder length is increased step-by-step from $0.1\,km$ to $0.5\,km$ and to $15\,km$, to see how the heating/cooling power demand responds to voltage band violations. For $0.1\,km$ feeder length in step 1 no violations occur. For $0.5\,km$ and $15\,km$ they do occur, and accordingly is the cooling power demand reduced in steps 2 and 3. The shapes are similar because the response is not scaled with the voltage deviation. Only the times at which the violation and the according response occurs shifts slightly.

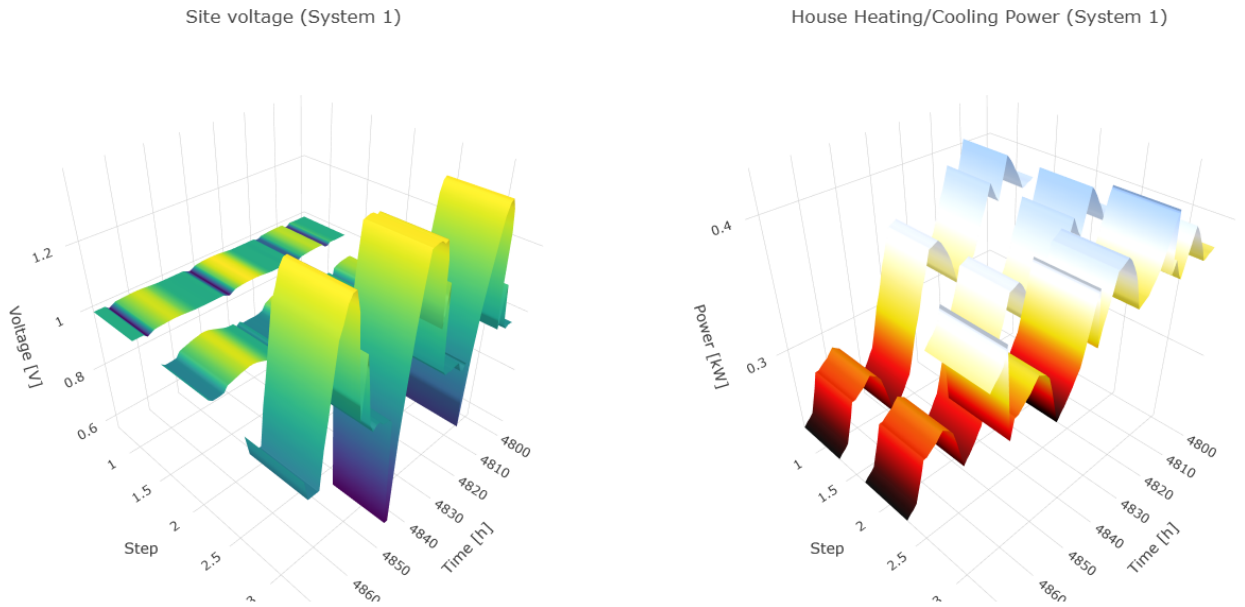


Figure 3.11 – Site voltage (left) and heating/cooling power demand (right) of an exemplary site simulated with OMPython and OpenModelica using weather data from Kiev 19th to 21st July 2009 as visualised in the simulation framework's GUI

3.5 Additional features of the simulation framework

While the core simulation functionality provides data samples only, the simulation framework also offers tools to configure or load scenarios and to comprehensively save, analyse and visualise gained simulation results.

3.5.1 Using last configuration as template, saving and loading scenarios

If a new simulation step is added, the configuration of the last specified step is used as template. The same applies for configuring new EV types. Wherever possible, the latest configured item is used as template for the specification of a new item of same type.

Once a simulation run has been configured step-by-step, the whole scenario can be saved in a JSON file using the *Save Parameters* button. With the *Load Parameters* button a saved configuration can be loaded from the saved JSON file

and adjusted using the GUI. An example configuration file for one site and three steps is presented in Appendix C, Listing 1.

3.5.2 Visualising of simulation results

The GUI offers the possibility to display traces and other data series graphically as soon as these are available to provide a quick feedback on the simulation progress. The 3D-visualisations can be rotated and zoomed as wished. Additionally, every graph can be individually saved in PNG and SVG format. Both these features are provided by the *plotly.js* library used to display the results in the GUI. Getting them to work properly can be challenging.

The parameters displayed step-by-step are specified in the scenario configuration. After the completion of all steps, the GUI remains open allowing the user to choose other data series to be displayed. Other parameters from the different simulation modules can be selected and with the *Load Additional Data* button these become displayed instead of the currently displayed graphs.

3.5.3 Premature termination of a simulation run

The application provides two options to interrupt an ongoing simulation run. Selecting the ’Stop Simulation’ button and confirming the action, as shown in Figure 3.12, ceases all ongoing processes. Alternatively, the ongoing simulation run is stopped in case the ’Start Simulation’ button is selected and the re-start of the simulation with the presently set parameters is confirmed.

In either case, the Simulation Control module stops the simulation run by cleaning the Event Stack. All simulation modules are terminated one-by-one, as always at the end of every simulation step when the Event Stack becomes idle.

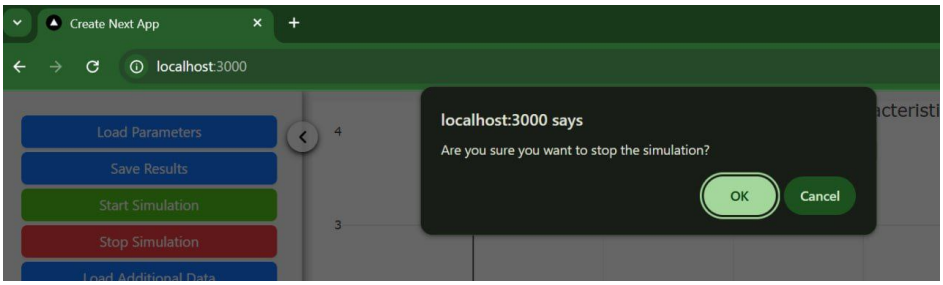


Figure 3.12 – Confirmation is required before a running simulation is stopped

3.6 Conclusion of Section 3

The integration of existing software tools was challenging. In particular, each tool follows it's own paradigm and assumes to cover all needs. They support integration of extensions, but not so much being integrated themselves. Interfaces (APIs) needed to be defined and implemented to access the other tools. Messages need to be translated (wrapped) to fit the specific information and operations structures and processes of the different tools.

The linking via events proved viable. In case of OpenModelica a simulated time period is redone on demand starting from the change time. Piece-by-piece the complete period is covered. Pandapower needs to be executed frequently to update the permanently changing power flows, which determine the voltages that control the smart algorithms evaluated. A huge variety of simulation studies can now be performed with the developed simulation framework. Comprehensive examples and possible extensions are presented and discussed in Chapter 4.

4 EXEMPLARY MULTI-SECTOR SCENARIO EVALUATION

The achieved simulation framework enables a sheer endless variety of studies, depending on system configurations and parameters chosen. The core feature is that all sites are linked by the voltage on the feeder network. If one site draws a lot of power, for example to charge EVs, than the voltage at all sites is reduced. Vice versa causes power insertion, here from PV systems, a voltage rise. These changes propagates without any delay into every branch (bus) of the feeder network. The effect decreases with distance, but never vanishes completely.

In Section 4.1, we first presented a study which shows that the voltage at premises' grid connection points can be used as input to smart algorithms that mitigate voltage variation across the feeder network. This study was performed with a preliminary version of the integrated pandapower module and is published at the *17th International Conference on Applied Energy* (ICAE2025), December 8th to 12th, 2025, Bangkok, Thailand [17].

In section 4.2, we apply the completed feature set of the simulation framework and show its functionality based on an exemplary scenario composed of three different sites. Two residential homes and one office building. The example is chosen to highlight the potential benefit of smart EV charging, a topic often postulated but rarely approved. Publication of these and some more results is envisioned for late Spring 2026.

In [16] we introduce how a more sophisticated cooperation is achievable if energy flexibilities become offered and delivered on demand. It is shown that the basis for reliable flexibility offering is good prediction. This study has been published at the IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), October 20th to 23rd, 2025, Valletta, Malt, and will become available on IEEE Xplore soon. However, implementation and evaluation of prediction based energy management is beyond the scope here, the section is an outlook to possible continuation of the development, as postulated in Section 5.

4.1 Cross-sector Cooperation and Optimisation

Beneficial cooperation of energy sectors is possible if at times when ample energy is available in a sector, the other sectors have an energy demand that can be served, and vice versa in case of an energy shortage other sectors can support the sector in need.

For example, around noon many PV systems on the roofs of a private houses produce more power than is currently consumed within these houses. Instead of inserting the excess production into the public grid when many neighbours do the same, it is more beneficial to charge the private EV or slightly increase the room heating to postpone the heating demand in the evening. Thus, to benefit from ample energy supply, the energy demand in the other sectors needs to be controlable. Without demand flexibility, meaning active control, the utilisation of ample supply happens only by chance, which is no true cooperation in the literal sense of the term.

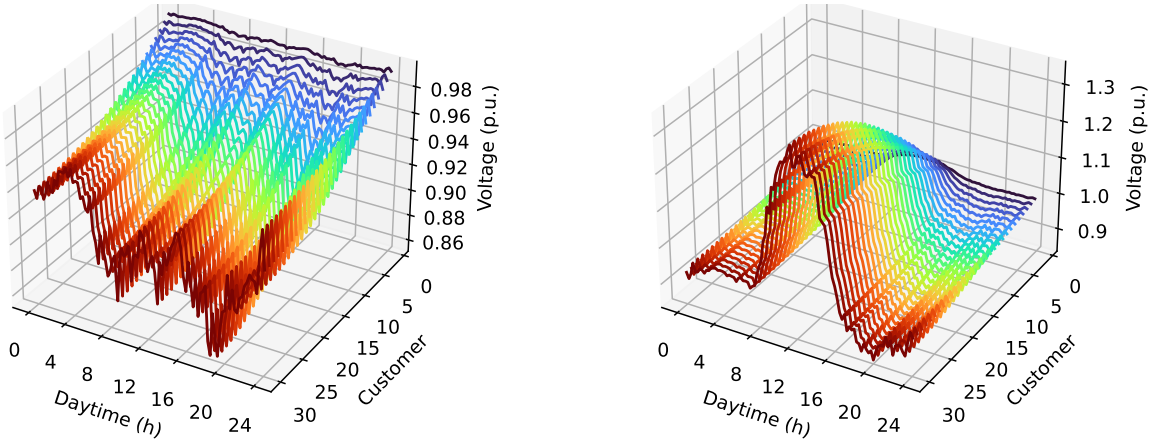
Vice versa, in the case of an energy shortage, other sectors can either temporarily raise their production or reduce their demand to support the sector in need. For example, the high electricity demand in the evening can either be mitigated by slightly discharging the battery of a vehicle to grid (V2G) capable EV, reducing the SoC by 5 to 10 kWh, which can be recharged after midnight at a cheaper energy price, or by postponing electric room heating by letting the temperature drop a little more than usual.

However, without knowledge about a surplus or need in one sector, the others cannot cooperate. In addition to flexible demands and controlable energy appliances, some means to signal needs across sector boundaries is required.

4.1.1 Modelling of controlable electricity assets and appliances

Using the same example feeder network as specified in Section 2.3, here one third of the customers is assumed to have a 5 kW PV system with its output

power scaled by the typical sine-square curve in between sun-rise and sun-set, and in addition varied by the same beta distribution to model cloudy conditions, as shown in Figure 2.7. The resultant line voltage behaviour without and with PV insertion is shown side-by-side in Figure 4.1



(a) without PV systems

(b) with PV systems

Figure 4.1 – Line voltage without and with distributed PV power insertion

A voltage rise and negative power flows toward the substation can be observed at times when along the feeder network more power is inserted by the customers' PV systems than is locally consumed. Observing varying conditions helps in identifying critical supply voltage violations, network bottlenecks in case of more complex topologies, and general performance issues, in particular energy loss due to the inner resistance of the used wire type.

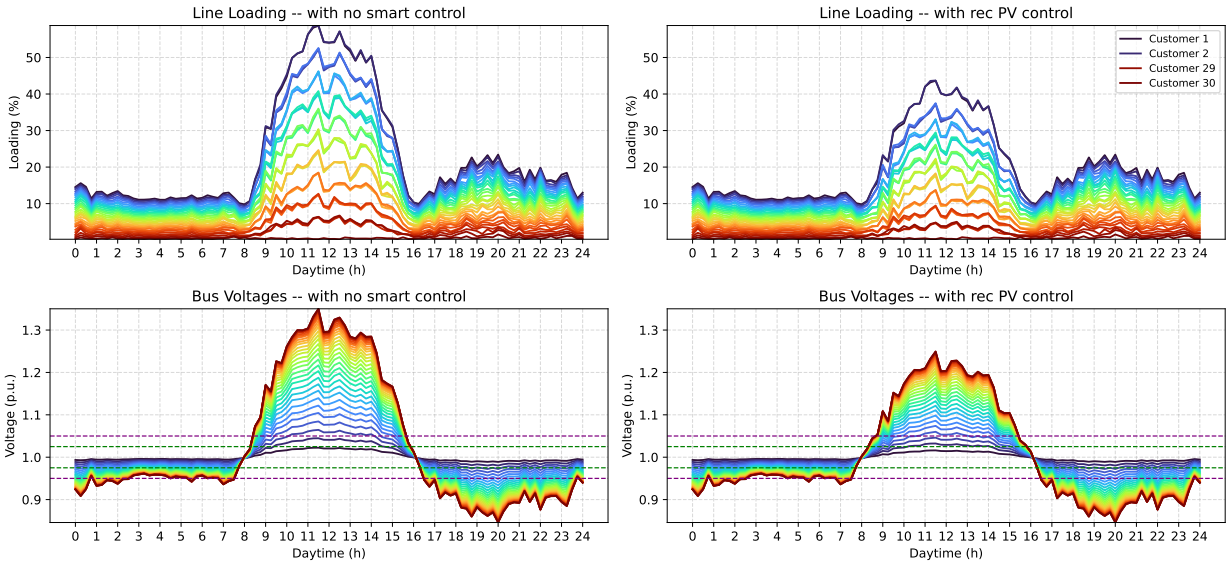
The core intent of a simulation tool is however, to validate counter measures. Smart PV inverters curtail the output power in case the voltage on the grid is too high. If in addition the grid operator provides a need indication, for example a kind of traffic light signal, such responses can be coordinated, yielding a fair curtailments distribution independent of the location along the feeder line.

A smart response to grid needs could be automated curtailment to for example 70 % of the peak output power of the PV inverter before the over-voltage becomes critical, as sketched in Algorithm 1. Should the voltage exceed the safety boundary, the PV inverter disconnects obeying regulatory grid codes. A preventive curtailment is therefore beneficial for both, the grid and the customer.

Algorithm 1 Simple voltage stabilising PV curtailment algorithm

Require: $U_{max} > U_0$ ▷ over voltage boundary > default voltage
Require: $U_{min} < U_{max}$ ▷ power output curtailment hysteresis
  $r \leftarrow 0.7$ ▷ set curtailment ratio
  for $j = 1,\ 2,\ \ldots,$ number of customers **do**
    if $U_j(t_i) - U_{max} \geq 0$ **then**
      $P_{max}^{PV_j} \leftarrow P_{max}^{PV_j} \times r$ ▷ curtail peak power output
    else if $U_{min} - U_j(t_i) \geq 0$ **then**
      $P_{max}^{PV_j} \leftarrow P_{max}^{PV_j} \times \frac{1}{r}$ ▷ revoke power output curtailment
    end if
  end for

To simulate such a smart algorithm, it shall be executed instantly whenever a power changing event causes the voltage level to exceed the set U_{min} or U_{max}. In the best case, this mitigates the voltage issue and no additional changes are required. If not, the loop proceeds until the issue is solved or no solution is achieved after all PV systems along the feeder have become curtailed. Figure 4.2 shows the effect of PV curtailment along the feeder line.



(a) without PV curtailment (b) with PV curtailment

Figure 4.2 – Line loads and bus voltages w/o PV curtailment (Algorithm 1)

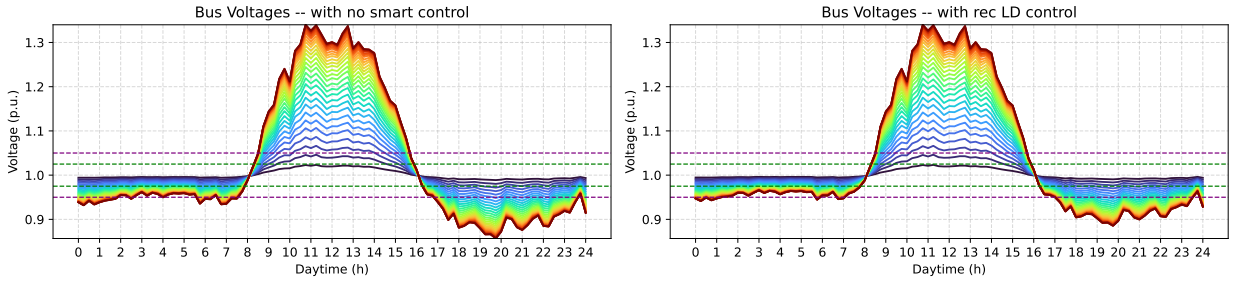
The effect of limiting the power inserted by local PV systems to 70 % of the rated peak power P_{max}^{PV} can be clearly seen. However, the effect is limited because

the curtailment is not possible recursively. In addition, customers close to the transformed do not see the need because at their position along the feeder line, the voltage is not in the region where Algorithm 1 causes a curtailment. Power insertion curtailment has evidently no effect on under-voltage situations.

The algorithm to determine when to cut the power demand is very similar to Algorithm 1. The difference is that Algorithm 2 intends to mitigate under-voltage by reducing the power drawn from the feeder line. Figure 4.3 shows the effect of load shedding (demand power cuts to 80 %) on the voltage along the feeder line.

Algorithm 2 Simple voltage stabilising demand cutting algorithm

Require: $U_{min} < U_0$ ▷ under voltage boundary < default voltage
Require: $U_{min} < U_{max}$ ▷ power cut hysteresis
  for $j = 1,\ 2,\ \ldots,$ number of customers **do**
    if $U_{min} - U_j(t_i) \geq 0$ **then**
      $\sigma_{off} = 1$ ▷ signal to cut power demand
    else if $U_j(t_i) - U_{max} \geq 0$ **then**
      $\sigma_{off} = 0$ ▷ signal to revoke demand cut
    end if
  end for

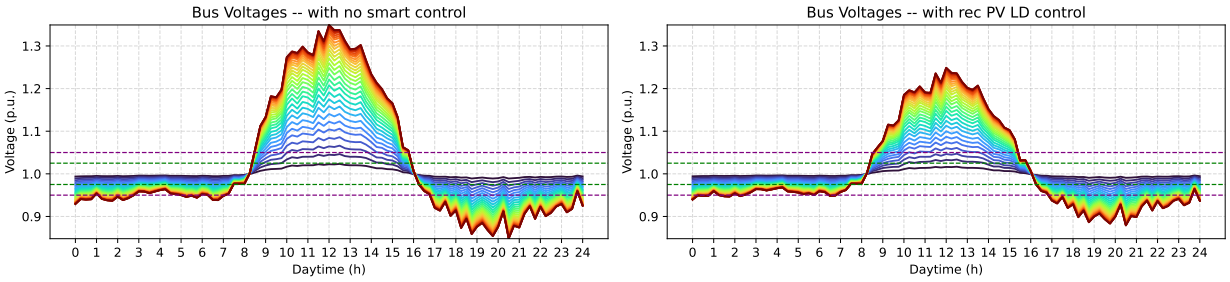


without load shedding with load shedding

Figure 4.3 – Customer bus voltages w/o load shedding (Algorithm 2)

The effect is visible, although small, and it has evidently no effect on the over-voltage introduced by neighbouring PV systems. Algorithm 1 and Algorithm 2 complement each other nicely by addressing the opposite issues of over- and under-voltage on the feeder line, as shown in Figure 4.4.

Typical appliances with on-off characteristic that can be controlled by Algorithm 2 are for example: hot water boilers, electric heating systems, refrigerators,



without PV curtailment and load shedding with PV curtailment and load shedding

Figure 4.4 – Customer bus voltages w/o Algorithm 1 and Algorithm 2

and private EV charging points in cased latter support no dynamic maximum power $P_{max}^{CP}(t_i)$ adjustment. However, their contribution to the power flow cannot be reduced at the same time. Not modelling these appliances individually, their contribution is assumed to be in average 20 % of the instantaneous load.

If dynamic adjustment or curtailment is possible, an algorithm very similar to Algorithm 1 can be implemented to determine power change events. Not being restricted to a static adjustment, it can be used recursively. Although applicable to many appliances, Algorithm 3 is here presented and used to evaluate smart EV charging. Alike Algorithm 2, it mitigates under-voltage by reducing the power drawn from the feeder line, and thus, compliments Algorithm 1. However, in case of over-voltage it increases the power drawn from the grid in order to mitigate the over voltage, thereby reducing the need to curtail the local PV production.

Algorithm 3 Simple voltage stabilising Smart Charging algorithm

Require: $U_{min} < U_0$ ▷ under voltage boundary < default voltage
Require: $U_{max} > U_0$ ▷ over voltage boundary > default voltage
  $r \leftarrow 0.5$ ▷ set adjustment ratio
  for $j = 1,\ 2,\ \ldots,$ number of customers **do**
    if $U_{min} - U_{site}(t) \geq 0$ **then**
      $P_{max}^{CS} \leftarrow P_{max}^{CS} \times r$ ▷ reduce power available for charging
    else if $U_{site}(t) - U_{max} \geq 0$ **then**
      $P_{max}^{CS} \leftarrow P_{max}^{CS} \times \frac{1}{r}$ ▷ increase power available for charging
    end if
  end for

Note that in Algorithm 3 the scheduling of the counter measure (*) triggers the re-calculation of the grid voltages from t_j till the last scheduled event. The $U_{site}(t_j)$ at the time t_j of the next event already considers all scheduled adjustments in consequence of previous iterations of the while-loop. Figure 4.5 shows the effect of load shedding (demand power cuts) on the voltage along the feeder line.

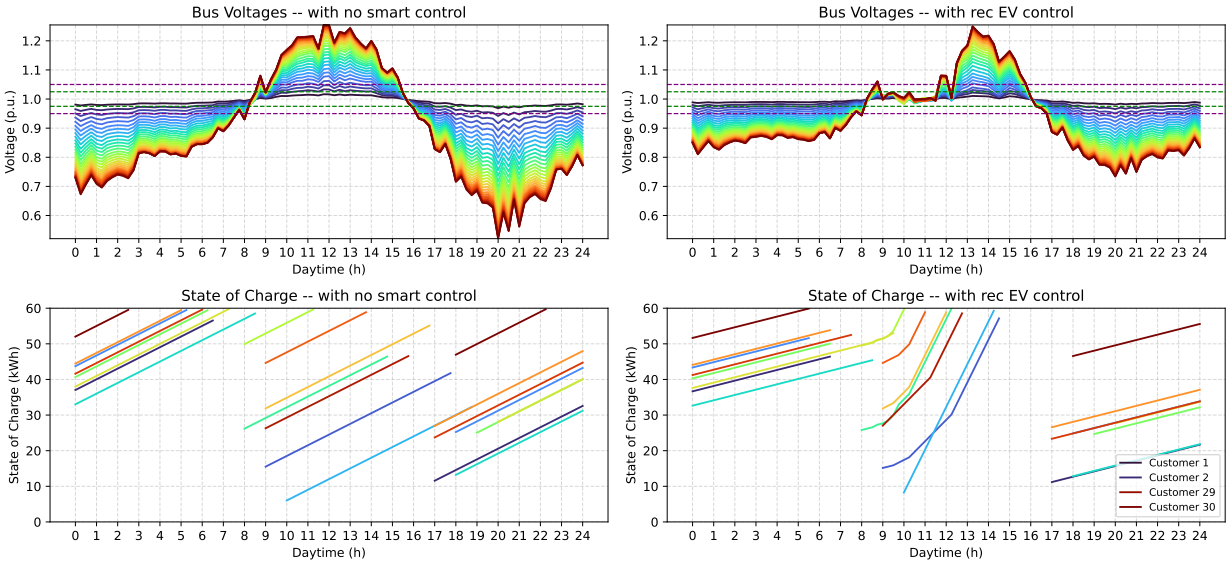


Figure 4.5 – Customer bus voltages w/o Smart Charging (Algorithm 3)

The lower graph in Figure 4.5 shows the EV charging. The rather low default $3\,kW$ charging power cause a linear rise of the battery state, as shown on the left side. When smart charging is used, the charging power becomes adjusted, which changes the steepness of the state-of-charge rise. Due to the extreme scenario chosen, all EV reduce the charging speed during the night. Every EV still gets at least $10\,kWh$, which is more than the average daily need of a commuter. During the day, the power inserted by the local PV systems is consumed locally by the EVs in the neighbourhood by increasing the charging power as much as possible. Over-voltage occurs only after all EVs in the vicinity finished charging, here after 12:15.

If all three algorithms are implemented and executed in parallel, a Smart Grid control is achieved in the way it was intended when the idea of Smart grids was born. Figure 4.6 shows the effect of all three algorithms together.

The three algorithms work nicely together. Smart Charging of EVs has

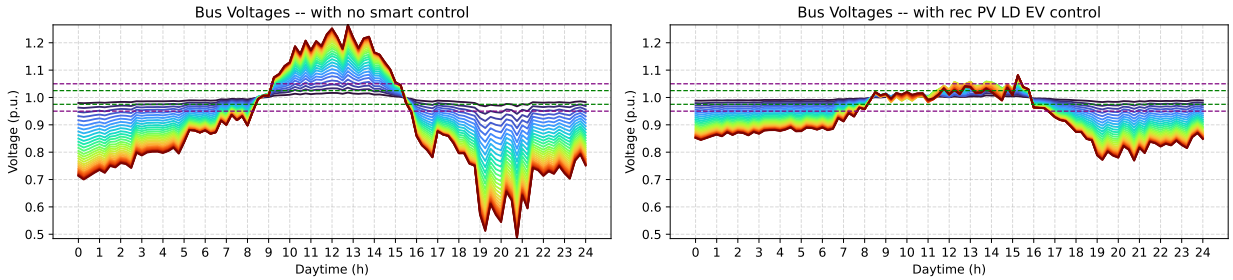


Figure 4.6 – Customer bus voltages with Algorithms 1, 2 and 3 acting together

definitely the highest impact. Only if no EVs are available, or the available have already become fully charged, PV curtailment and load shedding are required to counteract over and under voltage respectively. For the extreme scenario assumed here, Figure 4.6 shows that the existing wires can support higher PV insertion and EV charging load if smart grid technology is used to manage the demands of these customer appliances that more and more stress the traditional distribution grid.

The contributions of different customers along the feeder line differ. The contribution of Smart Charging outperforms both, PV curtailment and load shedding, as can be seen in the upper graph of Figure 4.7.

Due to showing absolute values, as it is common, the direction of the contribution is not clear. From the examples shown above, we know that the power flow changes during the night result from reduced charging speed, and the power flow changes during high PV production from increased charging speed. Environmentally and economically counterproductive PV curtailment is necessary only after the EVs in the neighbourhood have been fully charged. Evidently, the customer toward the far end of the feeder line benefit most in terms of voltage stability, as shown in the lower graph of Figure 4.7.

A too simplistic implementation of these algorithms turns out to be unfair because it discriminates customers connected further down the supply line, as shown in Figure 4.7. To balance that, the voltage limits U_{min} and U_{max} shall by specified more tight if the customer is connected less far down the supply line. In any case, it is wise to iterate from the transformer toward the far end, such that violations are solved as close to the transformer as possible because that minimises the individual adjustment required per customer.

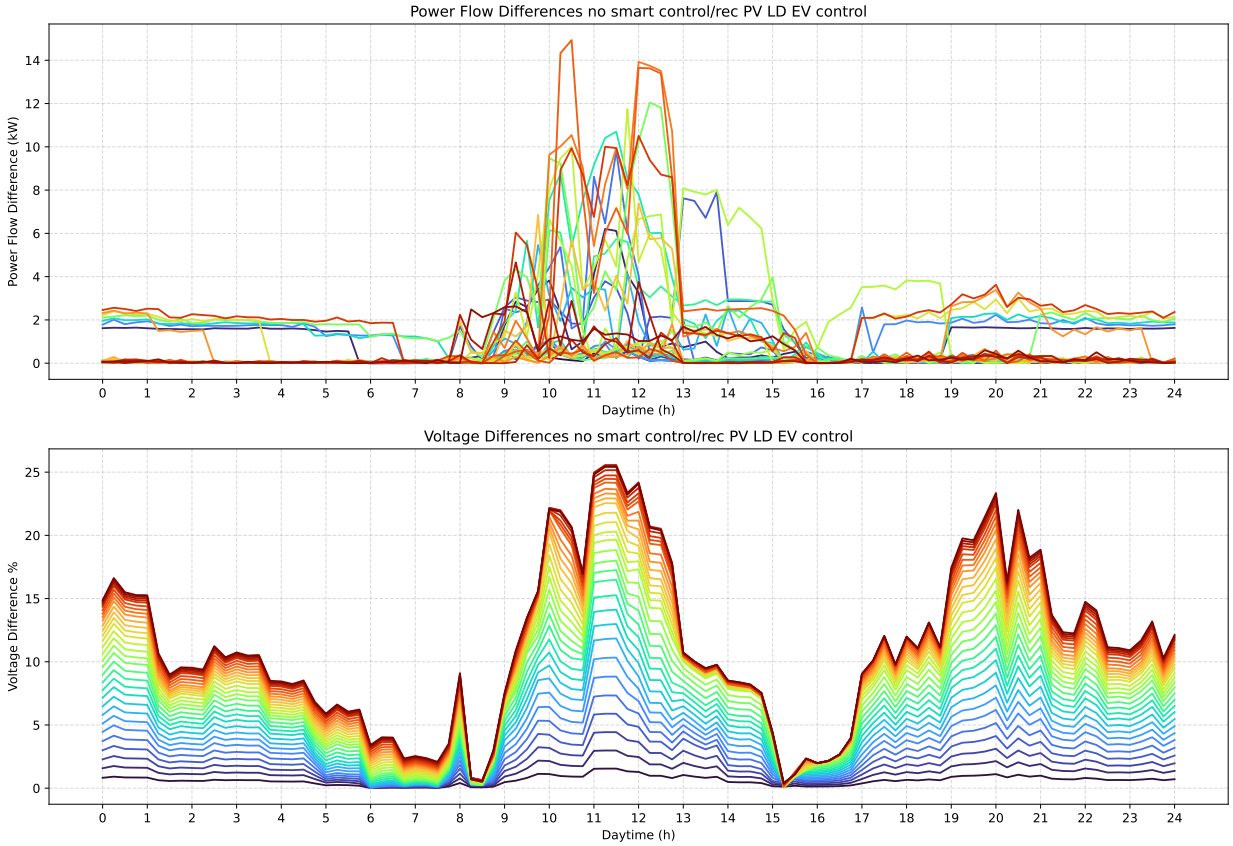


Figure 4.7 – Distribution of the customers' contributions and benefits to/from demand side grid stabilising along the feeder line

4.2 Simulating the interaction of diverse systems located at adjacent sites

The complete simulation framework can be used to do many different studies. Different ways to automatically adjust power demands to the current state of the feeder network can be evaluated depending on the configured scenario and chosen parameters. The here presented examples are focused on testing the functionality of the tool and not on the results of simulation runs as in Section 4.1.1.

4.2.1 Indirect power demand management for heating and cooling

The algorithms implemented for PV curtailment and EV charging power scaling, Algorithm 1 and Algorithm 3, remain as introduced in Section 4.1.1. Load

shedding, Algorithm 2, remains also available unchanged, but adjusting the demand of the heating system, as outline in Algorithm 4, is preferred.

Algorithm 4 Voltage stabilising heating and cooling curtailment algorithm

Require: $U_{min} < U_0$ ▷ under voltage boundary < default voltage
Require: $U_{min} < U_{max}$ ▷ heating reduction hysteresis
$T_{\text{heat}}, T_{\text{cool}}$ ▷ heating and cooling target temperature
$x\,^{\circ}\text{C}, y\,^{\circ}\text{C}$ ▷ allowed heating and cooling temperature deviation
for $j = 1,\ 2,\ \ldots,$ number of customers **do**
  if $U_{min} - U_j(t_i) \geq 0$ **then**
    $\delta_{on} = 1$ ▷ adjust target temperatures to delay possible power demand
    $T_{\text{heat}} \mathrel{-}= x\,^{\circ}\text{C}$
    $T_{\text{cool}} \mathrel{+}= y\,^{\circ}\text{C}$
  else if $U_j(t_i) - U_{max} \geq 0$ **then**
    $\delta_{on} = 0$ ▷ revoke the target temperatures adjustment → use power now
    $T_{\text{heat}} \mathrel{+}= x\,^{\circ}\text{C}$
    $T_{\text{cool}} \mathrel{-}= y\,^{\circ}\text{C}$
  end if
end for

Algorithm 4 does not affect the power demand directly. Instead, the target temperatures for the HVAC system become adjusted on-demand. A performed change can cause an instant or a delayed response, depending on the current room temperature and the inertia of the heating and cooling system modelled. Correct modelling of all the non-linearities of heating and cooling systems is provided by the Modelica model presented in Appendix C Listing 2. This is a clear advantage over linear modelling, although it complicates the model configuration.

4.2.2 Three buildings connected along the same feeder line

For the example here, where we want to approve correct inter-action of the systems modelled by different tools and linked by change events, very simple heating and cooling systems are used. To test the coordination of electric power flows, these are sufficient and yield results that can be backtracked somewhat

easier than with complex systems including diverse constraints, hydraulic inertia, and control latencies.

The base load of each building is again derived from the standard load profile, scaled to the mean demand per day and randomised using a Beta distribution in the range $(0 \text{ to } 5)$, where the mean is the daytime dependent value from the standard load. Thereby, the variations are bounded and high loads occur less frequent than low loads, which nicely approximates reality and is shown in Figure 4.8 for a ten days period (as used for Modelica).

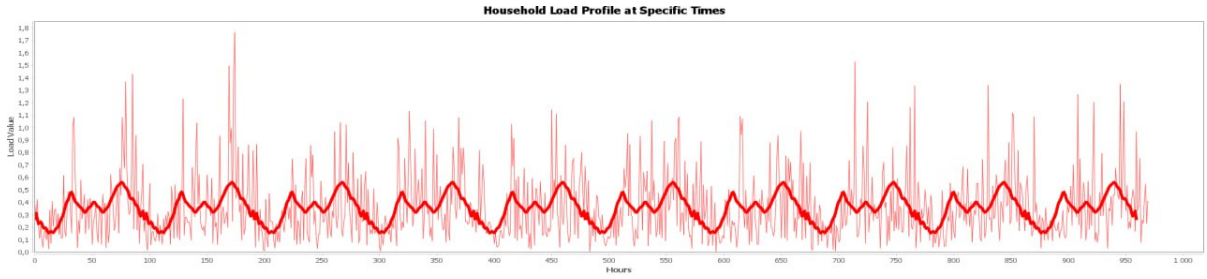


Figure 4.8 – Synthetic building baseload (narrow line) derived from standard load profile (bold line) by bounded random noise (Beta distributed)

The photovoltaic (PV) energy production, which in Section 4.1.1 was approximated by randomised $\sin^2$-function, is here modelled using Modelica where we use recorded weather data from 2009 for Kiev. The calculation is based on the PV-modules' peak DC output power $P_{\text{peak}}^{\text{PV}_{\text{modules}}}$ assuming that the installed PV-inverter can convert this DC PV-power into insertable AC PV-power $P_{\text{peak}}^{\text{PV}}$.

The Modelica model is basically the same as shown in Figure 2.10, here extended to provide the required modelling results needed for the integration in the framework. The PV modules are summarised in the symbol underneath the "Weather inputs" box. PV connects to outdoor temperature and solar irradiation parameters from the weather data and has no output into the heat modelling because the impact of building surface shading is assumed negligible. Small roof PV-systems start at $5\,kWp$, bigger have $\sim 20\,kWp$ (due to grid access limits), whereas on apartment and office buildings 50 to $100\,kWp$ are common. Industry complexes with several buildings and huge roof areas can go far beyond that, in particular if they anyhow need a strong grid access in the MW region at a higher

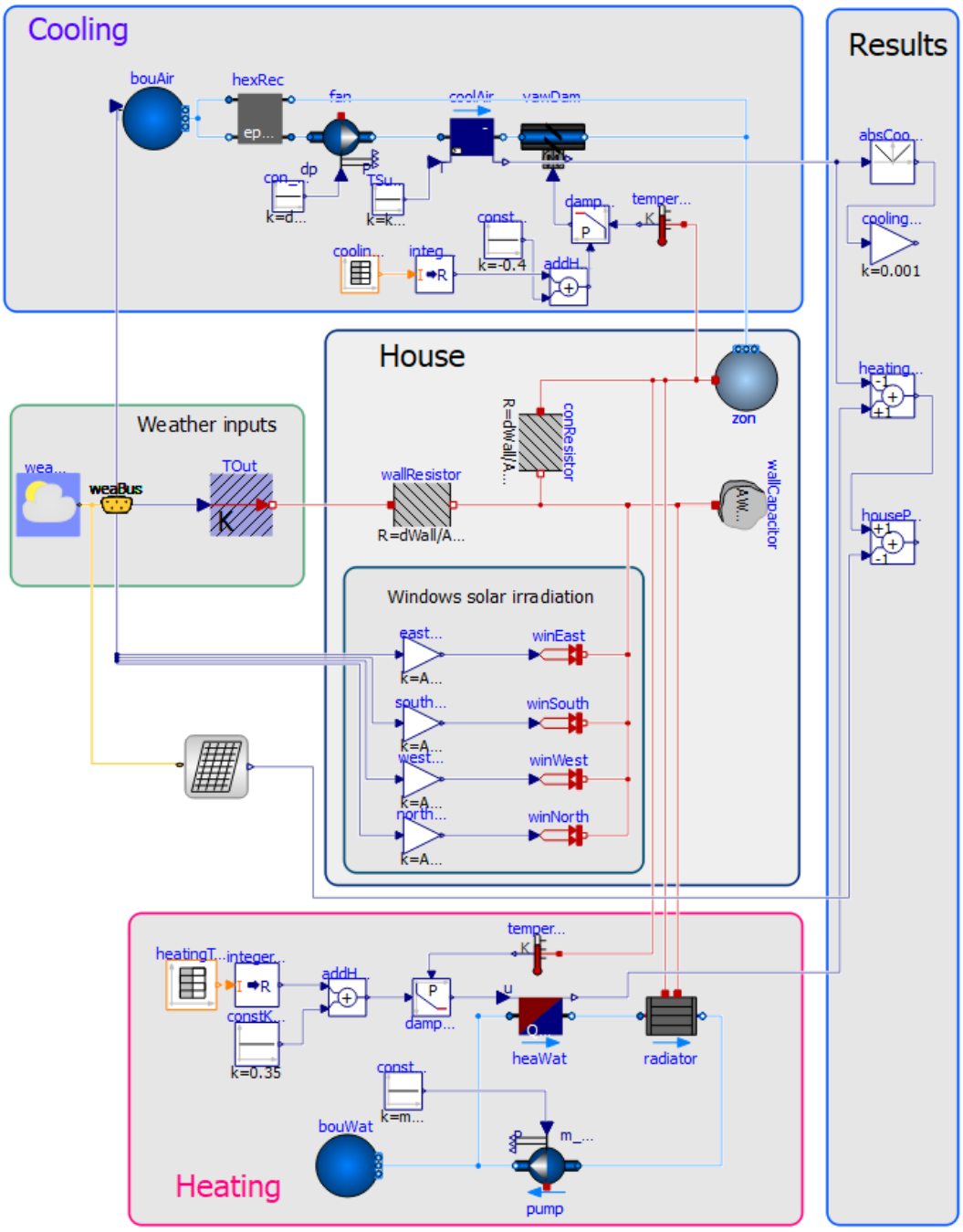


Figure 4.9 – Heating and cooling system model for a simplified house

grid level. Here we only consider lowest grid level access via feeder network connected to a local low voltage transformer.

The size of the heating and cooling system relates to the size and the building type. For a residential single household buildings we guess a peak heating demand of $\sim 4\,kW$ and for apartment and office buildings $\sim 50\,kW$.

The heating system is composed of a heat source with constant temperature

connected to a valve, a radiator and a pump that performs the circulation of the heat transporting medium, in general water. The control logic receives the heat signal from the room thermometer and activates the valve and the pump as needed. The cooling system uses the outside air as source with varying temperature according to the weather data. A fan and an air flow regulator are used to control the air flow into the room. The control logic includes a PI-controller to achieve smooth cooling. The house itself is modelled by a wall and an air volume with accordingly set thermic capacities and surface resistances. Four windows facing east, south, west, and north, model this heat ingress.

The EV charging is configured according to the assumed building type:

- Residential buildings:
  - at least one charging point per assigned EV ($n_{\mathrm{EV}} \leq n_{\mathrm{CP}}$),
  - no waiting space is needed ($n_{\mathrm{WS}} = 0$).
  - EVs arrive at around 18:00 with a standard deviation of ± 1 hour,
  - EVs park for 15 hours $\pm$ 1.5 hours.
- Office buildings:
  - assigned EVs may exceed the provided charging points,
  - waiting space $n_{\mathrm{WS}} \geq n_{\mathrm{EV}} - n_{\mathrm{CP}}$ needed to host all EVs.
  - EVs arrive at around 8:00 with a standard deviation of ± 1 hour,
  - EVs park for 8 hours $\pm$ 1.5 hours.

Charging points are assumed to support $22\,kW$ AC charging, although most modern EVs support $11\,kW$ only. Older EVs that do not support fast DC charging still use the 'fast' $22\,kW$ AC charging, some can charge AC up to $40\,kW$, but they fade out. The building's grid access limit may not allow multiples of $22\,kW$, therefore the maximum power available for EV charging $P_{\mathrm{max}}^{\mathrm{EVs}}(site)$ needs to be configured in the site parameters, as shown in Figure 3.3.

Most EVs support AC charging up to $11\,kW$ and are assumed to allow a charging power $P_{\mathrm{ch}}^{\mathrm{EV}}(t)$ reduction down to $1.25\,kW$. In between departure and arrival they are assumed to use energy and return with a charging state that is distributed as specified per EV type. Here, all EVs are assumed physically identical (only arrival and parking time distributions vary) to ease the backtracking of achieved simulation results.

4.2.3 Simulation results for the three adjacent buildings scenario

First, the result of applying the different algorithms needs to be confirmed. Figure 4.10 shows a series of days where the curtailment to 70 % of $P_{\text{max}}^{\text{PV}}$ (dashed line) is visible. The actual production loss is minimal because at non optimal weather conditions the PV system cannot produce so much power.

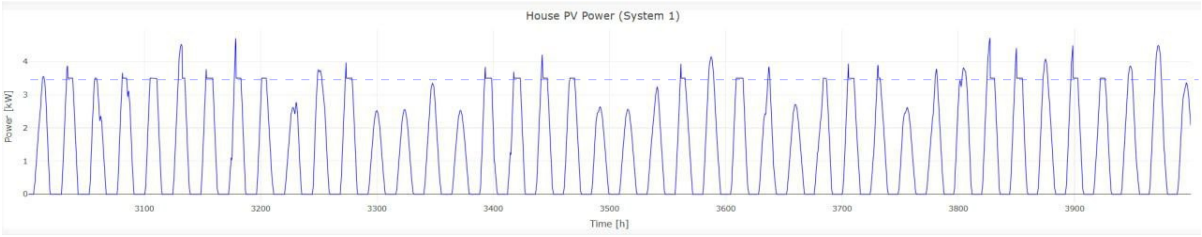


Figure 4.10 – PV output with correctly working PV curtailment algorithm

The EV charging power reduction becomes visible as slower rise of the State of Charge (SoC) when the charging power becomes reduced. This is shown in Figure 4.11 for two independent EVs. The timely correlation of the charging power reduction proves that the different sites where these EVs are charged are facing the same voltage drop on the feeder network and respond accordingly.

Figure 4.11 – Dynamic EV charging power adjustment across feeder network

The according traces are not so clear for the dynamic control of the heating and cooling systems' power demand. Because of the many non-linearities and the

delayed response in case the temperature is currently within the hysteresis range of the regular and the adjusted target temperature.

To see the correlation of the responses of the different systems, we show next exemplary 3D-graphs as they are provided by default in the GUI of the developed simulation framework. On the screen, these can be freely rotated and zoomed to see the interesting details more clearly. Printed on paper, this is obviously not possible and the important details are sometimes hard to spot.

The scenario are three houses connected to the feeder network in a row, each $500\,m$ from the next. The cable is maximally utilised to its specification to get serious voltage variation depending on the current load. The set parameters are summarised in Table 4.1.

Table 4.1 – Site parameterisation of the simulated example scenario

parameter		house 1	house 2	house 3
feeder network access power limit	$P_{\text{max}}^{\text{grid}}$	$20\,kW$	$10\,kW$	$100\,kW$
maximum heating/cooling power	$P_{\text{max}}^{\text{HVAC}}$	$5\,kW$	$4\,kW$	$50\,kW$
heating target temperature	$T_{\text{target}}^{\text{heating}}$	23 °C	23 °C	23 °C
cooling target temperature	$T_{\text{target}}^{\text{cooling}}$	25 °C	25 °C	25 °C
peak PV production capacity	$P_{\text{peak}}^{\text{PV}}$	$15\,kW$	$5\,kW$	$50\,kW$
site EV charging power limit	$P_{\text{max}}^{\text{EVs}}$	$20\,kW$	$10\,kW$	$80\,kW$
number of charging points ($11\,kW$)	n_{CP}	2	1	8
number of waiting spaces	n_{WS}	0	0	4
number of assigned EVs	n_{EV}	2	1	12

The first two houses are residential buildings, the third is an office building. Accordingly arrive the EVs in the evening or in the morning, as specified above. The arrival variation is assumed to be Normal distributed and the parking time is assumed to be Erlang distributed. Both these distributions are unbounded on at least one side, such that EVs may not arrive within the time window on the same day. In the case that the next arrival time occurs prior the parking time is expired, the EV is assumed to not having left, and the virtual re-arrival is discarded.

All electric vehicles (EVs) are assumed technically identical, for the simplicity of impact tracking, and modelled as specified in Table 4.2.

Table 4.2 – Example EV specification

battery size $E_{\max}^{\text{bat(EV)}}$	charging speed $P_{\max}^{\text{EV}}$	wished SoC $SoC_{\text{wished}}^{\text{EV}}$	charging demand		
			mean	range	distribution
$80\,kWh$	$11\,kW$	$80\,\%$	$40\,\%$	$\pm 30\,\%$	$Beta(a,b)$

Analysing the grid voltage at the feeder network connection point shown in Figure 4.12, we recognise the voltage rise during the day caused by the insertion of PV power. This is evidently less during winter and considerably higher in spring. In the evening, the EV charging causes a considerable voltage drop. That is not dependent on the season and appears less in spring due to the auto-scaling applied. The sharp drop underlines how important smart EV charging power adjustment is for the voltage stabilisation on the feeder network.

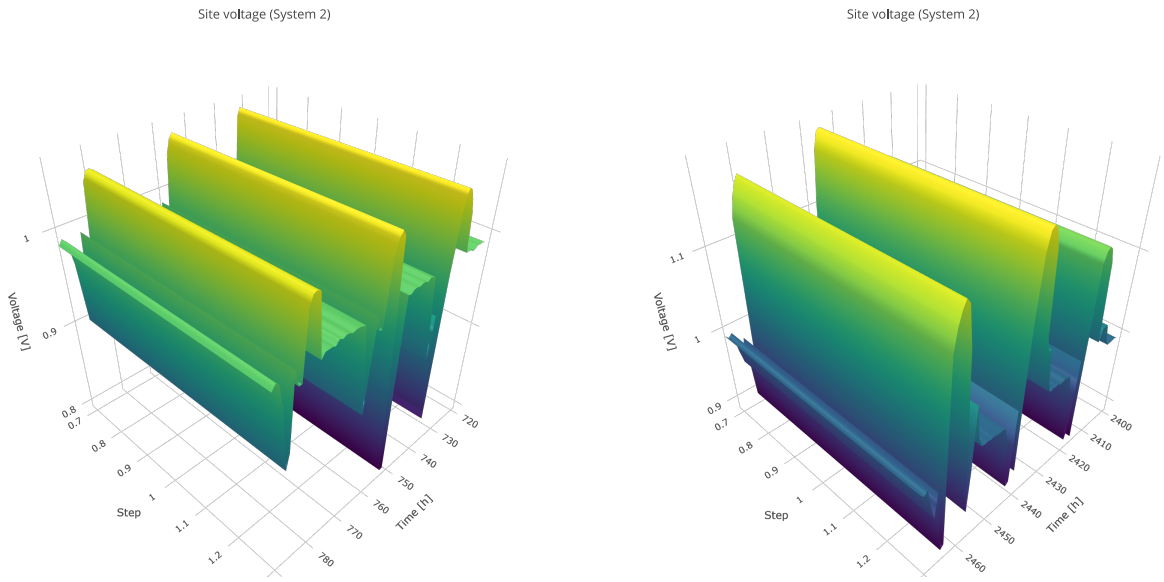


Figure 4.12 – Feeder voltage at house 2 across three days in winter, 30th Jan. to 1st Feb. (720 h to 792 h), and in spring, 10th Apr. to 12th Apr. (2400 h to 2472 h)

House 2 has no big PV or EV load by itself, but because it is connected to the same feeder line as the other houses, and in particular because the big office

building is connected at the end of the feeder line, it experiences a voltage swing that is much bigger than what it causes by itself.

The PV output curtailment at house 3 is shown in Figure 4.13. The actual power curtailment is small, but for the access capacity calculation, it makes a big difference because of the simultaneity factor of adjacent PV systems, which is close to 100 %.

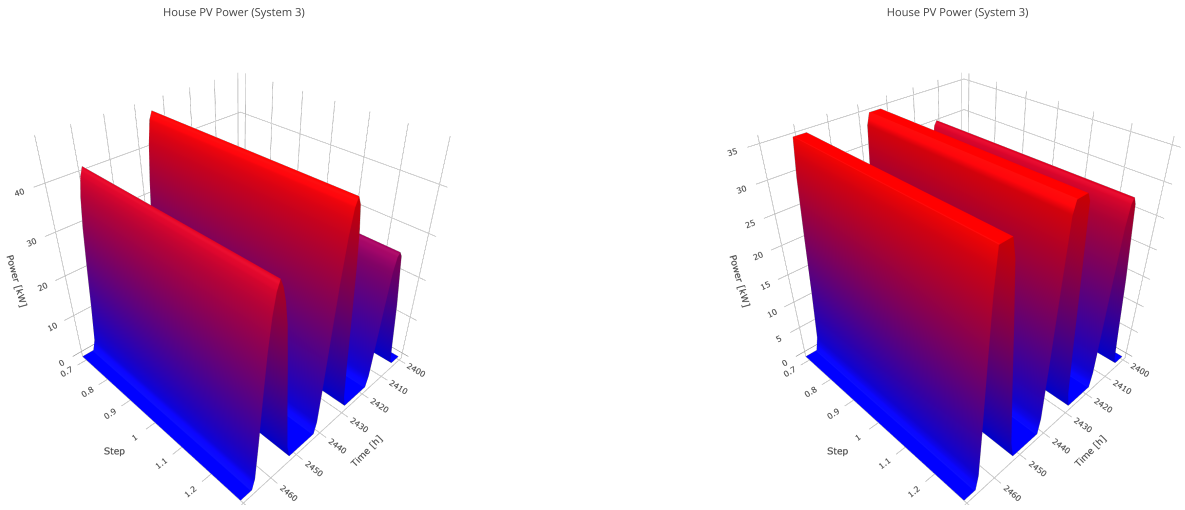


Figure 4.13 – PV output house 3 without (left) and with 70 % curtailment (right)

Here, the results for spring are shown because during winter the PV production never reaches the 70 % margin. Also on the first day (10th Apr.) it stays below and no curtailment happens. It needs to be noted, that due to auto-scaling the PV output on the right side appears bigger although it is actually smaller, which becomes evident if we look at the scaling of the vertical axis only.

As mentioned before, provides the EV charging power a load that is big and can be feasibly managed if smart EV charging stations are installed. Figure 4.14 shows the total EV charging power at house 2.

On the third day, we recognise a reduced charging power in the case that the scaling algorithm is activated. On the other days the lower voltage threshold that triggers the EV charging power reduction is just not reached. Why at house 2 the EV can charge with 20 kW yet needs to be verified.

As mentioned before, the effect of the heating power adjustment by altering the target temperature is difficult to present. However, Figure 4.15 shows the

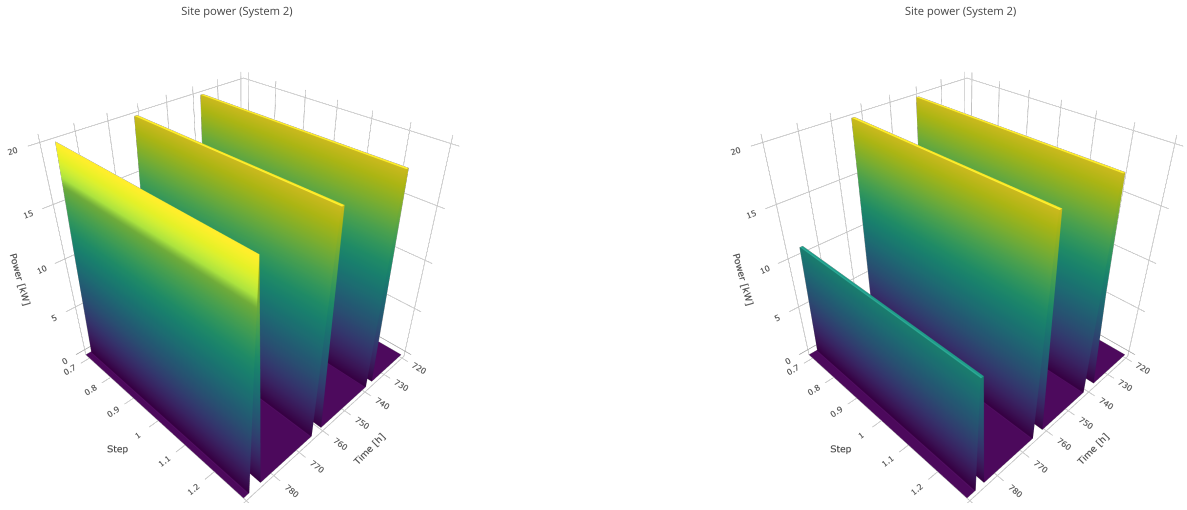


Figure 4.14 – EV charging power without (left) and with adjustment (right)

indoor temperature variation across three days in winter.

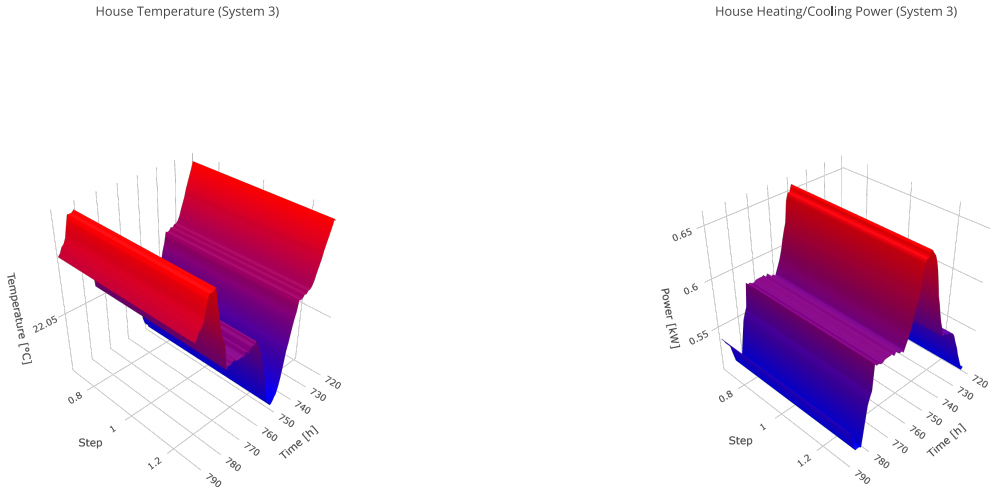


Figure 4.15 – Indoor temperature (left) and heating power (right) at house 3

On day two, we recognise a temperature drop that indicates an adjusted temperature target, here lowered by 2 °C. The power demand therefore does not rise during the second and third night, as it did in the first night. During the third day it even drops further because of the temperature rises without increased heating power, most probably due to sunshine and higher outdoor temperature in the weather data. Latter indicates a little how complex the relation of indoor temperature and heating system power demand is in reality where the weather plays a key role.

4.2.4 Interpreting complex simulation results

To see the correlation across systems, a parametrisation needs to be found that causes a clearly visible step-by-step variation caused by correlated systems interaction. Often the effects overlap and the reasoning becomes obscured. The stepping option comes in handy. It is used to vary a parameter such that at some point the envisaged correlation becomes recognisable. Simulation runs with diverse parameter stepping have been done. The best results yet found show the inter-correlation that results from increasing the feeder line length, from $0.1\,km$ to $10\,km$ in $0.1\,km$ steps. The results are presented in the appendix: Figure B.5 for winter times, Figure B.6 for spring, and Figure B.7 for summer.

Winter/Figure B.5: During winter, the PV output is rather small and the biggest loads are the EVs. The heating is modelled by a room that is so small that it's heat capacity is easily covered by a heating power no bigger than $1\,kW$. Hence, the heating power demands are the same at all houses. For the shortest line length (least voltage drop) the traces differ because no temperature adjustment to reduce the demand on the following days is triggered. For the second shortest line length, it is triggered for house 1 on the second day. In all other cases it is triggered on the first day and persists. The voltage swing increases linearly with the line length, as expressed by Ohm's law. The EV charging is limited as specified for the scenario, and seemingly does not contribute much to voltage stabilisation, except with short line lengths in house 2 and 3 and for long line length also in house 3.

Spring/Figure B.6: In spring, it might be expected that neither heating nor cooling is much active due to the moderate outside temperatures. However, for the small room modelled the heating caused by the increased sunshine requires cooling. Temperature adjustment is less triggered: it is not needed up to $400\,m$ for any house. This is probably caused by the reduced EV charging load and the increased PV output during the day. The voltage swing on the feeder network is rather symmetric but too wide for regular grid operation. Smart EV charging is scattered, but well visible for house 3 where many EVs can charge in parallel.

Summer/Figure B.7: In summer, the PV is the biggest load on the feeder

network. The voltage swings more to over- than to under-voltage. Here, the PV curtailment can be recognised because the peak output is beyond the 70 % boundary. For house 1, PV curtailment may not be triggered up to 300 m. But that depends on the date (how the weather evolves). On the second day it is triggered for 300 m, and on the third day straight from the start. Similar behaviour but at shorter line length is observed for house 2 and 3. The reduced charging of the employees' EVs at the office building is surprising. However, here the power management by smart EV charging seems to be effective.

The value of applying smart algorithms depends on the scenario. For the example scenario simulated, we achieve for a feeder line length of 0.5 km between the different buildings, a reduction of the absolute deviation from the normal voltage as shown in Table 4.3.

Table 4.3 – Reduction of maximum voltage deviation

	System 1	System 2	System 3
Winter	−2.15 %	−4.12 %	−6.10 %
Spring	−1.54 %	−4.73 %	−7.12 %
Summer	−2.64 %	−5.29 %	−7.97 %

This improvement of the voltage stability is calculated as difference of the extremes, according to

$$\max_{\text{samples}} \left\{ |1 - U_{\text{site}}^{\text{smart}}| \right\} - \max_{\text{samples}} \left\{ |1 - U_{\text{site}}^{\text{norm}}| \right\},$$

where $U_{\text{site}}^{\text{smart}}$ is the voltage that results if all three algorithms are invoked, and $U_{\text{site}}^{\text{norm}}$ without the smart load adjustment algorithms.

The achieved reduction may appear small, but that is due to the property that extremes tend to show up when the conditions are really poor. In such situations load adjustment alone is probably not enough to solve a voltage band violation. The traditional approach, emergency load shedding, is still required, but smart load management can do at least a part of the job most of the time.

4.3 Conclusion of Section 4

Simulating the three houses scenario sketched in the technical description, Figure 2.19, has been successfully achieved. The developed simulation framework is in beta-testing state, which is perfectly common for a university tool. Whoever uses this simulation framework has to know what he or she is doing and is assumed to improve the tool by providing feedback and amendments.

The presented exemplary simulation results show how important a bigger picture considering cross-sector inter-dependences is. Assessing the usefulness of modern energy-flow management approachers without considering potential cross-sector implications is careless.

The electricity grid may be considered too complex to grasp. However, that a physical copper line that connects buildings also causes an all in the same boat situation, should be easy to understand. Still, many believe that electricity comes out of the wall-socket. Actually, it is generated the moment it is consumed, no instant earlier.

Electricity is transported at the speed of light over shared lines (cables). These have finite capacity, which we should not ignore when designing modern energy management solutions. The developed simulation framework achieves exactly this for the academic R&D community working on solutions for the near future, in five to twenty years time, and some visionaries think beyond.

5 THE ECONOMIC SECTION

Software tools in the scientific area are commonly offered open source to gain trust and to enable adaptation and extension to the individual needs of other researchers. Selling of open source provided software is prevented by the share alike restriction. However, that does not exclude economic use by those that created the software tool and others that are granted the right to do so. Neither is the aim of the client, a university department, for whom the simulation framework is developed and who will continue development in the course of research projects.

This aim to continue using and developing the tool further and further, yields the third economic path: Use the tool to acquire project funding. As the major part of the employees at that department is paid from funded projects, having tools that support high quality research is essential. That is the true business case, which also the assessment by the experts in Section 5.1 found most promising.

5.1 The technological audit of the developed simulation framework

To establish the commercial potential for the developed simulation framework, three prominent experts were invited: Candidate of Technical Sciences, Associate Professor, Kulyk Y., Candidate of Technical Sciences, Associate Professor, Harmash V., and Candidate of Technical Sciences, Associate Professor, Kabachii V., to assess it.

The assessment of the business cases was conducted based on the criteria summarized in Table B.1 in the appendix. Each expert assessed the system's viability and potential for profitability individually. The overall level of commercial potential was determined based on the criteria outlined in Table 5.1 [50].

The variance of the results relates most likely to slight differences in the envisaged usage scenario and the yet unknown efforts required for a successful commercialisation of the prototype (university/expert tool).

Table 5.1 – Levels of technical and commercial potential of development

Arithmetic mean of scores calculated based on the experts' conclusion	Technical and commercial potential of development
0 – 10	very low
11 – 20	low
21 – 30	average
31 – 40	high
41 – 48	very high

5.1.1 Business case 3: Use your own tool for funded research

A little complex in detail, this business case is focused on persistence and sustainability, and not on profit. Having a powerful tool to perform groundbreaking simulation studies is wonderful. However, research is always based on the results of previous research: What has been found triggers new questions to be answered. Tools have to change over time to integrate the knowledge found.

Having a tool that supports performing studies at the forefront of current knowledge is a unique sales point. The monitory value is hard to predict, like the value of a prototype, but the potential for a stable income in terms of funding money, is outstanding.

Table 5.2 – Assessment of the commercial potential of business case 3

Criterion	Last name, initials of the expert		
	Kulyk Y.	Harmash V.	Kabachii V.
	Scores given by the experts		
1	4	3	4
2	4	4	4
3	4	4	4
4	4	4	4
5	3	4	3
6	3	3	1
7	4	4	4
8	4	4	4
9	4	4	4
10	3	4	4
11	4	1	1
12	4	4	4
Sum of grades	45	43	41

Arithmetic mean, variance, and standard deviation of the performed expert assessment:

$$mean = E\{X\} = \frac{\sum_{i=1}^{n} X_i}{n} = \frac{45+43+41}{3} = \frac{129}{3} = 43$$

$$variance = var\{X\} = \frac{\sum_{i=1}^{n} (E\{X\} - X_i)^2}{n} = \frac{4+0+4}{3} = \frac{8}{3} = 2.67$$

$$standard\,devation = std\{X\} = \sqrt{var\{X\}} = \sqrt{\frac{8}{3}} = 1.63$$

According to the ratings, is converting the university tool into a licensable product with all required support and guarantees (business case 1), achieving in average 27 points, a not really promising option. More promising is, with in average 33.67 points, the business case 2, to release the simulation framework open source and sell related services. Most promising is however, business case 3, achieving 43 points, which also matches the development intention and is further elaborated in the following.

5.2 The cost estimation for developing the system

During the work on the developed simulation framework, the following expenses were incurred:

1: Primary salary of executors $З_o$:

$$З_o = \frac{M}{T_p} \cdot t \tag{5.1}$$

where M represents the monthly base salary of a specific executor in UAH, T_p the average working days per month, assumedly 20 days, and t the number of working days spent on the development of the simulation framework.

In 2025, the salary ranges for researchers fall within 6 700 to 26 000 UAH/month. The resultant calculations of the primary salary of the executors related to the development is summarised in Table 5.3.

Table 5.3 – Calculation of primary salary of executors

Position title of the executor	Monthly base salary (UAH)	Payment per working day (UAH)	Number of working days/hours	Remuneration costs (UAH)
Scientific supervisor of the Student's qualification work	22250	1059.52	200 h 6 h/d	≈ 35317
Student developer (Master student)	6700	319.05	75 d	≈ 23929
Consultant in the economic section	19800	942.86	1.5 h 6 h/d	≈ 236
Other consultants	18500	880.92	3.5 d	≈ 3083
			Total	≈ 62 565

2: Additional remuneration of the executors $З_д$ is calculated as 10 % to 12 % of the primary salary, i.e.,

$$З_д = (0.10 \text{ to } 0.12) \cdot З_о \tag{5.2}$$

which results in 6257 to 7508 UAH for the example shown in Table 5.3.

3: Accruals to the payroll $Н_{зп}$ are calculated by the formula

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100} \tag{5.3}$$

where β is the rate of the unified social security contribution for mandatory state social insurance. Currently $\beta = 22\,\%$ and thus, $Н_{зп} \approx (62565 + 7435) \times 0,22 =$ 15400 UAH.

4: Material expenses M are calculated per material type i according to

$$\mathrm{M} = \sum_i \mathrm{H}_i \mathrm{Ц}_i \mathrm{K}_i - \sum_i \mathrm{B}_i \mathrm{Ц}_\mathrm{в} \tag{5.4}$$

where H_i states the amount of material used in weight units (kg), $\mathrm{Ц}_i$ states the coat of the material per weight unit (UAH/kg), K_i represents the transportation cost coefficient, in the range 1.10 to 1.15. B_i states the material disposal mass (kg), and $\mathrm{Ц}_\mathrm{в}$ states the waste price of material i (UAH/kg).

Actually, developing software needs no materials. Paper and printer ink, and all office materials in general, are covered by the lump sum stated for the required components.

5: The expenses of used components K result from

$$\mathrm{K} = \sum_i \mathrm{H_i Ц_i K_i} \tag{5.5}$$

where $\mathrm{H_i}$ is the number of components of type i, $\mathrm{Ц_i}$ is the price per component, and $\mathrm{K_i} \in [1.10, 1.15]$ is the manipulation expenses coefficient, covering for transportation, tracking and interim storage.

Following the analogy with other software developments, the cost of all material resources is approximately 900 UAH.

6: Depreciation of equipment and premises can be calculated by

$$\mathrm{A} = \frac{\mathrm{ЦH_a}}{100} \frac{\mathrm{T}}{12} \tag{5.6}$$

where Ц is the total book value of the stationary assets, $\mathrm{H_a} \in [2\,\%$ up to $25\,\%]$ is the annual depreciation rate, and $\mathrm{T}/12$ is the equipment, premises, etc. usage period in months.

The calculation of the different deprecation deduction to be considered is summarised in Table 5.4. In total $\sim 20\,000$ UAH need to be considered to cover all depreciation of used equipment and premises.

Table 5.4 – Calculation of depreciation deduction

	Book value (UAH)	Deprecation rate (%)	Usage period (months)	Deprecation deduction (UAH)
Server, laptop, printer, etc.	199000	25	6.5 (70 %)	18863.54
Department and faculty premises	152000	2.5	6.5 (35 %)	720.42
			Total	≈ 19584

7: Expenses for the electrical power consumed are calculated as

$$\text{В}_\text{е} = \frac{\text{ВПФК}_\text{п}}{\text{К}_\text{д}} \tag{5.7}$$

where В is the electricity price pre kilowatt-hour, at time of writing 7.41 UAH/kWh for business, П is the installed load capacity of the equipment, here 1.05 kW for the power supply of a server, IP router, 2 ports of the LAN switch, and the laptop assigned to the student. Ф is the actual number of operating hours, for example 730 hours in average for all. Finally, $\text{К}_\text{п} = 0.83$ is the power usage coefficient, and $\text{К}_\text{д} = 0.76$ the useful action coefficient.

The resultant total expense for electricity $\text{В}_\text{е} = 6202,90 \approx 6200$ UAH, if the assumptions mentioned are still correct.

8: Other expenses $\text{В}_\text{ін}$ can be estimated as 50 to 300 % of the initial salary of the performers. Thus we can calculate

$$\text{В}_\text{ін} = \text{К}_\text{ін} \times \text{З}_\text{о} = (0.5 \text{ to } 3.0) \times \text{З}_\text{о}. \tag{5.8}$$

where $\text{К}_\text{ін}$ is the other expenses factor. With for example $\text{К}_\text{ін} \approx 0.75$ we get $\text{В}_\text{ін} \approx 47500$ UAH.

9: The sum of all the previous expenses yields the total expenses B of the current stage, as executed and completed by the master student.

$$\mathrm{B} = \mathrm{З_о} + \mathrm{З_д} + \mathrm{Н_{зп}} + \mathrm{М} + \mathrm{К} + \mathrm{А} + \mathrm{В_е} + \mathrm{В_{ін}} \tag{5.9}$$

which for the simulation framework sums up to approximately 160 000 UAH.

10: The total costs for the development and final refinement of the works that have been done, is calculated according to

$$\mathrm{ЗВ} = \frac{\mathrm{В}}{\beta} \tag{5.10}$$

where the coefficient β characterises the completion stage yet reached. Since the delivered simulation framework is operational and further development part of the business, it can be assumed that $\beta \approx 0.97$.

In the end, the projected total expenses for the development of the simulation framework amount to approximately 165 000 UAH.

5.3 Calculation of the economic effect from potential utilisation of the developed simulation framework to acquire research funding money

The market analysis indicates that the market for a multi-sector simulation framework is small, but growing as the energy transition becomes more and more realistic. Expert tools, like serious textbooks, are not developed and sold to make profit. Such products are developed to gain knowledge and recognition.

In the case of the client for whom the simulation framework is developed, the intended economic benefit is winning the competition for public funding money. Tenders for public funding are reviewed by peer experts. An application has a chance for funding only if the promised research work is rated as being smart – specific, measurable, attractive, realistic, and timed – and worthy – promising, impactful, effective, and efficient in terms of the achieved output.

Good simulation tools help to achieve realistic results that can show the possible future impact of envisaged and upcoming changes in technology, regulation, and policy. Such studies are commonly required to proof expert believes and educated guesses, but can also raise awareness and may reveal surprising cross-correlations and unintended rebound effects.

Having a tool that can handle complex relations, fosters in-depth knowledge and reveals unexpected correlations, if correctly used by experts, supports trustworthy and explainable research results, recommendations and plausible explanation thereof. A benefit helping the authorities to rate the value of the results gained from public research funding. Therefore, having a good tool, helps to gain public funding money.

In this respect, we assume a success boost factor of 1.5 times the tender success rate reached before the tool has been available.

On the other hand reduces the tool the time needed to achieve good results, especially if the operators of the tool are experienced experts that know how to configure the tool and the scenario simulated to reveal interesting results.

The time saving factor gained from sped-up results achievement is assumed to be 3 times faster than using the integrated tools independently. Adapting the tool to address new issues and maintaining the framework stable reduce that by assumedly 30 % to a saving factor of 2.1, approximately halving the personnel cost or doubling the efficiency of the spent funding money.

5.3.1 Expected revenue gains and cost savings in the coming years

If the simulation framework is used to execute cross-sector simulation studies and is considered in funding proposals (tenders) starting with January 1st, 2026, the results manifest in the years 2026, 2027, 2028, and beyond, depending on funding lead time and funding period. The assumptions underlying the economic evaluation are summarised in Table 5.5.

Table 5.5 – Expected workload savings and added funding revenue gain

Year	Savings (working hours efficiency)	Gained funding revenue increase (kEUR)
2025	1.00	500
2026	1.03	+50
2027	1.11	+150
2028	1.33	+300
2029	1.66	+450
2030	2.00	+500

The potential increase in revenue $\Delta\Pi_i$ per year i from using the simulation framework amounts to

$$\Delta\Pi_i = \sum_1 (\Delta Ц_o \, N + Ц_o \, \Delta N)_i \, \lambda \, \rho \, (1 - \frac{\nu}{100}) \tag{5.11}$$

where $\Delta Ц_o$ is an improvement in the primary qualitative indicator (working hour savings) from using the simulation framework in this year. In our case, $\Delta Ц_o = 1.03$ in 2026, 1.11 in 2027, 1.33 in 2028, 1.66 in 2029, and 2.00 in 2030, where it saturates. The ratio reflects the improvement compared to performing the same annual workload assuming the simulation framework would not be used.

N is the main quantitative indicator (revenue) that defines the scope of the improvement from the years before the availability of the simulation framework, here $N = 500$ kEUR.

ΔN is the improvement of the main quantitative indicator (revenue) due to the use of the simulation framework in funding tenders. The annual improvement is assumed to be +50 kEUR in 2026 (due to long lead times), +100 kEUR in 2027, +150 kEUR in 2028, +150 kEUR in 2029, and +50 kEUR in 2030.

$Ц_o$ is the primary qualitative indicator (i.e., the relative working hour savings) determining the operative benefit achieved from using the simulation framework, prior the introduction of the simulation framework $Ц_o = \frac{1720}{1720} = 1.0$

n is the total number of years during which positive results from using the developed simulation framework is expected, in this scenario n = 5.

λ is the coefficient that takes into account the value-added tax (VAT) payment, which for funding money is 1.00 because no VAT is applicable.

ρ is the coefficient that considers the product's profitability. It is recommended to assume $\rho = (0.2\,\text{to}\,0,5)$, we choose $\rho = 0.2$ to consider the common user interface inefficiencies of expert tools.

ν is the corporate tax rate. As that is raised on net profits and not on revenue, and because in general universities make no profit, $\nu = 0\,\%$ for public entities.

The potential increase in revenue $\Delta\Pi_{\text{i}}$ during the first year after the tool becomes used (2026) is small because of the long lead times from designing a funding proposal till receiving a first funding advancement.

$$\Delta\Pi_1 = (1.03 \cdot 500 + 1.0 \cdot 50) \cdot 1.0 \cdot 0.2 \cdot (1 - \frac{0}{100}) = 113\,\text{kEUR}$$

In the second year, proposals submitted in the first year achieve first funding advancements. According to equation (5.11) the department revenue increases by

$$\Delta\Pi_2 = (1.11 \cdot 550 + 1.0 \cdot 150) \cdot 1.0 \cdot 0.2 \cdot (1 - \frac{0}{100}) = 152.1\,\text{kEUR}$$

In the third year, projects submitted in first and second year yield funding advancements and accordingly the budget rises and the staff uses the simulation framework more and more with increasing efficiency.

$$\Delta\Pi_3 = (1.33 \cdot 650 + 1.0 \cdot 300) \cdot 1.0 \cdot 0.2 \cdot (1 - \frac{0}{100}) = 232.9\,\text{kEUR}$$

In the fourth year, projects submitted in first, second, and third year yield funding advancements and accordingly budget and usage efficiency rises further.

$$\Delta\Pi_4 = (1.66 \cdot 800 + 1.0 \cdot 450) \cdot 1.0 \cdot 0.2 \cdot (1 - \frac{0}{100}) = 355.6\,\text{kEUR}$$

In the fifth year most projects that were submitted in the first year already ended, such that only projects submitted and won between the 2nd to 4th year contribute to the improvement. Accordingly, the raise starts to saturate, also the

work efficiency cannot be increased much further.

$$\Delta\Pi_5 = (2.00 \cdot 950 + 1.0 \cdot 500) \cdot 1.0 \cdot 0.2 \cdot (1 - \frac{0}{100}) = 480\,\text{kEUR}$$

In the end, after 5 years, the available funding budget is predictably doubled and the usage of the simulation framework should have halved the workload per achieved research result.

5.3.2 Calculating the economical value of the increased revenue gained from using the simulation framework

The total value ΠΠ can be calculated from

$$\Pi\Pi = \sum_{1}^{\mathrm{T}} \frac{\Delta\Pi_i}{(1+\tau)^t} \tag{5.12}$$

where $\Delta\Pi_i$ is the increase in revenue in each of the years when the results of the completed and implemented work are manifested, $\mathrm{T} = 5$ years is the time period during which the results of the implemented work are manifested, τ is the discount rate, assumedly 0.1 (10 %), and t is the "development-to-profit period"or "investment gestation period being the time from the initiation of the development of the simulation framework to the moment when potential revenue gains are obtained by the investor.

This value of the growth ΠΠ that can be gained from using the simulation framework results as

$$\Pi\Pi = \frac{113}{(1.1)^2} + \frac{152.1}{(1.1)^3} + \frac{232.9}{(1.1)^4} + \frac{355.6}{(1.1)^5} + \frac{480}{(1.1)^6} \approx 858\ \text{kEUR} \approx 41\,613\ \text{kUAH}$$

The present value of investments (PV) that should be allocated towards the implementation of the development would be

$$\mathrm{PV} = (1.0 \text{ to } 5.0) \times \mathrm{B}_{\text{заг}} \tag{5.13}$$

In our case $PV = 3 \times 165 = 495$ kUAH ≈ 10.2 kEUR.

The absolute effect of the potential investment made in the implementation of the developed simulation framework would be $E_{абс}$

$$E_{абс} = ПП - PV \tag{5.14}$$

The notation "ПП"stands for the present value of the increase in all net profits for the potential investor from the potential commercialization of the development.

The absolute effect from the potential implementation will be

$$E_{абс} = 858 - 10.2 = 847.8 \text{ kEUR} \approx 41\,613 - 495 = 41\,118 \text{ kUAH}$$

5.3.3 Calculating the internal rate of return (IRR) of invested capital

This internal return rate results from

$$E_{в} = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1 \tag{5.15}$$

where $E_{абс}$ is the absolute effect of the invested capital ($E_{абс} = 363$ kEUR), PV is the present value of the initial investment ($PV = 495$ kEUR), and $T_ж$ is the development lifecycle in years ($T_ж = 6$ years from start of development).

The calculation for the cross-sector simulation framework yields

$$E_{в} = \sqrt[6]{1 + \frac{41118}{495}} - 1 = \sqrt[6]{84.0667} - 1 = 1.0930 = 109.3\,\%$$

To answer the here rather theoretic question whether the investment would be also economically profitable, the minimum profitability is calculated as

$$\tau_{мін} = d + f \tag{5.16}$$

where d is the weighted average rate on deposit operations in commercial banks, in 2025 in Austria $d = (0.0000 \text{ to } 0.0025)$, and f is an indicator characterizing the riskiness of investments, $f = (0.1 \text{ to } 0.5)$.

If we assume realistic $d = 0.0010$ and $f = 0.1$ for public funding and for a more greedy funding by an investor $d = 0.03$ and $f = 0.3$, we get

$$\tau_{\text{мін}}^{\text{public}} = 0.0010 + 0.1000 = 0.1010 = 10.1\,\% \text{ and}$$
$$\tau_{\text{мін}}^{\text{investor}} = 0.0300 + 0.3000 = 0.3300 = 33.0\,\%$$

Given the internal return on investment $E_в = 109.3\,\% \gg 10.1/33.0\,\% = \tau_{\text{мін}}$, it becomes clear that the cross-sector simulation framework is a very promising investment. However, the "investment" stems from public funding of the simulation framework development. No alternative usage of this money is eligible. In addition, results the high IRR from using revenue as the basis for calculation. Therefore, it could be argued that only the 9.3 % above 100 % contribute to actual growth, kind of sustainable profit.

5.3.4 Calculation of the payback period

The payback period $T_{ок}$ is calculated by

$$T_{ок} = \frac{1}{E_в} \tag{5.17}$$

which for the simulation framework results in $T_{ок} = \frac{1}{1.093} = 0.91$ years. This is a very short time for investments in a persistent research infrastructure, where a return of investment is commonly expected in decades, not years.

In practice, long lead times toward funding advancements prevent a short payback time. In addition, we need to consider that the benefit is increased funding revenue and efficiency in using research funds, not net profit. Non of the gained

funding revenue can be used to pay interest rates to investors. The only "investor" is the funding agency that administers public financed research funding programs.

5.3.5 Relationship between the internal rate of return of potential investments and the inflation rate

If the inflation rate $\tau = 5\,\%$ instead of $10\,\%$, then

$$\begin{aligned} \text{ПП} &= \frac{113}{(1.05)^2} + \frac{152.1}{(1.05)^3} + \frac{232.9}{(1.05)^4} + \frac{355.6}{(1.05)^5} + \frac{480}{(1.05)^6} \\ &= 1062.3 \text{ kEUR} \approx 51\,521 \text{ kUAH} \end{aligned}$$

and we get

$$\text{E}_{\text{абс}} = 51\,521 - 495 = 51\,096 \text{ kUAH}$$

and an IRR of

$$\text{E}_{\text{в}} = \sqrt[6]{1 + \frac{51\,096}{495}} - 1 = \sqrt[6]{104,08} - 1 = 1,1689 \approx 117\,\%.$$

If we perform this calculation for increasing inflation rates, i.e., $20\,\%$, $50\,\%$, $100\,\%$, and $200\,\%$, we get $\text{E}_{\text{в}} = 96.2\,\%, 68.1\,\%, 41.1\,\%,$ and $14.7\,\%$ respectively, as shown in Figure 5.1.

The analysis of the charts in Figure 5.1 reveals that even up to a inflation rate $\tau > 200\,\%$ public investment might be economically interesting. Institutional investors would invest up to slightly above $100\,\%$ inflation. However, a final decision on investments requires additional calculations and considerations, e.g., increasing the investment risk level, adjusting the demand for the development, availability of public funding, and variations and combinations of different business cases, as well as financial and economic prognoses for the next five to ten years.

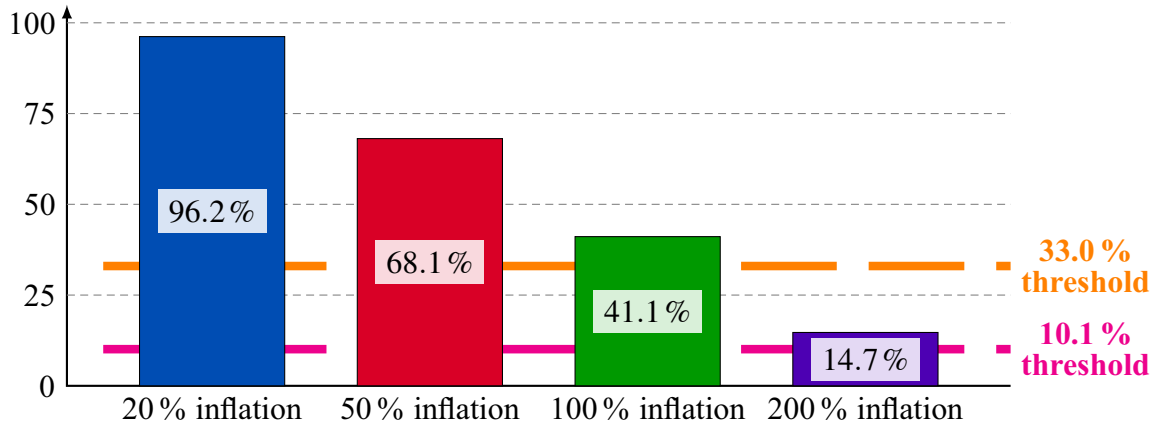


Figure 5.1 – Modelling of the relationship between internal return of investment rate and an inflation rate of 20 %, 50 %, 100 %, and 200 %

5.4 Conclusion of Section 5

All the key technical and economic indicators of the developed cross-sector simulation framework have been successfully achieved as defined by the client. The outcomes of the performed economic part of the master's qualification theses are summarised in the Table 5.6.

Table 5.6 – The outcomes of the performed economic part

Indicator	Defined in Technical task	Attained in Master's thesis	Conclusion
Development expense (kUAH)	< 200	165	achieved ✓
Absolute effect from implementing (kUAH)	> 24 250	41 613 (10 % inflation)	achieved ✓
Internal Rate of Return on Investment (%)	> 10.1/33.0 %	109.3 % (10 % inflation)	achieved ✓
Payback Period of Investment (years)	< 3 years	0.91 years	achieved ✓

The requirements stated by the client for the development have been successfully achieved.

CONCLUSIONS

The literature survey revealed several sector specific tools that can simulate systems very well, but only individually. Some tools cover also systems from other sectors as peripheral add-on squeezed into the sector specific view. General purpose simulation tools are also available, but no tool or set of tools that focuses on the inter-operation and inter-dependences of energy related systems.

In this master work, existing tools for different sectors, pandapower, OpenModelica and smartDCsim, are linked together by effectively parsing the output of one as input to the other and vice versa. Thereby, a cross-sectorial co-simulation environment is achieved based on established sector specific system modelling tools. The overarching simulation control is provided by an agnostic event simulation core and assignable data recording and visualisation options. Selecting and successfully integrating feasible tools to effectively simulate several buildings equipped with modern energy assets and appliances, and inter-connected by the local feeder network, covers the core research question and the development task of the performed work.

If we integrate the developed tool into the comparison done in Section 1.2, Table 1.2, we see that it does not support all features, as shown in Table 6.1, but inserting it in Table 1.1, combining the features of pandapower, OpenModelica and smartDCsim, it proves to serve all needs stated, as shown in Table 6.2.

Table 6.1 – Analogous simulation environment comparison

feature → ↓ **tool**	electricity	heat	mobility	H_2 e-fuels	var. demand	distr. coord.	self optim.	open source
Calliope	+	+	+	+	-	-	-	+
OpenModelica	+	+	+	+	+	+	-	+
JuliaSim	+	+	+	+	+	+	+	+
MATLAB	+	+	+	+	+	+	+	-
pandapower	+	-	-	-	-	-	-	+
new tool	+	+	+	-	+	+	-	○

Table 6.2 – Features supported by tools in the respective energy sector

feature → ↓**tool**	power flow	energy flow	demand model	ext. signals	contr. charge	trans. analysis	demand opt.	data in-export
pandapower	+	-	-	-	-	+	-	+
Modelica	+	-	+	-	-	+	-	+
AixLib	-	+	+	+	-	+	-	+
openModelica	+	+	+	+	-	+	+	+
bi-levelEVsim	-	+	+	-	+	-	-	-
smartDCsim	+	+	+	-	+	-	+	-
oemof	-	-	-	+	-	-	+	+
Calliope	-	+	+	-	-	-	-	-
new tool	+	+	+	+	+	+	+	+

The developed simulation framework serves to evaluate energy-flow management approaches, bringing light into the design of: smart systems control, cross-sector efficiency measures, demand-response coordination approaches, and other procedures proposed to support the energy transition across all sectors. The chosen combination of tools and their integration based on an event-based core proved viable. In combination with automated simulation steps and convenient visualisation options the simulation framework serves the purpose to analyse and compare cross-sector integration and the correlation introduced by the connecting electricity grid.

Balancing EV charging and local generation has often been proposed to be the smart solution to solve the volatility disadvantage of renewable sources. With the developed simulation framework this plausible postulation now can be evaluated and approved, as done in Section 4.1. The simulation of the scenario stated by the client has been achieved as intended, as detailed in Section 4.2 and shown in the Figures B.5, B.6, and B.7, presented in the appendix. The gained information is utile to design efficient and cooperative energy management systems and to implement coordination means and paradigms that effectively support the energy transition and not the energy business only.

Developing the simulation framework that effectively models the complete system-of-systems bridging sector boundaries required and trained several skills,

including the design of a layered software architecture, back-end and front-end programming, numeric analysis, stochastic evaluation, modelling, statistical evaluation using mathematics tools, integration of visualisation libraries, and composing a utile GUI. The result is a configurable and scalable simulation of scenarios, implemented to analyse smart energy management algorithms and their impact on balancing the load across the local grid section, from feeder line to distribution region. The implemented visualisation of statistical results enables the accurate assessment of the behaviour, the performance, and the impact caused on the electricity grid.

Applying smart algorithms ensures a reduction in the dynamic deviation from normal voltage at the local electricity supply connection point. How effective, depends on the scenario and can effectively and efficiently be evaluated using the developed simulation tool. The exemplary simulation study covering three automation objects, being the three buildings connected by the local feeder network evaluated in Section 4.2.3, each with a smart controlled PV system, one to eight EV charging stations, and a small electric heating and cooling system, showed an absolute reduction in deviation from normal voltage from 2.15 to 6.10 % for the winter period, from 1.54 to 7.12 % for spring, and from 2.64 to 7.97 % for summer, where the largest improvements are achieved by the smart building at the far end of the feeder line, which also contributes the most load adjustments.

The research objective to develop a tool to evaluate cross-sector energy-flow management, including electric load management, EV charging management, heating and cooling management, and cross-sector energy demand adjustment signalling, has been overall achieved. Cross-sector simulation addressing multi-dimensional opportunities has been demonstrated successfully. The developed user interface provides the means to save and re-load scenario configurations, conveniently visualise and evaluate simulation results and storing of results data in CSV files and generated graphs in publishable quality PNG and scalable SVG format. All client requirements have been considered and implemented with exemplary rigour.

LIST OF REFERENCED LITERATURE

1. *European Commission*. Clean Energy For All Europeans. 2016-11-30. EUR-Lex – 52016DC0860 – EN. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52016DC0860 (online; accessed: 2024-09-28).
2. *European Commission*. REPowerEU Plan. 2022-05-18. EUR-Lex – 52022DC0230 – EN. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2022:230:FIN (online; accessed: 2024-09-28).
3. Smart grids: the forgotten key to decarbonization / Deven Chhaya, Nicolas Leonetti, Ciarán Rabbitt, Sophie Shen // *KPMG – Plugged In – Power and utilities magazine*. 2024. no. 3. P. 15–21. URL: https://kpmg.com/us/en/articles/2024/smart-grids.html (online; accessed: 2024-09-28).
4. Transforming small-island power systems: Technical planning studies for the integration of variable renewables / Francisco Gáfaro, Tomás Jil, Gayathri Nair et al. International Renewable Energy Agency, 2018. URL: https://www.irena.org/publications/2019/Jan/Transforming-small-island-power-systems (online; accessed: 2024-09-28).
5. *Drtil Michael*. Why are smart grids important? 2024. URL: https://www.iea.org/reports/smart-grids (online; accessed: 2024-09-28).
6. *Li Ruonan, Mahalec Vladimir*. Greenhouse gas emissions reduction by cross-sector integration of energy systems: Optimal sizing of integrated entities // *Energy Conversion and Management*. 2021. Vol. 248. P. 114788. URL: https://www.sciencedirect.com/science/article/pii/S0196890421009640.
7. *European Parliament, Council of the European Union*. DIRECTIVE (EU) 2018/2001 – RED II. No. L 328/82. Official Journal of the European Union, 2018. URL: http://data.europa.eu/eli/dir/2018/2001/oj.

8. *Cruz Igor, Ilić Danica Djurić, Johansson Maria T.* Using flexible energy system interactions amongst industry, district heating, and the power sector to increase renewable energy penetration // *Energy Efficiency*. 2023. Vol. 16, no. 6. P. 53. URL: https://doi.org/10.1007/s12053-023-10134-4.
9. *European Parliament Council of the European Union*. DIRECTIVE (EU) 2012/27/EU – EED. No. L-315/1. Official Journal of the European Union, 2012. URL: http://data.europa.eu/eli/dir/2012/27/oj.
10. *European Parliament Council of the European Union*. DIRECTIVE (EU) 2023/1791 – EEF. No. L-231/1. Official Journal of the European Union, 2023. URL: http://data.europa.eu/eli/dir/2023/1791/oj.
11. *Reallabor 100% EE Waldviertel Betriebs-GmbH*. Reallabor Waldvirtel. 2025-2029. URL: https://reallaborw4.at/ (online; accessed: 2025-10-16).
12. *Марія Сергіївна Форкалюк, Бісікало Олег Володимирович*. Cross-Sector Energy Systems Co-Simulation Framework Desing // LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025). Vinnytsya National Technical University. С. УДК 004.94 : 620.92. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23036/19178 (дата звернення: 25.02.2025).
13. *Форкалюк М.* Оцінка швидкої зарядки електромобілів вздовж автомагістралей за допомогою моделювання скінченної багатосерверної системи масового обслуговування / Марія Форкалюк, Олег Бісікало // Медико-технічна співпраця заради перемоги: актуальні завдання медичної, біологічної фізики та інформатики: мат. доп. та вист. III Всеукр. наук.-практ. конф. з міжн. участю (5-6 квітня 2024 р., ВНМУ ім. М.І. Пирогова, м. Вінниця). – Вінниця : Едельвейс, 2024 –. С. 176–181. – ISBN: 978-617-7417-21-6. URL: https://dspace.vnmu.edu.ua/123456789/6560 (дата звернення: 01.06.2024).

14. *Lukic Filip, Forkaliuk Maria S., Franzl Gerald*. Integrating Fast Electric Vehicle Charging and Charge-Site Power-Limiting in SUMO // Simulation Tools and Techniques / Ed. by Angel A. Juan, José-Luis Guisado-Lizar, María-José Morón-Fernández, Elena Perez-Bernabeu. Cham : Springer Nature Switzerland, 2025. P. 88–104.
15. Modelling and Simulation of Smart Electric Vehicle Charging Sites / Gerald Franzl, Mariia S. Forkaliuk, Albert Treytl, Oleg V. Bisikalo // 2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA). 2025. P. 1–8.
16. *Franzl Gerald, Forkaliuk Mariia S., Wilker Stefan*. Flexibility constraints specification with volatility assessment for reliable demand side flexibility // 2025 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT- Europe). 2025. in print.
17. *Forkaliuk Mariia S., Franzl Gerald, Treytl Albert*. Simulating the Contribution of Smart EV Charging and PV Curtailment to Grid Congestion Mitigation // 17th International Conference on Applied Energy (ICAE2025). 2025. accepted.
18. *European Parliament Council of the European Union*. DIRECTIVE (EU) 2019/944 – IME. No. L 158/125. Official Journal of the European Union, 2019. URL: http://data.europa.eu/eli/dir/2019/944/oj.
19. Pandapower — An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems / L. Thurner, A. Scheidler, F. Schäfer et al. // *IEEE Transactions on Power Systems*. 2018. Nov. Vol. 33, no. 6. P. 6510–6521.
20. *pandas via NumFOCUS, Inc.* pandas. 2024. URL: https://pandas.pydata.org/ (online; accessed: 2024-12-16).
21. *Zimmerman Ray D., Murillo-Sánchez Carlos E.* MATPOWER User's Manual, Version 8.0, 2023. URL: https://matpower.org/docs/MATPOWER-manual.pdf (online; accessed: 2024-12-16).
22. *Zimmerman Ray, Lincoln Richard*. Package pypower :: Module caseformat. 2011. URL: https://rwl.github.io/PYPOWER/api/pypower.caseformat-module.html (online; accessed: 2024-12-16).

23. *Xiang Yu, Salemink Peter, Bharambe Nitish et al.* PowerGridModel/power-grid-model. URL: https://github.com/PowerGridModel/power-grid-model.
24. *RTE France*. LightSim2Grid – Available “solvers” (doc in progress). 2019. URL: https://lightsim2grid.readthedocs.io/en/latest/solvers.html.
25. PowerModels.jl: An Open-Source Framework for Exploring Power Flow Formulations / Carleton Coffrin, Russell Bent, Kaarthik Sundar et al. // 2018 Power Systems Computation Conference (PSCC). 2018. June. P. 1–8.
26. *Vecchi Francesca, Stasi Roberto, Berardi Umberto*. Modelling tools for the assessment of Renewable Energy Communities // *Energy Reports*. 2024. Vol. 11. P. 3941–3962. URL: https://www.sciencedirect.com/science/article/pii/S2352484724001926.
27. *OpenRemote Inc.* The 100% Open Source IoT Platform for Manufacturers and Integrators. URL: https://www.openremote.io/ (online; accessed: 2024-11-28).
28. *Open Home Foundation*. Home Assistant. URL: https://www.home-assistant.io/ (online; accessed: 2024-11-28).
29. *Mattsson Sven Erik, Elmqvist Hilding*. Modelica – An International Effort to Design the Next Generation Modeling Language // *IFAC Proceedings Volumes*. 1997. Vol. 30, no. 4. P. 151–155. 7th IFAC Symposium on Computer Aided Control Systems Design (CACSD ’97), Gent, Belgium, 28-30 April. URL: https://www.sciencedirect.com/science/article/pii/S1474667017436287.
30. Introduction to Modelica Modeling for Building Systems – A Tutorial Developed for MIT Class 4.421: Space-Conditioning Systems for Low-Carbon Buildings : Rep. / Lawrence Berkeley National Laboratory ; Executor: David Blum : 2021. URL: https://simulationresearch.lbl.gov/modelica/downloads/workshops/2021-03-29-mit/IntroductionToModelicaModelingForBuildingSystems.pdf (online; accessed: 2024-10-26).

31. *Tiller Michael M.* Modelica by Example. URL: https://mbe.modelica.university/ (online; accessed: 2024-10-25).
32. AixLib: an open-source Modelica library for compound building energy systems from component to district level with automated quality management / Laura Maier, David Jansen, Fabian Wüllhorst et al. // *Journal of Building Performance Simulation*. 2024. Vol. 17, no. 2. P. 196–219. https://doi.org/10.1080/19401493.2023.2250521.
33. *IBPSA Modelica Working Group*. Modelica IBPSA library. online; accessed: https://github.com/ibpsa/modelica-ibpsa (online; accessed: 2024-11-04).
34. *Dassault Systèmes AB*. Dymola – Multi-Engineering Modeling and Simulation based on Modelica and FMI. URL: https://www.3ds.com/products/catia/dymola (online; accessed: 2014-11-04).
35. *RWTH Aachen University – E.ON Energy Research Center*. Energysystemsimulation with Modelica/Dymola. URL: https://www.ebc.eonerc.rwth-aachen.de/cms/e-on-erc-ebc/forschung/ausstattung/simulation/~obmj/1d/?lidx=1 (online; accessed: 2024-11-04).
36. *Dassault Systèmes AB*. Free Trial Version of DYMOLA for Windows. The Trial Version has the functionality of DYMOLA standard, but with limited capacity. URL: https://discover.3ds.com/free-trial-version-of-dymola-for-windows (online; accessed: 2014-11-04).
37. *Farimani Foad S.* JModelica. URL: https://github.com/JModelica/JModelica (online; accessed: 2024-10-26).
38. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development / Peter Fritzson, Adrian Pop, Karim Abdelhak et al. // *Modeling, Identification and Control*. 2020. Vol. 41, no. 4. P. 241–295.
39. How will electric vehicles affect traffic congestion and energy consumption: an integrated modelling approach / Artur Grigorev, Tuo Mao, Adam Berry et al. // 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). 2021. P. 1635–1642.

40. *Forkaliuk Maria, Franzl Gerald, Bisikalo Oleg*. Evaluating fast charging of electric vehicles along motorways using finite multi-server queueing system simulation // *Information Technologies and Computer Engineering*. 2024. Vol. 21, no. 2. P. 77–90. URL: https://itce.com.ua/web/uploads/pdf/IT_2024_2_7.pdf (online; accessed: 2024-11-11).
41. oemof.solph—A model generator for linear and mixed-integer linear optimisation of energy systems / Uwe Krien, Patrik Schönfeldt, Jann Launer et al. // *Software Impacts*. 2020. Vol. 6. P. 100028. URL: https://www.sciencedirect.com/science/article/pii/S2665963820300191.
42. *Pfenninger Stefan, Pickering Bryn*. Calliope: a multi-scale energy systems modelling framework // *Journal of Open Source Software*. 2018. Vol. 3, no. 29. P. 825.
43. *Forkaliuk M. S., Bisikalo O. V.* A Simulation Model of a Finite Multi-Server Queuing System for Evaluating Fast Charging of Electric Vehicles Along Motorways, bachelor thesis, НТКП ВНТУ, Факультет інтелектуальних інформаційних технологій та автоматизації, Вінниця, Україна. 2024. URL: https://iq.vntu.edu.ua/repository/getfile.php/8332.pdf (online; accessed: 2025-10-16).
44. *Franzl Gerald*. Queueing models for multi-service networks : Ph. D. thesis / Gerald Franzl ; Vienna University of Technology. 2015.
45. *Döring Markus*. standardlastprofile. CC0. 2023. URL: https://flrd.github.io/standardlastprofile/ (online; accessed: 2025-02-02).
46. *Grandjean A., Adnot J., Binet G.* A review and an analysis of the residential electric load curve models // *Renewable and Sustainable Energy Reviews*. 2012. Vol. 16, no. 9. P. 6539–6565. URL: https://www.sciencedirect.com/science/article/pii/S1364032112004820.
47. *Lomax K. S.* Business Failures: Another Example of the Analysis of Failure Data // *Journal of the American Statistical Association*. 1954. Vol. 49, no. 268. P. 847–852.

48. *Open-Meteo.com*. Historical Weather API. 2022-2025. URL: https://open-meteo.com/en/docs/historical-weather-api (online; accessed: 2025-08-17).
49. *Harchol-Balter Mor*. Performance Modeling and Design of Computer Systems. New York, USA : Cambridge University Press, 2013. ISBN: 978-1-107-02750-3.
50. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт : Rep. / Вінницький національний технічний університет, інформаційний редакційно-видавничий центр ; виконавець: В. О. Козловський, О. Й. Лесько, В. В. Кавецький : Вінниця, ВНТУ, 2021. URL: https://epvm.vntu.edu.ua/wp-content/uploads/2021/09/Ekonom_magister_tehn_2021.pdf (дата звернення: 17.11.2025).

Appendices

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«ІНТЕГРАЦІЯ ІНСТРУМЕНТІВ МОДЕЛЮВАННЯ З СЕКТОРІВ ЕЛЕКТРОЕНЕРГЕТИКИ, МОБІЛЬНОСТІ ТА ОПАЛЕННЯ В СИСТЕМУ МОНІТОРИНГУ ПОТОКІВ ЕНЕРГІЇ, КЕРОВАНУ ПОДІЯМИ»
08-31.МКР.012.02.000 ТЗ

Керівник роботи:
д.т.н., проф. каф. АІІТ
Олег БІСІКАЛО
«16» жовтня 2025 р.

Виконавець:
ст. гр. 1АКІТ-24м
Марія ФОРКАЛЮК
«16» жовтня 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Інтеграція інструментів моделювання з секторів електроенергетики, мобільності та опалення в систему моніторингу потоків енергії, керовану подіями.

Дослідження та розробка сучасних систем енергопостачання та розподілу, інтелектуальних систем управління енергоспоживанням, автоматизації реагування на попит, диспетчерського контролю та збору даних, технологій кооперативної експлуатації.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є розробка нової симуляційної моделі для академічних досліджень, що проводяться в рамках програм, фінансованих з державного бюджету та / або грантових коштів, яка забезпечить:

- графічний інтерфейс, що дозволяє користувачеві налаштувати будь-який можливий сценарій від окремого будинку до цілого району чи регіону;
- використання спеціалізованих сучасних інструментів моделювання від спільноти відкритого програмного забезпечення, розроблених для різних секторів (електроенергетика, мобільність, опалення);
- генерацію випадкових навантажень для нових технологій: зарядка електромобілів, теплові насоси, системи акумуляторних батарей;
- комплексну статистичну обробку, що дає середні значення з довірчими інтервалами, стандартним відхиленням та 10% і 90% процентилями;
- визначення послідовностей тестів зі зміною параметрів або повтореннями з новими випадковими навантаженнями для надійної статистичної оцінки;
- зручну візуалізацію результатів з можливістю масштабування та обертання графіків і збереження поточного виду як масштабованої векторної графіки.

Призначення симуляційної платформи полягає у забезпеченні автоматизованої міжсекторальної симуляції взаємодіючих систем у сферах енергетики, мобільності, опалення, вентиляції та кондиціонування повітря за допомогою сучасних інструментів моделювання систем вільного користування. Необхідно також забезпечити користувачеві максимальну гнучкість у складанні сценаріїв симуляції та виборі пріоритетів оцінки, надаючи зручні інструменти та засоби візуалізації, що полегшують презентацію та інтерпретацію отриманих результатів симуляції.

4. Джерела розробки

1. Smart grids: the forgotten key to decarbonization / Deven Chhaya, Nicolas Leonetti, Ciar ´an Rabbitt, Sophie Shen // KPMG – Plugged In – Power and utilities magazine. 2024. no. 3. P. 15–21. URL: https://kpmg.com/us/en/articles/2024/smart-grids.html
2. Pandapower — An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems / L. Thurner, A. Scheidler, F. Sch ¨afer et al. // IEEE Transactions on Power Systems. 2018. Nov. Vol. 33, no. 6. P. 6510–6521.
3. Forkaliuk Maria, Franzl Gerald, Bisikalo Oleg. Evaluating fast charging of electric vehicles along motorways using finite multi-server queueing system simulation // Information Technologies and Computer Engineering. 2024. Vol. 21, no. 2. P. 77–90. URL: https://itce.com.ua/web/uploads/pdf/IT_2024_2_7.pdf.
4. G. Franzl, M. S. Forkaliuk, A. Treytl and O. V. Bisikalo, "Modelling and Simulation of Smart Electric Vehicle Charging Sites," 2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA), Porto, Portugal, 2025, pp. 1-8, doi: 10.1109/ETFA65518.2025.11205798.
5. Introduction to Modelica Modeling for Building Systems – A Tutorial Developed for MIT Class 4.421: Space-Conditioning Systems for Low-Carbon Buildings : Rep. / Lawrence Berkeley National Laboratory ; Executor: David Blum : 2021. URL: https://simulationresearch.lbl.gov/modelica/downloads/workshops/2021-03-29-mit/IntroductionToModelicaModelingForBuildingSystems.pdf
6. AixLib: an open-source Modelica library for compound building energy systems from component to district level with automated quality management / Laura Maier, David Jansen, Fabian W ¨ullhorst et al. //

Journal of Building Performance Simulation. 2024. Vol. 17, no. 2. P. 196–219. https://doi.org/10.1080/19401493.2023.2250521

7. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development / Peter Fritzson, Adrian Pop, Karim Abdelhak et al. // Modeling, Identification and Control. 2020. Vol. 41, no. 4. P. 241–295.
8. How will electric vehicles affect traffic congestion and energy consumption: an integrated modelling approach / Artur Grigorev, Tuo Mao, Adam Berry et al. // 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). 2021. P. 1635–1642.
9. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт : Rep. / Вінницький національний технічний уні-верситет, інформаційний редакційно-видавничий центр ; виконавець: В. О. Козловський, О. Й. Лесько, В. В. Кавецький : Вінниця, ВНТУ, 2021. URL: https://epvm.vntu.edu.ua/wp-content/uploads/2021/09/Ekonom_magister_tehn_2021.pdf

5. Показники призначення

5.1. Основні технічні характеристики системи моделювання і моніторингу потоків енергії, керовану подіями, що інтегрує незалежні системи підзарядки електромобілей, фотоелектричні системи енергопостачання, а також системи опалення, вентиляції та кондиціонування повітря комплексу житлових і офісних будівель.

Функціональні можливості:

5.1.1 Детальне моделювання галузевих систем:

- Моделювання різних часів прибуття та паркування електромобілів (smartDCsim);
- Врахування різних потреб у заряджанні та швидкості заряджання (smartDCsim);
- Застосування алгоритмів інтелектуального заряджання (smartDCsim);
- Моделювання систем опалення, вентиляції та кондиціонування повітря за допомогою популярного сучасного інструменту Modelica;
- Розрахунок потоків енергії за допомогою популярного сучасного інструменту pandapower;

- Генерація випадкових значень за допомогою вибраних і настроюваних законів розподілу: Normal, Lomax, Erlang, Beta, Uniform, Deterministic;
- Використання записаних даних про погоду для моделювання властивостей, що залежать від погоди.

5.1.2 Зручний графічний інтерфейс користувача, що відображає результати, та різноманітні опції збереження:

- Розробку та конфігурацію сценаріїв у графічному інтерфейсі користувача;
- Автоматичне покрокове моделювання зі зміною параметрів;
- Візуалізацію вільно вибираних результатів у графічному інтерфейсі користувача;
- Збереження необроблених результатів у файлі CSV та графіків у форматі SVG і PNG.

5.1.3 Комплексна статистична оцінка результатів:

- Розрахунок середніх значень з довірчими інтервалами;
- Розрахунок стандартного відхилення, а також 10 % і 90 % процентилей.

5.2. Мінімальні системні вимоги

5.2.1 Апаратне забезпечення:

- Процесор: Intel Core i5 / AMD Ryzen 5;
- Оперативна пам'ять: 16 GB RAM, (рекомендовано 32 ГБ для одночасного запуску фронтенду, бекенду та симуляцій на довгі часові проміжки);
- Місце на диску: 2 GB для програмного забезпечення та бібліотек;

5.2.2 Програмне забезпечення:

- Операційна система: Windows 10/11, Linux (Ubuntu 20.04+);
- Python: версія 3.9–3.12, встановлені пакети pandapower, resampy, OMPython;
- Java: версія 17 або новіша (рекомендовано LTS);
- OpenModelica: встановлений OpenModelica (версія 1.19+), налаштований для інтеграції з Python;
- Node.js: версія 18+ (для запуску Next.js фронтенду);
- Браузер: Google Chrome або Microsoft Edge (для роботи з UI).

5.3. Вхідні дані

5.3.1 Типові метеорологічні дані за рік у форматі .epw/.mos: з веб-сайту https://energyplus.net/weather.

5.3.2 Modelica model templates from AixLib.

5.3.3 Конфігурація симуляції задається користувачем через графічний інтерфейс і зберігається у форматі JSON. Вхідні дані включають:

- Загальні параметри: кількість та тривалість кроків симуляції.
- Параметри об'єктів: ліміти потужності, наявність та характеристики акумуляторних батарей.
- Параметри зарядних станцій: кількість точок зарядки та місць для очікування.
- Параметри електромобілів: кількість, час прибуття, тривалість паркування та рівень заряду.
- Параметри домогосподарств: потужність систем опалення / охолодження, характеристики сонячних електростанцій та потрібна температура.

5.4 Результати роботи програми

5.4.1 Графічне представлення результатів:

Програма генерує набір графіків для візуального аналізу динамічних процесів. Усі графіки можна експортувати у форматах PNG та SVG, а також зберегти разом у єдиний ZIP-архів.

Графіки для зарядної інфраструктури: аналіз ймовірності блокування та споживання потужності.

Графіки для електромобілів: відстеження процесу та рівня заряду.

Графіки для електромережі та домогосподарств: аналіз стабільності напруги, температури в приміщеннях, генерації сонячної енергії та побутового споживання.

5.4.2 Статистичні характеристики випадкових результатів:

Середнє значення з довірчими інтервалами та стандартним відхиленням;

10 % та 90 % процентилі.

5.4.3 Збереження параметрів симуляції:

Файл конфігурації (JSON) зберігає повний набір вхідних параметрів, що забезпечує можливість точного відтворення та верифікації результатів симуляції.

6. Економічні показники

До економічних показників входять:

- витрати на розробку – до 200 тис. грн.;
- узагальнений коефіцієнт якості розробки – більше 2-х;
- термін окупності – до 3х років.

7. Стадії розробки:

1. Розділ 1 «Стан проблеми та огляд аналогічних робіт» має бути виконаний до 05.10.2025 р.
2. Розділ 2 «Інтеграція моделей та підхід до спільного моделювання» має бути виконаний до 25.10.2025 р.
3. Розділ 3 «Розробка програмного забезпечення» має бути виконаний до 20.11.2025 р.
4. Розділ 4 «Приклад оцінки багатосекторіального сценарію» має бути виконаний до 28.11.2025 р.
5. Розділ 5 «Економічний розділ» має бути виконаний до 01.12.2025 р.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 14.11.2025 р.
2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025 р.
3. Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 р. до 19.12.2025 р.

Appendix B
(compulsory)

ILLUSTRATIVE PART

«INTEGRATION OF MODELLING TOOLS FROM ELECTRICITY, MOBILITY AND HEATING SECTORS INTO AN EVENT-DRIVEN ENERGY-FLOW MONITORING FRAMEWORK»

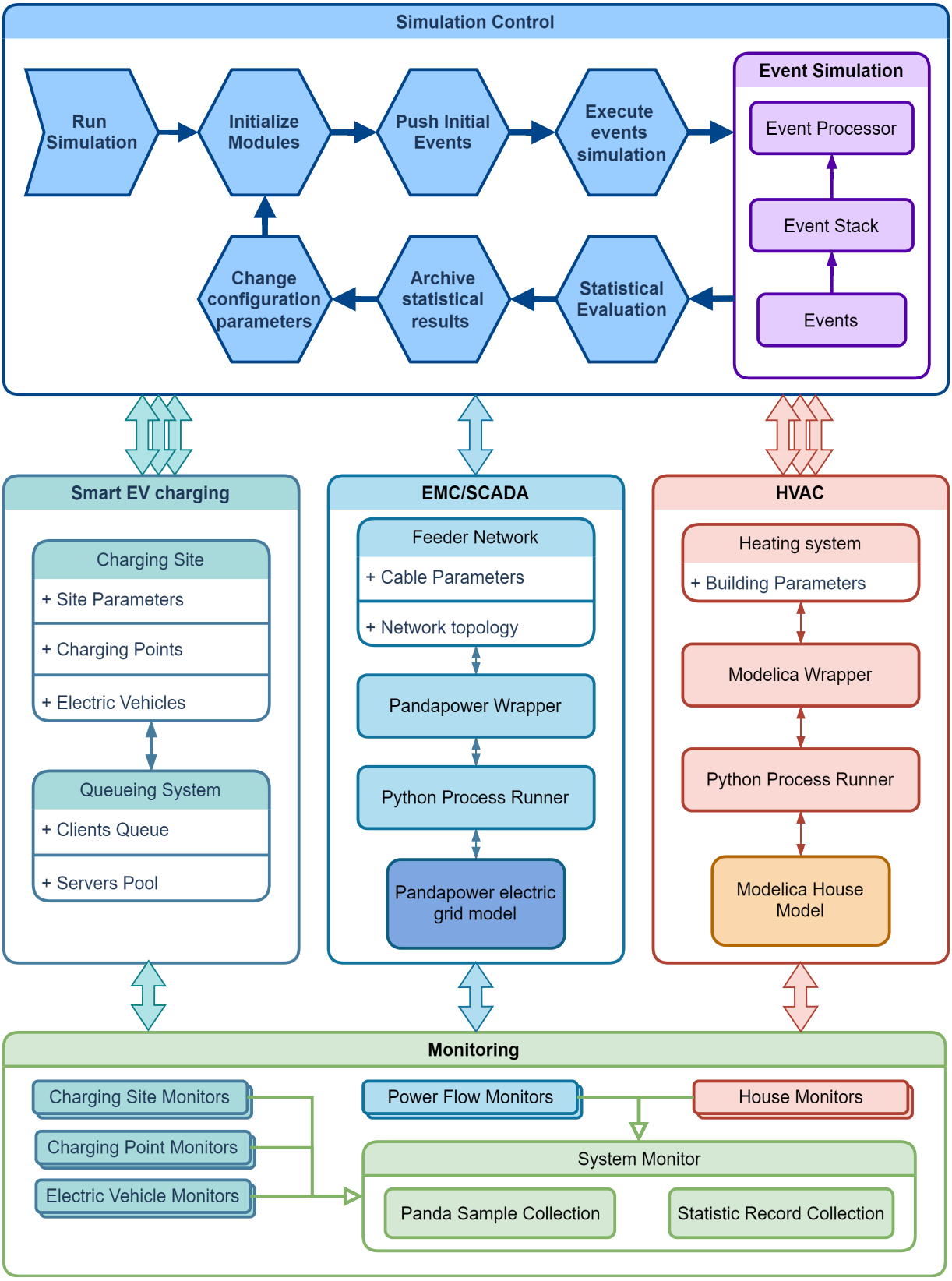


Figure B.1 – The simulation framework's architecture

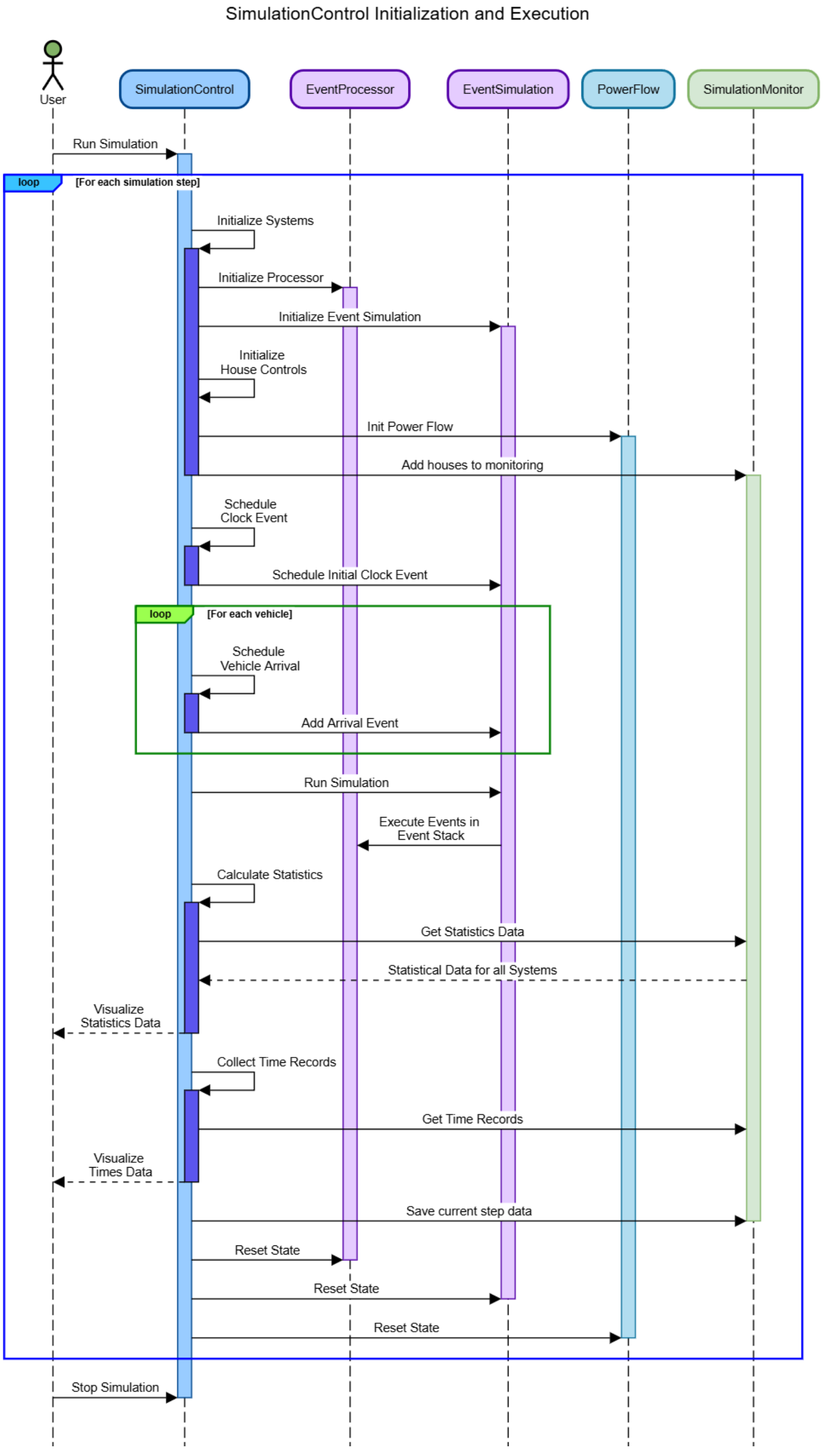


Figure B.2 – Simulation Control sequence diagram

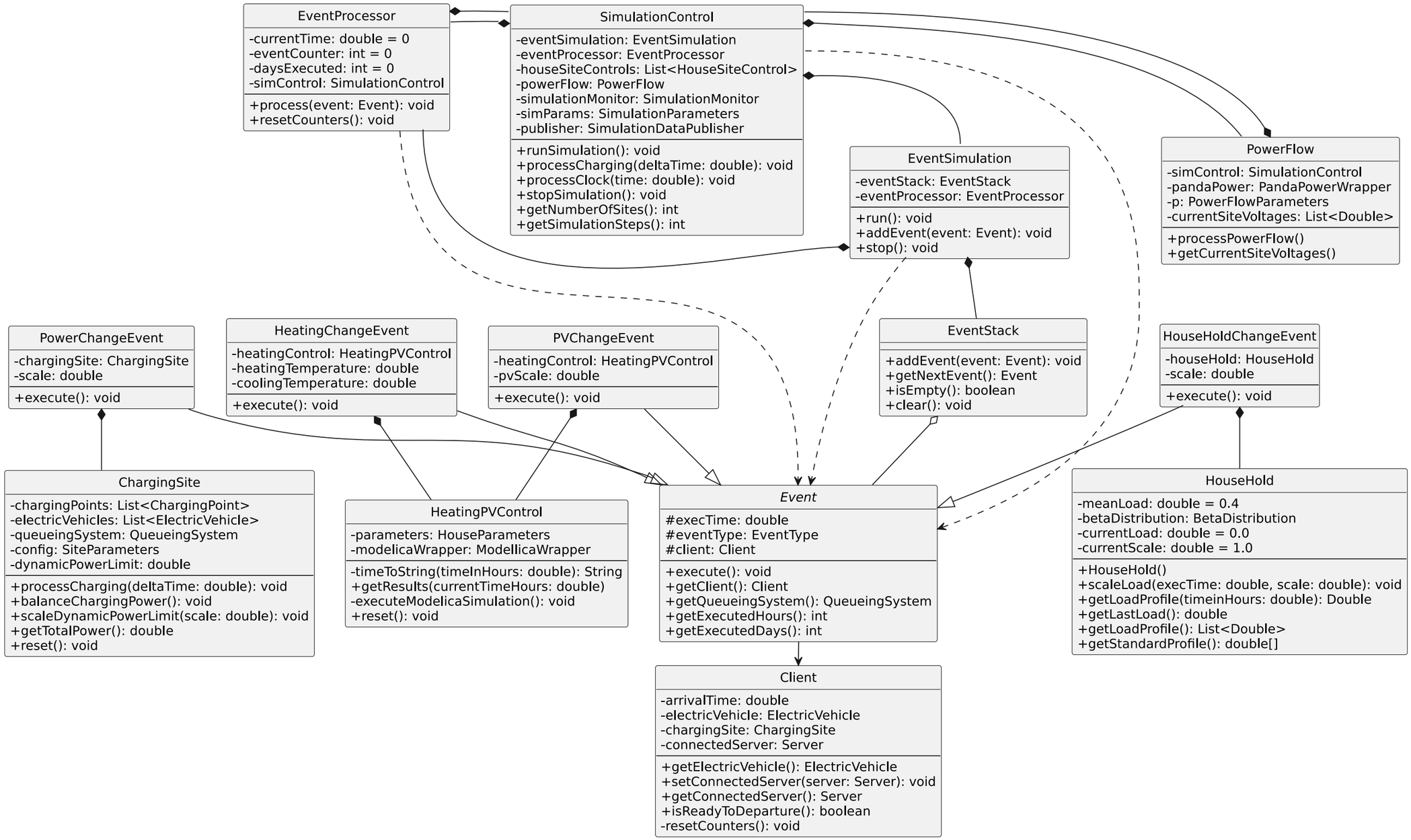


Figure B.3 – Class diagram of the simulation framework modules

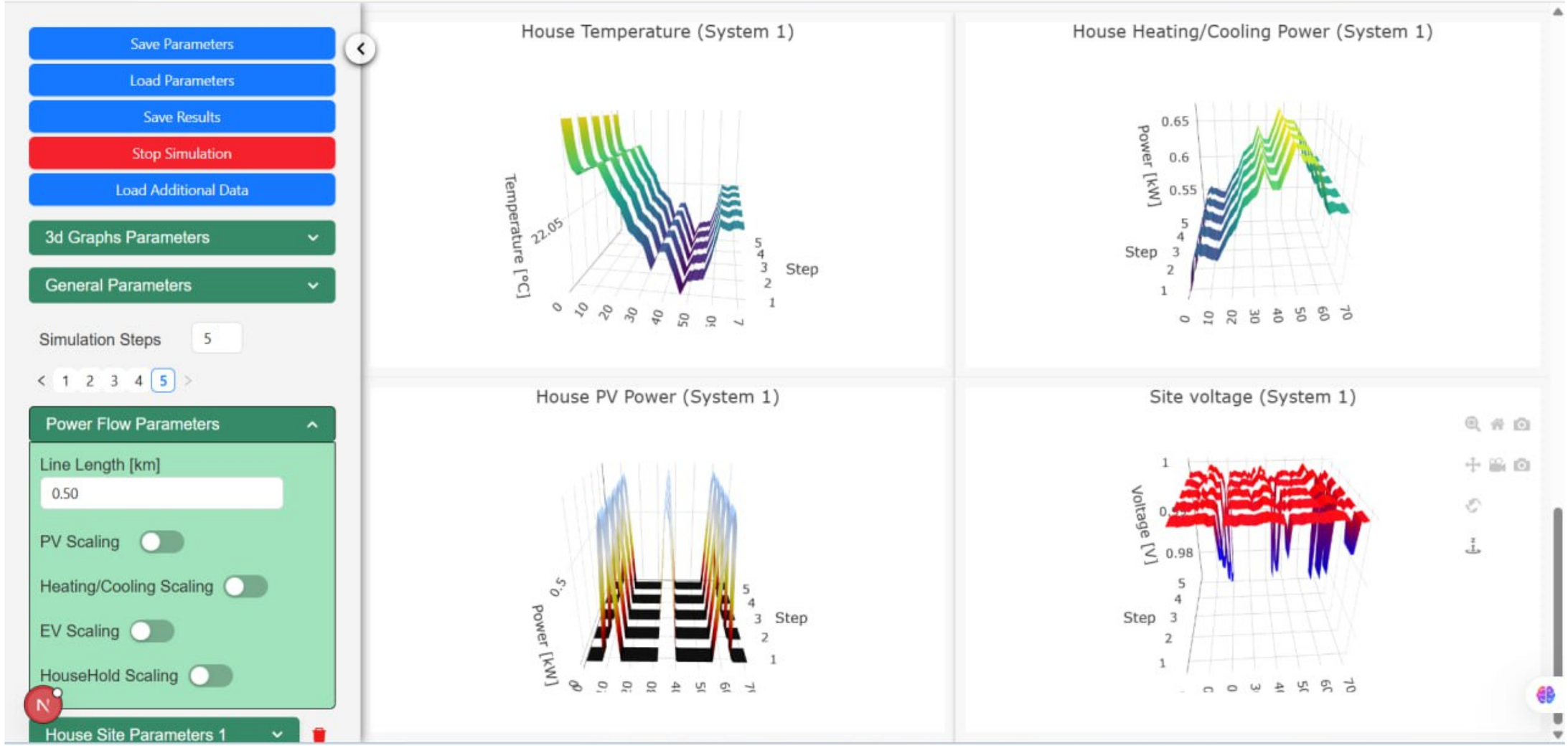
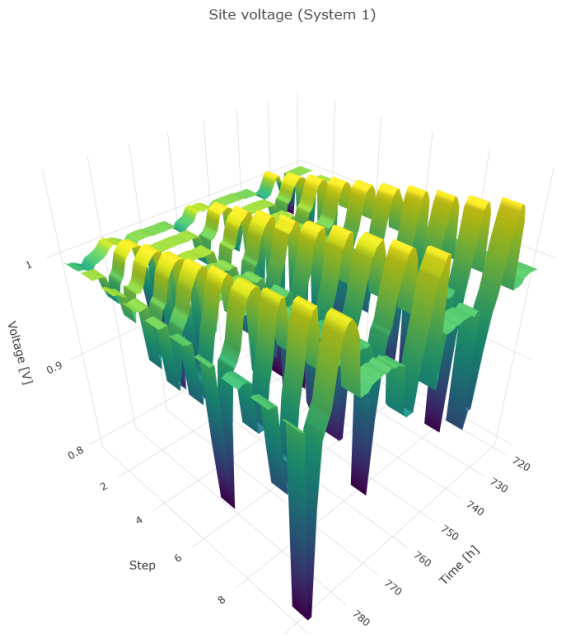
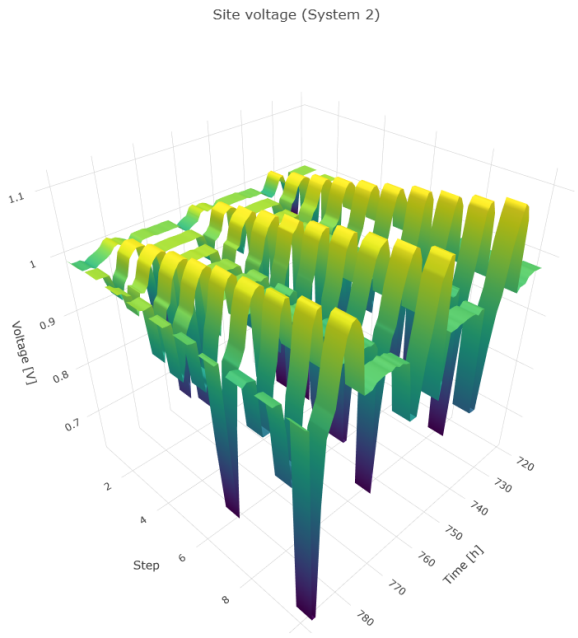
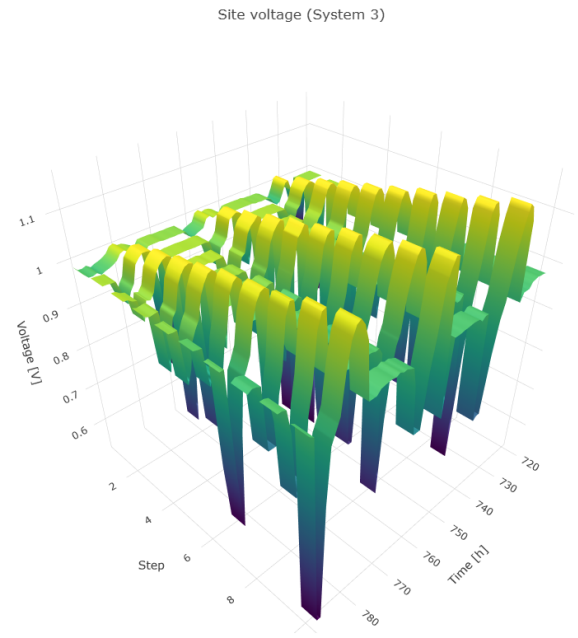
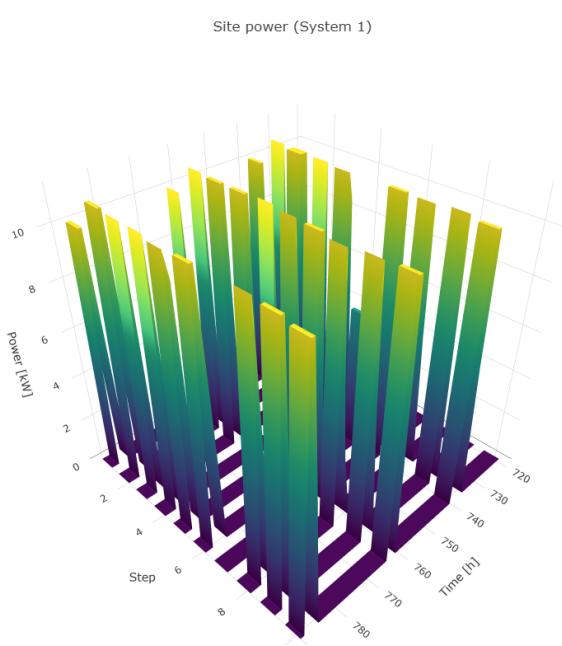


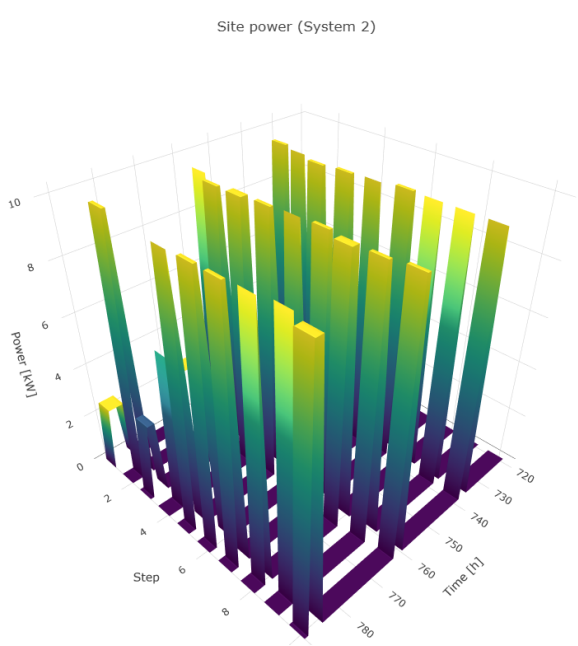
Figure B.4 – Graphical User Interface (GUI) of the developed cross-sector Simulation Framework

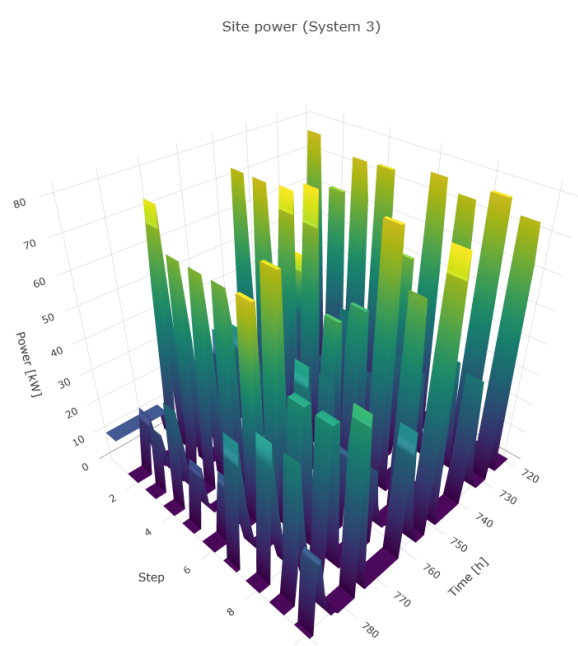


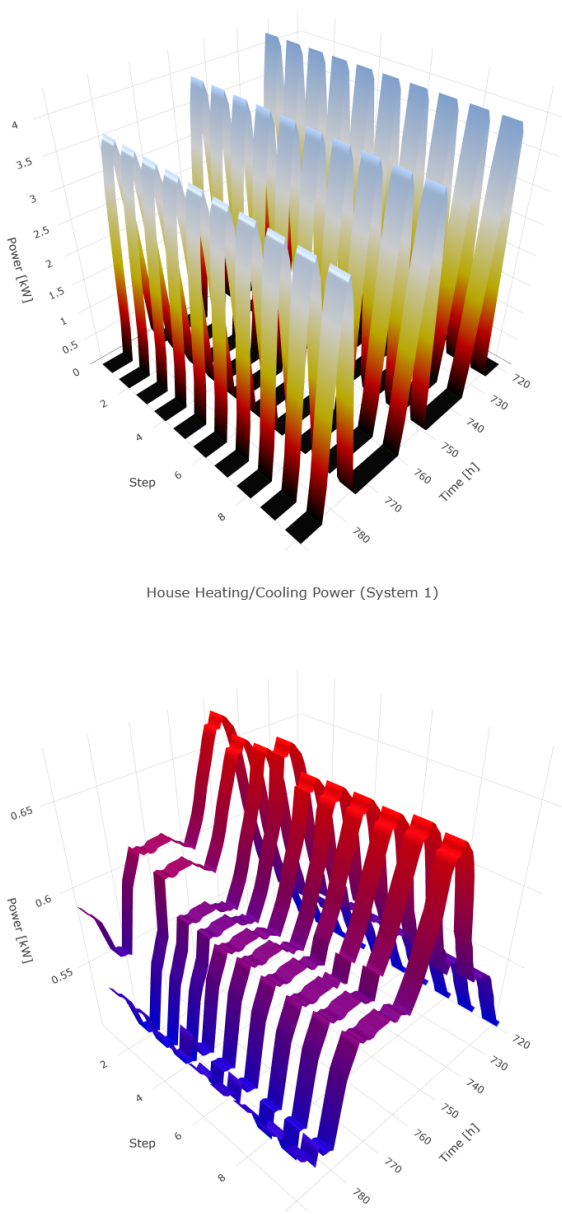


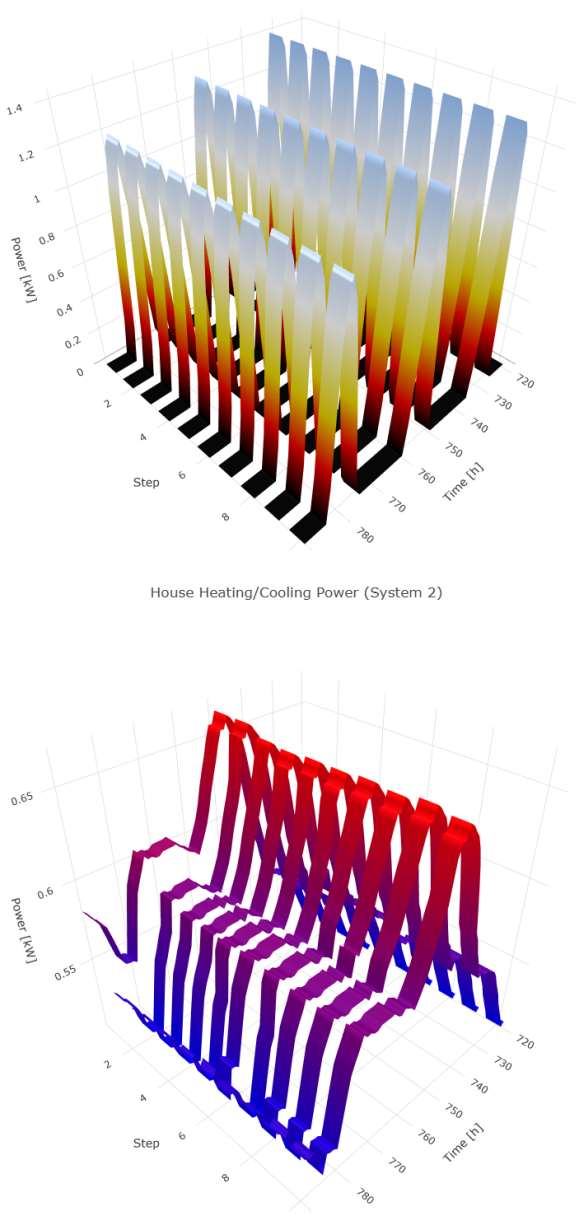












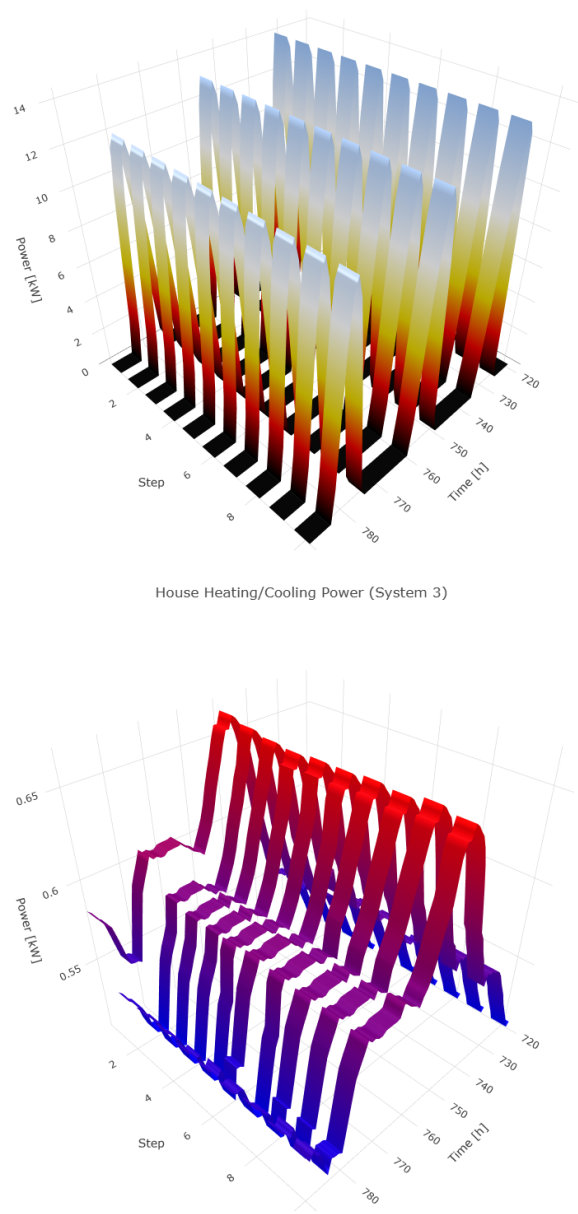
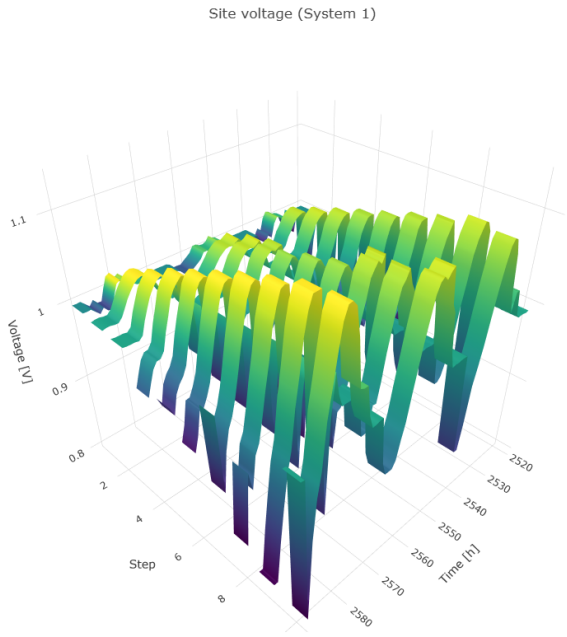
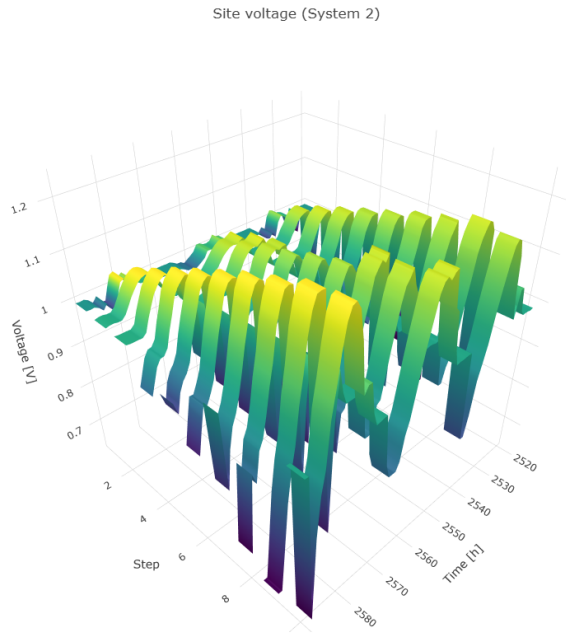
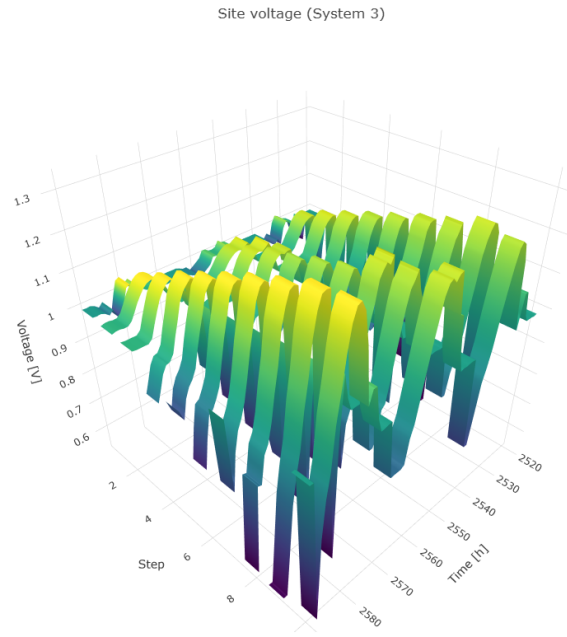
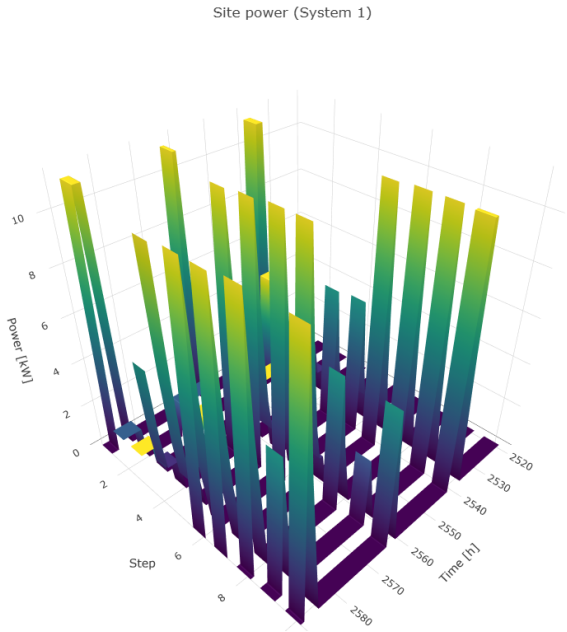


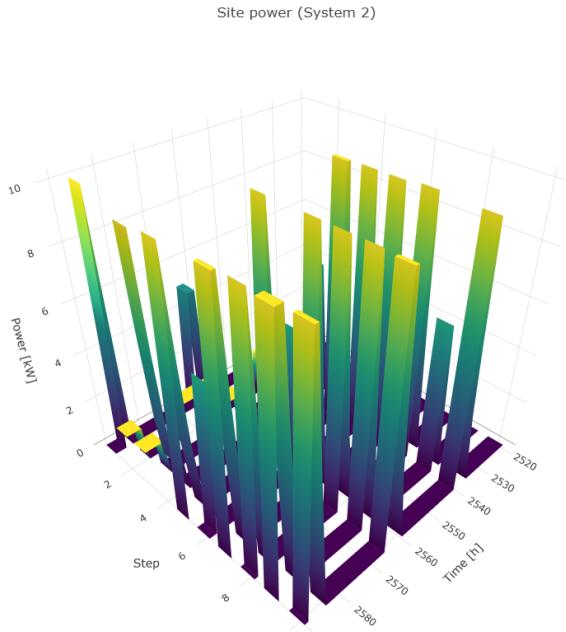
Figure B.5 – Three houses in winter (2009-01-30 to 2009-02-01)

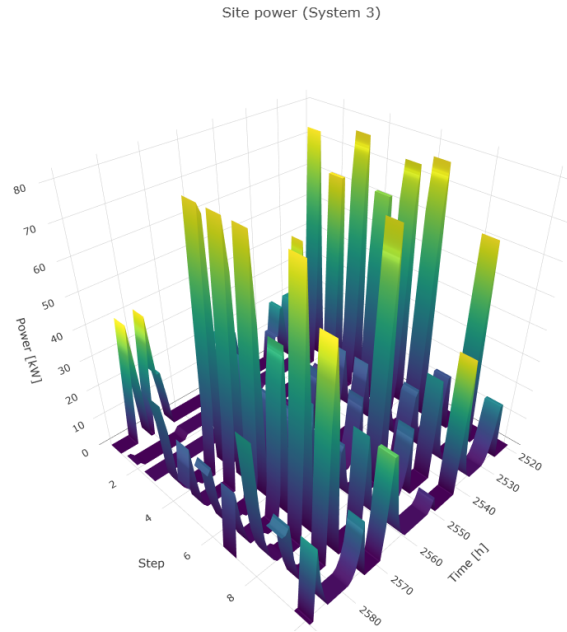


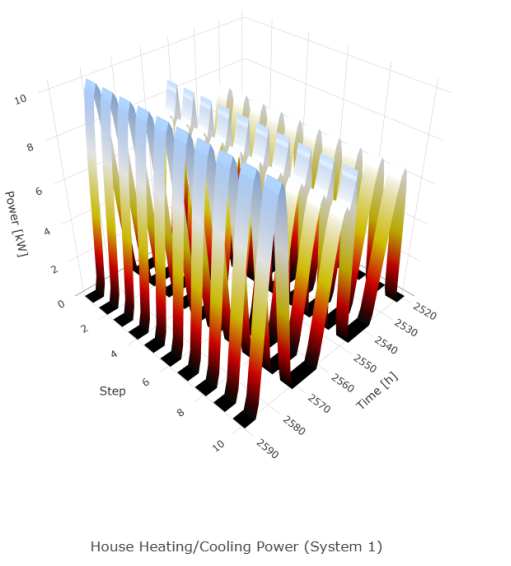


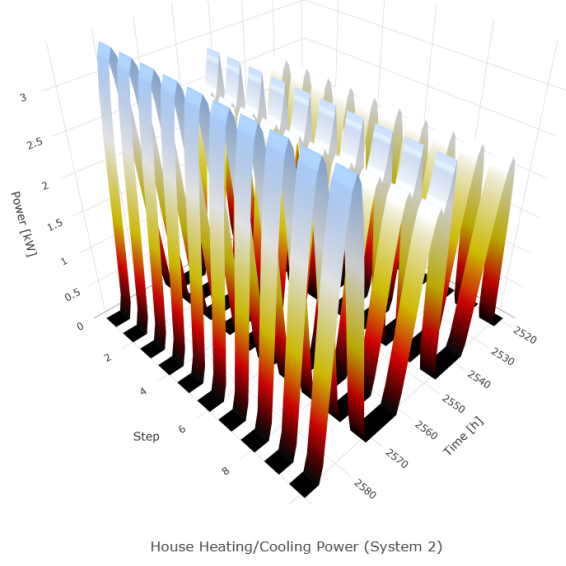


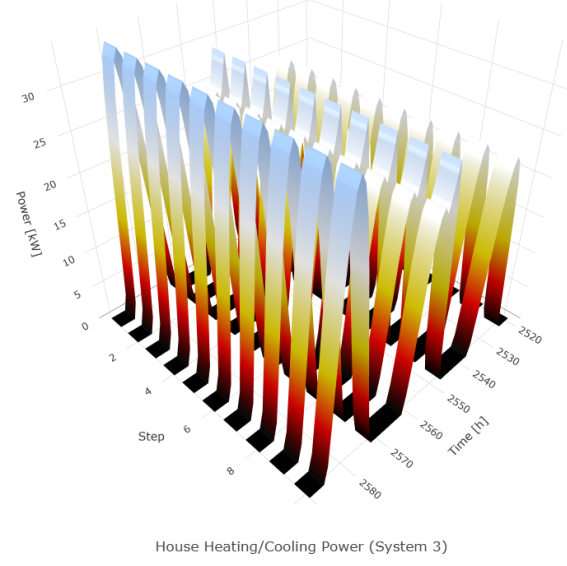


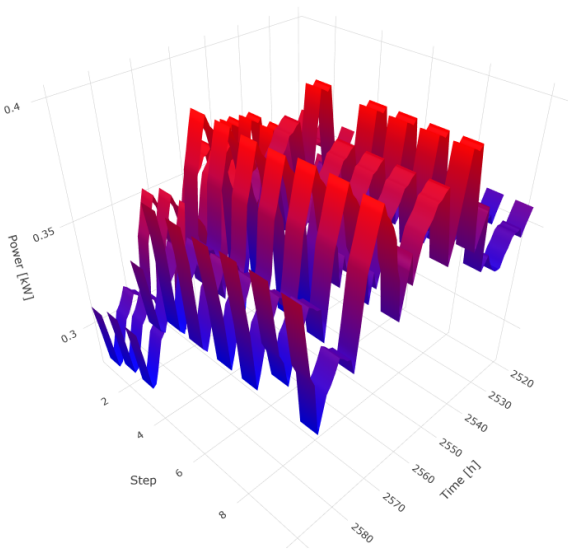


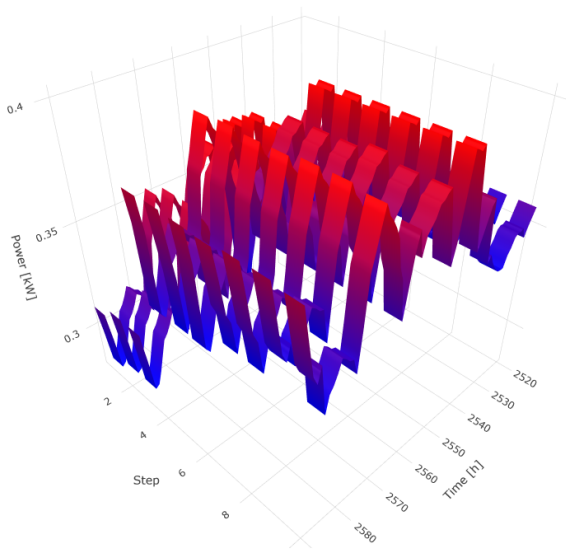












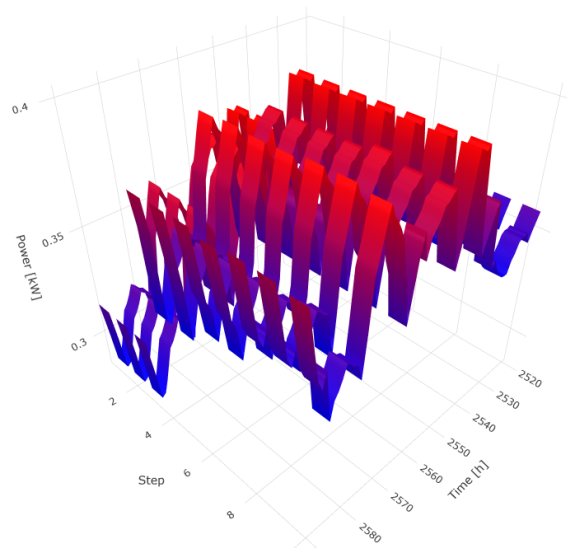
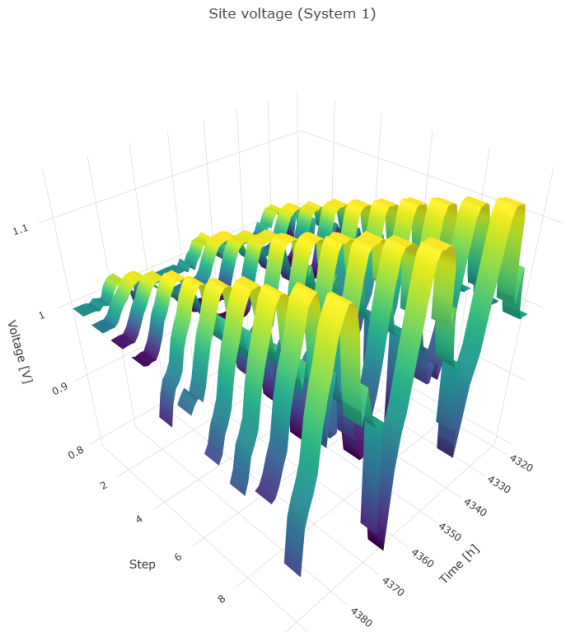
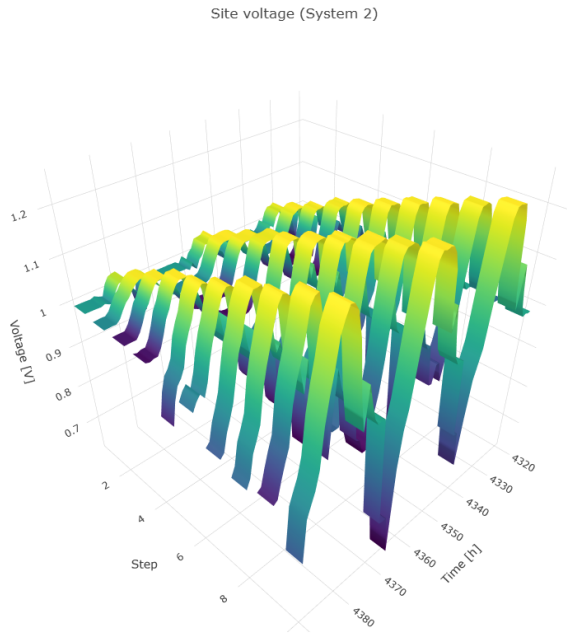
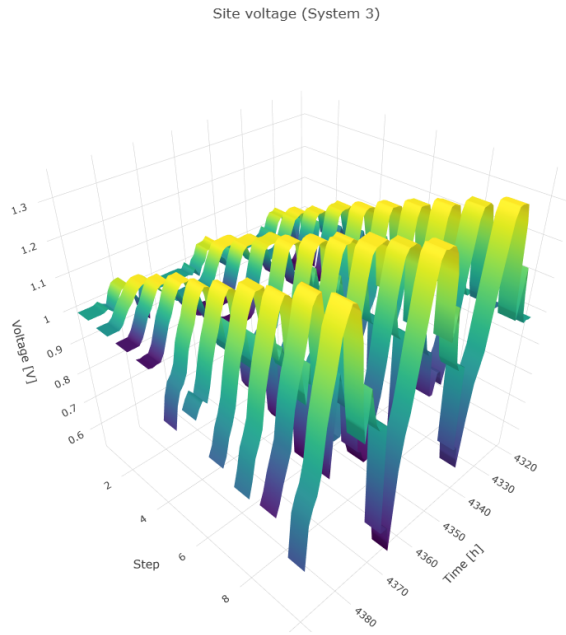
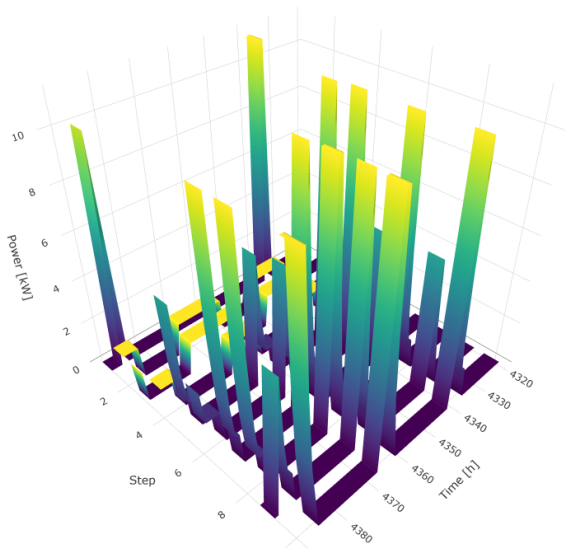


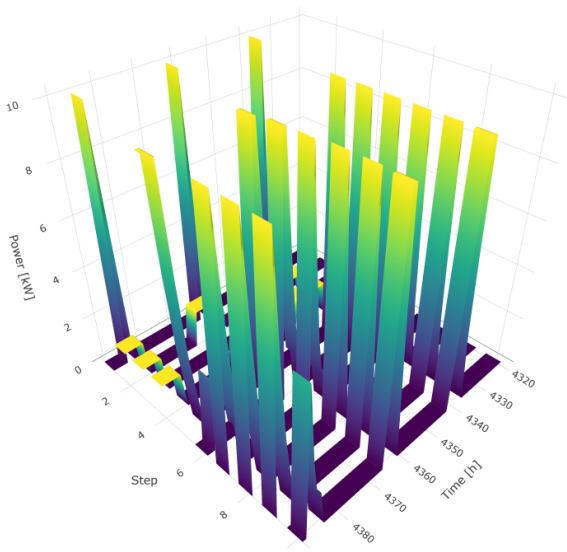
Figure B.6 – Three houses in spring (2009-04-15 to 2009-04-17)

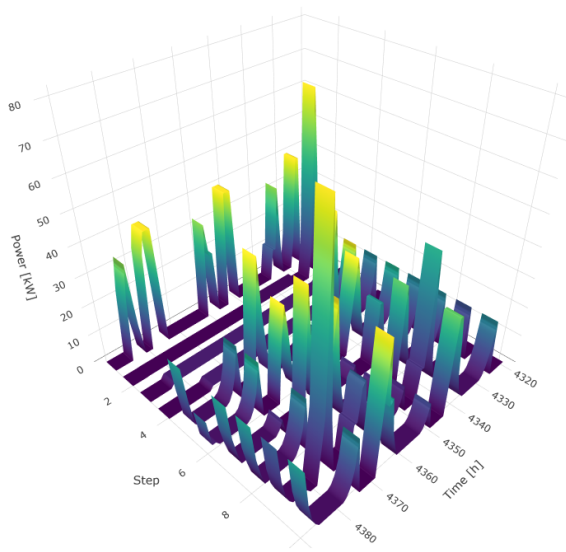


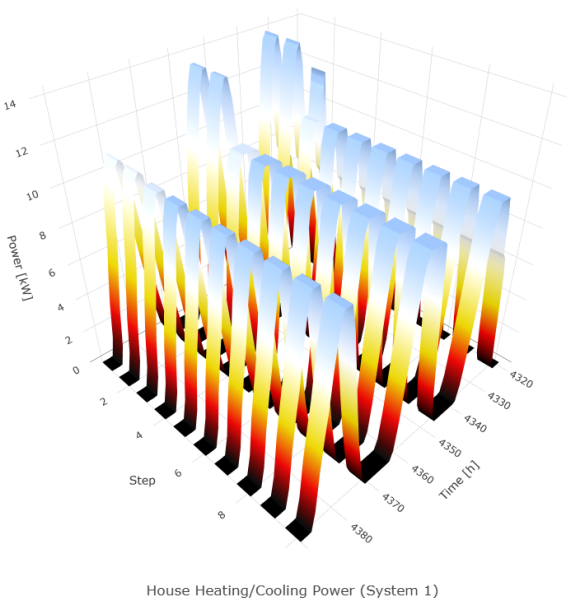


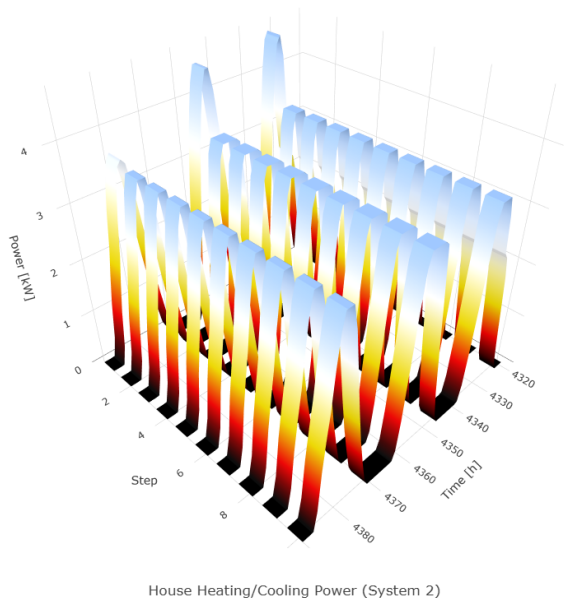


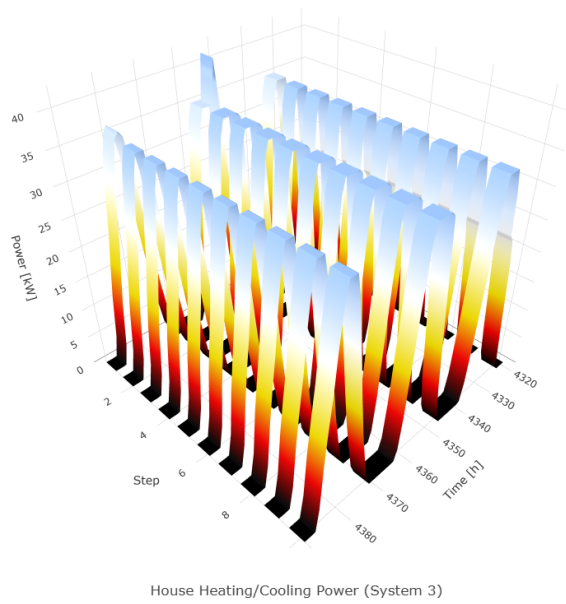


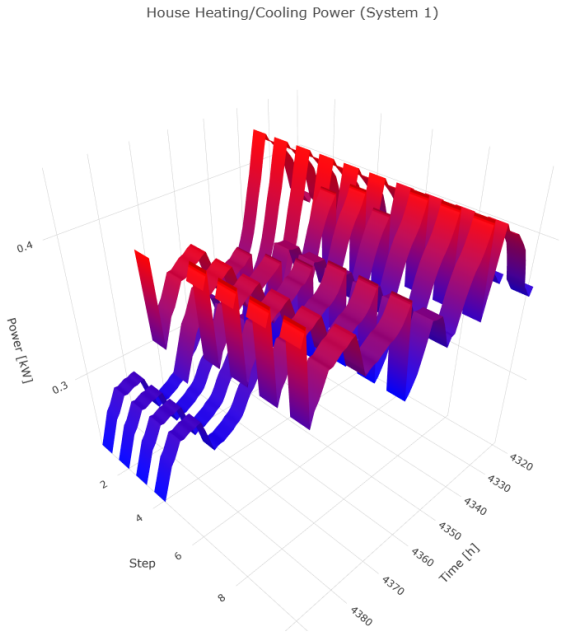


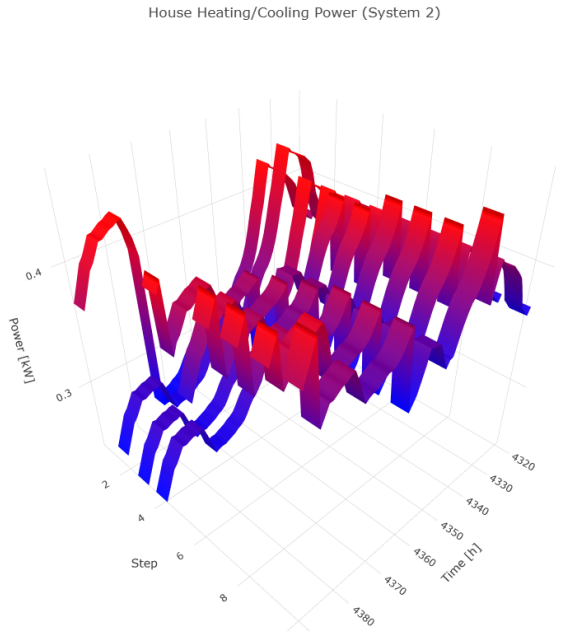












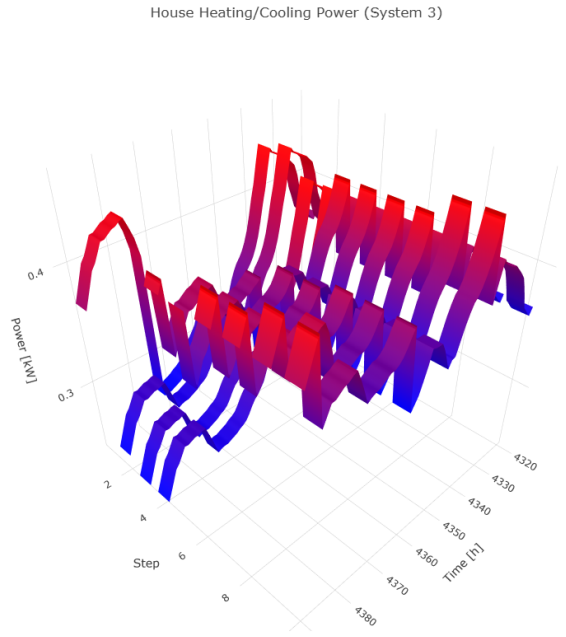


Figure B.7 – Three houses in summer (2009-06-29 to 2009-07-01)

Table B.1 – Assessment criteria for evaluating the commercial potential of any development and their score rating (on a scale of 0 to 4 points)

Evaluation Criteria and Scores (on a 5-point scale)					
Criterion\Score	0	1	2	3	4
Technical Feasibility of the Concept					
1	Credibility of the concept is not confirmed	Concept validated by expert opinions	Concept validated by calculations	Concept tested in practice	Operation verified in real-world conditions
Market Advantages/Disadvantages					
2	Numerous analogues on the market	Few analogues in a small market	Several analogues in a large market	Few analogues in a large market	No analogues on the market
3	Price is significantly higher than that of analogues	Price is slightly higher than that of analogues	Price is approximately equal to that of analogues	Price is slightly lower than that of analogues	Price is significantly lower than that of analogues
4	Properties are significantly worse than those of analogues	Properties are slightly worse than those of analogues	Properties are on par with those of analogues	Properties are slightly better than those of analogues	Properties are significantly better than those of analogues
Market Prospects					
5	OPEX are significantly higher than those of analogues	OPEX are slightly higher than those of analogues	OPEX are on par with those of analogues	OPEX are slightly lower than those of analogues	OPEX are significantly lower than those of analogues
6	Small market with no positive dynamics	Small market with positive dynamics	Medium-sized market with positive dynamics	Large and stable market	Large market with positive dynamics
7	Heavy competition by major companies	Strong competition by many companies	Moderate competition by few/small companies	Slight competition by few small companies	No competition (monopolist status)
Practical Feasibility					
8	Lack of experts for technical and commercial implementation of the concept	Requires hiring of experts or significant investment in training of existing staff	Minor training demand for existing staff plus slight expansion of existing team	Minor training demand for existing staff and no team expansion required	Experts are available in existing team for both technical and commercial implementation
9	Significant financial resources needed but funding sources available	Slight financial resources needed but no funding source available	Substantial financial resources needed and funding source exists	Slight financial resources needed and funding source exists	No need for additional funding
10	Requires very costly development of new materials	Hardly accessible and very expensive materials needed	Accessible but very expensive materials needed	Only easy accessible and affordable materials needed	Only cheap and ample available materiels used
11	Implementation $>$ 10 years and RoI $>$ 10 years	Implementation $<$ 10 years and RoI $>$ 10 years	Implementation $<$ 5 years and RoI $<$ 10 years	Implementation $<$ 3 years and RoI $<$ 5 years	Implementation $<$ 3 years and RoI $<$ 3 years
12	Requires novel regulatory documents and acquiring numerous permits for production and product implementation	Requires numerous permits for production and product implementation causing significant costs and delays	Obtaining permits for production and product implementation causes only minor costs and delays	Requires only notification to relevant authorities about production and product implementation	No regulatory constraints on production and product implementation exist

Appendix C
(compulsory)
Fragment of the listing

1 – JSON configuration file

ScenarioConfiguration20251127223313.json


```
{
  "general": {
    "startupSimulationTimeInDays": 200,
    "stepDurationInHours": 72,
    "confidenceIntervalLevel": 95,
    "clockTime": 1,
    "archiveData": true,
    "numberOfSimulationSteps": 3
  },
  "steps": [
    {
      "powerFlow": {
        "lineLength": 0.1,
        "enablePVScaling": false,
        "maxPVVoltage": 1.05,
        "minPVVoltage": 1.025,
        "scalePV": 0.7,
        "normalPV": 1,
        "enableHeatingCoolingScaling": false,
        "maxHeatingCoolingVoltage": 0.975,
        "minHeatingCoolingVoltage": 0.95,
        "heatingNormalTemperature": 23,
        "coolingNormalTemperature": 24,
        "heatingScaleTemperature": 21,
        "coolingScaleTemperature": 26,
        "enableEVScaling": false,
        "maxSiteVoltage": 1.025,
        "minSiteVoltage": 0.975,
        "scaleEVPower": 0.5,
        "enableHouseHoldScaling": false,
        "maxHouseHoldVoltage": 1,
        "minHouseHoldVoltage": 0.98,
        "houseHoldNormal": 1,
        "houseHoldScale": 0.8
      },
      "houseSites": [
        {
          "id": "cfcc21cd-3947-4a1d-a3f1-e660ce4a3194",
          "heatingEnabled": true,
          "pvEnabled": true,
          "chargingSiteEnabled": true,
          "householdEnabled": true,
          "powerLimitKW": 22,
          "batteryEnabled": false,
          "batteryCapacityKWh": 10,
          "batteryMaxPowerKW": 5,
          "houseParameters": {
            "maxHeatingPower": 5000,
            "wishedHeatingTemperature": 23,
            "wishedCoolingTemperature": 25,
            "maxPVPower": 15
          },
          "chargingSiteParameters": {
            "parkingSpace": 0,
            "maxSitePower": 22,
            "initSitePower": 5,
            "minSitePower": 1.5,
            "numberOfChargingPoints": 2,
            "maxPowerPerPoint": 11,
            "queueingType": "FIFO",
            "evTypes": [
              {
                "id": "9ea83898-815e-40cd-9f26-574a975c0318",
                "instances": 2,
                "batteryCapacity": 80,
                "maxChargingPower": 11,
```

```
                  "wishedStateOfCharge": 0.8,
                  "arrivalTime": {
                    "type": "Normal",
                    "mean": 18,
                    "range": 1
                  },
                  "parkingTime": {
                    "type": "Erlang",
                    "mean": 15,
                    "range": 0.5,
                    "level": 5
                  },
                  "chargingDemand": {
                    "type": "Beta",
                    "mean": 0.4,
                    "range": 0.3
                  }
                },
                {
                  "id": "c8c1974b-b447-453b-9b1c-ab04f9e28ed8",
                  "instances": 2,
                  "batteryCapacity": 80,
                  "maxChargingPower": 11,
                  "wishedStateOfCharge": 0.8,
                  "arrivalTime": {
                    "type": "Normal",
                    "mean": 8,
                    "range": 1
                  },
                  "parkingTime": {
                    "type": "Erlang",
                    "mean": 8,
                    "range": 1.5,
                    "level": 5
                  },
                  "chargingDemand": {
                    "type": "Beta",
                    "mean": 0.4,
                    "range": 0.3
                  }
                }
              ]
            }
          }
        ]
      },
      {
        "powerFlow": {
          "lineLength": 0.5,

          "... same as above ..."

        },
        "houseSites": [

        "... same as above ..."

        ]
      },
      {
        "powerFlow": {
          "lineLength": 1.0,

          "... same as above ..."

        },
        "houseSites": [

        "... same as above ..."

        ]
      }
    ]
}
```

House.mo

```
model House
  package MediumAir = AixLib.Media.Air "Medium model for air";
  package MediumWater = AixLib.Media.Water "Medium model for water";
  parameter Real kelvinKoef = 273.15;
  parameter Real initialWallTemperature = 293;

  parameter Integer startSimulation = 0;
  parameter Integer changeTime = 86400;
  parameter Integer previousHeatingT = 273 + 22;
  parameter Integer newHeatingT = 273 + 22;
  parameter Integer previousCoolingT = 273 + 23;
  parameter Integer newCoolingT = 273 + 23;
  // BUILDING Params
  parameter Modelica.Units.SI.Area AWall = 220 "Wall area";
  parameter Modelica.Units.SI.Length dWall = 0.25 "Wall thickness";
  parameter Modelica.Units.SI.ThermalConductivity kWall = 0.04 "Wall thermal
   ↪  conductivity";
  parameter Modelica.Units.SI.Density rhoWall = 2000 "Wall density";
  parameter Modelica.Units.SI.SpecificHeatCapacity cpWall = 1000 "Wall specific heat
   ↪  capacity";
  parameter Modelica.Units.SI.CoefficientOfHeatTransfer hWall = 2 "Convective heat
   ↪  transfer coefficient at the wall";
  parameter Modelica.Units.SI.Volume VZone = AWall*3 "Zone volume";
  // WINDOWS Params
  parameter Real eastAzimuth = 1;
  parameter Real southAzimuth = 0;
  parameter Real westAzimuth = 1;
  parameter Real northAzimuth = 0;
  parameter Modelica.Units.SI.Area AWinEast = 7.5 "East Window area";
  parameter Modelica.Units.SI.Area AWinSouth = 10 "South Window area";
  parameter Modelica.Units.SI.Area AWinWest = 7.5 "West Window area";
  parameter Modelica.Units.SI.Area AWinNorth = 5 "North Window area";
  parameter Real gWin(min = 0, max = 1, unit = "1") = 0.3 "Solar heat gain
   ↪  coefficient of window";
  // HEATING Params
  parameter Real wishedHeatingT = 22;
  parameter Real wishedCoolingT = 24;

  parameter Real heating_power = 2000;
  parameter Modelica.Units.SI.HeatFlowRate QHea_flow_nominal = heating_power "Nominal
   ↪  capacity of heating system";
  parameter Modelica.Units.SI.MassFlowRate mWat_flow_nominal =
   ↪  QHea_flow_nominal/10/4200 "Nominal mass flow rate for water loop";
  parameter Modelica.Units.SI.MassFlowRate mAir_flow_nominal = VZone*2*1.2/3600
   ↪  "Nominal mass flow rate for air loop";
  // COOLING Params
  parameter Real coolingSetT = 20;
  parameter Modelica.Units.SI.PressureDifference dpAir_nominal = 200 "Pressure drop
   ↪  at nominal mass flow rate for air loop";
  // WEATHER Connector
  AixLib.BoundaryConditions.WeatherData.ReaderTMY3 weaDat(filNam =
   ↪  "C:/Users/.../modelicaModels/UKR_Kiev.333450_IWEC.mos") annotation(
    Placement(transformation(origin = {-172, 28}, extent = {{-10, -10}, {10, 10}})));
  AixLib.BoundaryConditions.WeatherData.Bus weaBus annotation(
    Placement(transformation(origin = {-154, 32}, extent = {{-10, -10}, {10, 10}}),
     ↪  iconTransformation(origin = {172, -182}, extent = {{-10, -10}, {10, 10}})));
  Modelica.Thermal.HeatTransfer.Sources.PrescribedTemperature TOut annotation(
    Placement(transformation(origin = {-114, 32}, extent = {{-10, -10}, {10, 10}})));
  // HOUSE Elements
  Modelica.Thermal.HeatTransfer.Components.ThermalResistor wallResistor(R =
   ↪  dWall/AWall/kWall) annotation(
    Placement(transformation(origin = {-56, 32}, extent = {{-10, -10}, {10, 10}})));
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor wallCapacitor(C =
   ↪  AWall*dWall*cpWall*rhoWall, T(start = initialWallTemperature)) annotation(
    Placement(transformation(origin = {48, 32}, extent = {{-10, -10}, {10, 10}},
     ↪  rotation = -90)));
  // Windows parameters
  Modelica.Blocks.Math.Gain eastWindow(k = AWinEast*eastAzimuth*gWin) annotation(
    Placement(transformation(origin = {-56, -16}, extent = {{-6, -6}, {6, 6}})));
  Modelica.Blocks.Math.Gain southWin(k = AWinSouth*southAzimuth*gWin) annotation(
    Placement(transformation(origin = {-56, -38}, extent = {{-6, -6}, {6, 6}})));
  Modelica.Blocks.Math.Gain westWin(k = AWinWest*westAzimuth*gWin) annotation(
    Placement(transformation(origin = {-56, -56}, extent = {{-6, -6}, {6, 6}})));
  Modelica.Blocks.Math.Gain northWin(k = AWinNorth*northAzimuth*gWin) annotation(
```

```
  Placement(transformation(origin = {-56, -74}, extent = {{-6, -6}, {6, 6}})));
Modelica.Thermal.HeatTransfer.Sources.PrescribedHeatFlow winEast annotation(
  Placement(transformation(origin = {-12, -16}, extent = {{-8, -8}, {8, 8}})));
Modelica.Thermal.HeatTransfer.Sources.PrescribedHeatFlow winSouth annotation(
  Placement(transformation(origin = {-12, -38}, extent = {{-8, -8}, {8, 8}})));
Modelica.Thermal.HeatTransfer.Sources.PrescribedHeatFlow winWest annotation(
  Placement(transformation(origin = {-12, -56}, extent = {{-8, -8}, {8, 8}})));
Modelica.Thermal.HeatTransfer.Sources.PrescribedHeatFlow winNorth annotation(
  Placement(transformation(origin = {-12, -74}, extent = {{-8, -8}, {8, 8}})));
AixLib.Fluid.MixingVolumes.MixingVolume zon(redeclare package Medium = MediumAir,
↪   m_flow_nominal = mAir_flow_nominal, V = VZone, energyDynamics =
↪   Modelica.Fluid.Types.Dynamics.FixedInitial, nPorts = 2, massDynamics =
↪   Modelica.Fluid.Types.Dynamics.DynamicFreeInitial, T_start = 293.15) annotation(
  Placement(transformation(origin = {48, 72}, extent = {{-10, 10}, {10, -10}})));
Modelica.Thermal.HeatTransfer.Components.ThermalResistor conResistor(R =
↪   dWall/AWall/kWall) annotation(
  Placement(transformation(origin = {-18, 52}, extent = {{-10, -10}, {10, 10}},
  ↪   rotation = -90)));
// Heating
AixLib.Fluid.HeatExchangers.Radiators.RadiatorEN442_2 radiator(redeclare package
↪   Medium = AixLib.Media.Water "Water", T_a_nominal = kelvinKoef + 50, T_b_nominal
↪   = kelvinKoef + 40, energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
↪   allowFlowReversal = false, Q_flow_nominal = QHea_flow_nominal) annotation(
  Placement(transformation(origin = {24, -132}, extent = {{-10, -10}, {10, 10}})));
AixLib.Fluid.HeatExchangers.HeaterCooler_u heaWat(redeclare package Medium =
↪   AixLib.Media.Water "Water", m_flow_nominal = mWat_flow_nominal, energyDynamics
↪   = Modelica.Fluid.Types.Dynamics.SteadyState, allowFlowReversal = false,
↪   dp_nominal = 5000, Q_flow_nominal = QHea_flow_nominal) annotation(
  Placement(transformation(origin = {-18, -132}, extent = {{-10, -10}, {10,
  ↪   10}})));
AixLib.Fluid.Sources.Boundary_pT bouWat(redeclare package Medium =
↪   AixLib.Media.Water "Water", nPorts = 1) annotation(
  Placement(transformation(origin = {-80, -180}, extent = {{-10, -10}, {10,
  ↪   10}})));
Modelica.Thermal.HeatTransfer.Sensors.TemperatureSensor temperatureSensor
↪   annotation(
  Placement(transformation(origin = {21, 122}, extent = {{7, -7}, {-7, 7}})));
// Cooling
AixLib.Fluid.Sources.Boundary_pT bouAir(redeclare package Medium = AixLib.Media.Air
↪   "Moist air", use_T_in = true, nPorts = 2) annotation(
  Placement(transformation(origin = {-142, 156}, extent = {{-10, -10}, {10,
  ↪   10}})));
AixLib.Fluid.HeatExchangers.ConstantEffectiveness hexRec(redeclare package Medium1
↪   = AixLib.Media.Air "Moist air", redeclare package Medium2 = AixLib.Media.Air
↪   "Moist air", m1_flow_nominal = mAir_flow_nominal, m2_flow_nominal =
↪   mAir_flow_nominal, dp1_nominal = 0, dp2_nominal = 0, eps = 0.85) annotation(
  Placement(transformation(origin = {-108, 156}, extent = {{-10, -10}, {10,
  ↪   10}})));
AixLib.Fluid.Actuators.Dampers.Exponential vawDam(redeclare package Medium =
↪   AixLib.Media.Air "Moist air", m_flow_nominal = mAir_flow_nominal, from_dp =
↪   true, dpDamper_nominal = 10, dpFixed_nominal = dpAir_nominal - 10) annotation(
  Placement(transformation(origin = {-14, 150}, extent = {{-10, 10}, {10, -10}})));
AixLib.Fluid.HeatExchangers.SensibleCooler_T coolAir(redeclare package Medium =
↪   AixLib.Media.Air "Moist air", allowFlowReversal = false, m_flow_nominal =
↪   mAir_flow_nominal, dp_nominal = 0) annotation(
  Placement(transformation(origin = {-40, 150}, extent = {{-10, 10}, {10, -10}})));
AixLib.Fluid.Movers.FlowControlled_dp fan(redeclare package Medium =
↪   AixLib.Media.Air "Moist air", energyDynamics =
↪   Modelica.Fluid.Types.Dynamics.SteadyState,
↪   nominalValuesDefineDefaultPressureCurve = true, use_inputFilter = false,
↪   m_flow_nominal = mAir_flow_nominal, dp_nominal = dpAir_nominal) annotation(
  Placement(transformation(origin = {-82, 150}, extent = {{-10, 10}, {10, -10}})));
Modelica.Blocks.Sources.Constant con_dp(k = dpAir_nominal) annotation(
  Placement(transformation(origin = {-99, 131}, extent = {{-5, -5}, {5, 5}})));
Modelica.Blocks.Sources.Constant TSupAirCoo(k = kelvinKoef + 20) annotation(
  Placement(transformation(origin = {-67, 130}, extent = {{-5, -5}, {5, 5}})));
AixLib.Controls.Continuous.LimPID damperController(controllerType =
↪   Modelica.Blocks.Types.SimpleController.P, yMin = 0.25) annotation(
  Placement(transformation(origin = {-2, 122}, extent = {{6, -6}, {-6, 6}},
  ↪   rotation = -0)));
```

```
  AixLib.Fluid.Movers.FlowControlled_m_flow pump(redeclare package Medium =
      AixLib.Media.Water "Water", energyDynamics =
    ↪ Modelica.Fluid.Types.Dynamics.SteadyState, allowFlowReversal = false, inputType
    ↪ = AixLib.Fluid.Types.InputType.Continuous,
    ↪ nominalValuesDefineDefaultPressureCurve = true, use_inputFilter = false,
    ↪ m_flow_nominal = mWat_flow_nominal, massFlowRates = mWat_flow_nominal*{1})
    ↪ annotation(
      Placement(transformation(origin = {-16, -180}, extent = {{10, -10}, {-10,
      ↪ 10}})));
  AixLib.Electrical.PVSystem.PVSystem pVSystem(redeclare
      AixLib.DataBase.SolarElectric.QPlusBFRG41285 data, redeclare model
    ↪ IVCharacteristics = AixLib.Electrical.PVSystem.BaseClasses.
    ↪ IVCharacteristics5pAnalytical, redeclare model CellTemperature =
    ↪ AixLib.Electrical.PVSystem.BaseClasses. CellTemperatureMountingCloseToGround,
    ↪ n_mod = 21, til = 0, azi = 0, lat = 0.8805186076311393, lon =
    ↪ 0.5326744877086693, alt = 10, timZon = weaDat.timZon) annotation(
      Placement(transformation(origin = {-122, -52}, extent = {{-10, -10}, {10,
      ↪ 10}})));
  Modelica.Blocks.Math.Add heatingCoolingDemand(k1 = -1) annotation(
      Placement(transformation(origin = {95, 67}, extent = {{-7, -7}, {7, 7}})));
  Modelica.Blocks.Math.Abs absCooling annotation(
      Placement(transformation(origin = {95, 142}, extent = {{-7, -7}, {7, 7}})));
  Modelica.Blocks.Math.Gain coolingInKW(k = 0.001) annotation(
      Placement(transformation(origin = {95, 116}, extent = {{-7, -7}, {7, 7}})));
  Modelica.Blocks.Math.Add housePower(k2 = -1) annotation(
      Placement(transformation(origin = {93, 20}, extent = {{-7, -7}, {7, 7}})));
  Modelica.Thermal.HeatTransfer.Sensors.TemperatureSensor temperatureSensHeat
    ↪ annotation(
      Placement(transformation(origin = {-19, -112}, extent = {{7, -7}, {-7, 7}})));
  Modelica.Blocks.Sources.Constant constPump(k = mWat_flow_nominal) annotation(
      Placement(transformation(origin = {-62, -156}, extent = {{-6, -6}, {6, 6}})));
  Modelica.Blocks.Math.IntegerToReal integerToReal annotation(
      Placement(transformation(origin = {-109, -122}, extent = {{-7, -7}, {7, 7}})));
  Modelica.Blocks.Math.IntegerToReal integerToReal1 annotation(
      Placement(transformation(origin = {-30, 106}, extent = {{-8, -8}, {8, 8}})));

  Modelica.Blocks.Sources.IntegerTable heatingTable(table = [startSimulation,
    ↪ previousHeatingT; changeTime, newHeatingT])  annotation(
      Placement(transformation(origin = {-136, -122}, extent = {{-10, -10}, {10,
      ↪ 10}})));
  Modelica.Blocks.Sources.IntegerTable coolingTable(table = [startSimulation,
    ↪ previousCoolingT; changeTime, newCoolingT])  annotation(
      Placement(transformation(origin = {-58, 106}, extent = {{-8, -8}, {8, 8}})));
  AixLib.Controls.Continuous.LimPID damperController1(controllerType =
    ↪ Modelica.Blocks.Types.SimpleController.P, yMin = 0, yMax = 1) annotation(
      Placement(transformation(origin = {-53, -126}, extent = {{-6, 6}, {6, -6}})));
  Modelica.Blocks.Math.Add addHeating annotation(
      Placement(transformation(origin = {-80, -126}, extent = {{-6, -6}, {6, 6}})));
  Modelica.Blocks.Sources.Constant constKelvin(k = 0.35)  annotation(
      Placement(transformation(origin = {-108, -144}, extent = {{-10, -10}, {10,
      ↪ 10}})));
  Modelica.Blocks.Sources.Constant constKelvin1(k = -0.4) annotation(
      Placement(transformation(origin = {-30, 87}, extent = {{-6, -6}, {6, 6}})));
  Modelica.Blocks.Math.Add addHeating1 annotation(
      Placement(transformation(origin = {-9, 102}, extent = {{-6, -6}, {6, 6}})));
equation
  connect(weaDat.weaBus, weaBus) annotation(
      Line(points = {{-162, 28}, {-160, 28}, {-160, 32}, {-154, 32}}, color = {255,
      ↪ 204, 51}, thickness = 0.5));
  connect(TOut.T, weaBus.TDryBul) annotation(
      Line(points = {{-126, 32}, {-154, 32}}, color = {0, 0, 127}));
  connect(TOut.port, wallResistor.port_a) annotation(
      Line(points = {{-104, 32}, {-66, 32}}, color = {191, 0, 0}));
  connect(wallResistor.port_b, wallCapacitor.port) annotation(
      Line(points = {{-46, 32}, {38, 32}}, color = {191, 0, 0}));
  connect(weaBus.HGloHor, eastWindow.u) annotation(
      Line(points = {{-154, 32}, {-154, -16}, {-63, -16}}, color = {0, 0, 127}));
  connect(winEast.port, wallCapacitor.port) annotation(
      Line(points = {{-4, -16}, {10, -16}, {10, 32}, {38, 32}}, color = {191, 0, 0}));
  connect(eastWindow.y, winEast.Q_flow) annotation(
      Line(points = {{-49.4, -16}, {-20.8, -16}}, color = {0, 0, 127}));
  connect(southWin.y, winSouth.Q_flow) annotation(
      Line(points = {{-49.4, -38}, {-20.8, -38}}, color = {0, 0, 127}));
  connect(westWin.y, winWest.Q_flow) annotation(
      Line(points = {{-49.4, -56}, {-20.4, -56}}, color = {0, 0, 127}));
```

```
  connect(northWin.y, winNorth.Q_flow) annotation(
    Line(points = {{-49.4, -74}, {-20.4, -74}}, color = {0, 0, 127}));
  connect(winSouth.port, wallCapacitor.port) annotation(
    Line(points = {{-4, -38}, {10, -38}, {10, 32}, {38, 32}}, color = {191, 0, 0}));
  connect(winWest.port, wallCapacitor.port) annotation(
    Line(points = {{-4, -56}, {10, -56}, {10, 32}, {38, 32}}, color = {191, 0, 0}));
  connect(winNorth.port, wallCapacitor.port) annotation(
    Line(points = {{-4, -74}, {10, -74}, {10, 32}, {38, 32}}, color = {191, 0, 0}));
  connect(weaBus.HGloHor, southWin.u) annotation(
    Line(points = {{-154, 32}, {-154, -17}, {-72, -17}, {-72, -38}, {-63, -38}},
     ↪  color = {0, 0, 127}));
  connect(weaBus.HGloHor, westWin.u) annotation(
    Line(points = {{-154, 32}, {-154, -18}, {-74, -18}, {-74, -56}, {-63, -56}},
     ↪  color = {0, 0, 127}));
  connect(weaBus.HGloHor, northWin.u) annotation(
    Line(points = {{-154, 32}, {-154, -19}, {-76, -19}, {-76, -74}, {-63, -74}},
     ↪  color = {0, 0, 127}));
  connect(conResistor.port_b, wallCapacitor.port) annotation(
    Line(points = {{-18, 42}, {-18, 32}, {38, 32}}, color = {191, 0, 0}));
  connect(radiator.heatPortRad, wallCapacitor.port) annotation(
    Line(points = {{26, -125}, {26, 32.2}, {38, 32.2}}, color = {191, 0, 0}));
  connect(radiator.heatPortCon, zon.heatPort) annotation(
    Line(points = {{22, -125}, {22, 72.2}, {38, 72.2}}, color = {191, 0, 0}));
  connect(heaWat.port_b, radiator.port_a) annotation(
    Line(points = {{-8, -132}, {14, -132}}, color = {0, 127, 255}));
  connect(weaBus.TDryBul, bouAir.T_in) annotation(
    Line(points = {{-154, 32}, {-154, 160}}, color = {0, 0, 127}));
  connect(bouAir.ports[1], hexRec.port_a1) annotation(
    Line(points = {{-132, 156}, {-128, 156}, {-128, 162}, {-118, 162}}, color = {0,
     ↪  127, 255}));
  connect(bouAir.ports[2], hexRec.port_b2) annotation(
    Line(points = {{-132, 156}, {-128, 156}, {-128, 150}, {-118, 150}}, color = {0,
     ↪  127, 255}));
  connect(zon.ports[1], vawDam.port_b) annotation(
    Line(points = {{48, 82}, {48, 150}, {-4, 150}}, color = {0, 127, 255}));
  connect(zon.ports[2], hexRec.port_b1) annotation(
    Line(points = {{48, 82}, {48, 162}, {-98, 162}}, color = {0, 127, 255}));
  connect(coolAir.port_b, vawDam.port_a) annotation(
    Line(points = {{-30, 150}, {-24, 150}}, color = {0, 127, 255}));
  connect(fan.port_b, coolAir.port_a) annotation(
    Line(points = {{-72, 150}, {-50, 150}}, color = {0, 127, 255}));
  connect(hexRec.port_a2, fan.port_a) annotation(
    Line(points = {{-98, 150}, {-92, 150}}, color = {0, 127, 255}));
  connect(con_dp.y, fan.dp_in) annotation(
    Line(points = {{-93.5, 131}, {-82.9, 131}, {-82.9, 138}, {-82, 138}}, color = {0,
     ↪  0, 127}));
  connect(TSupAirCoo.y, coolAir.TSet) annotation(
    Line(points = {{-61.5, 130}, {-53.75, 130}, {-53.75, 142}, {-52, 142}}, color =
     ↪  {0, 0, 127}));
  connect(radiator.port_b, pump.port_a) annotation(
    Line(points = {{34, -132}, {40, -132}, {40, -180}, {-6, -180}}, color = {0, 127,
     ↪  255}));
  connect(heaWat.port_a, pump.port_b) annotation(
    Line(points = {{-28, -132}, {-38, -132}, {-38, -180}, {-26, -180}}, color = {0,
     ↪  127, 255}));
  connect(bouWat.ports[1], heaWat.port_a) annotation(
    Line(points = {{-70, -180}, {-38, -180}, {-38, -132}, {-28, -132}}, color = {0,
     ↪  127, 255}));
  connect(weaDat.weaBus, pVSystem.weaBus) annotation(
    Line(points = {{-162, 28}, {-162, -51}, {-134, -51}}, color = {255, 204, 51},
     ↪  thickness = 0.5));
  connect(coolAir.Q_flow, absCooling.u) annotation(
    Line(points = {{-29, 142}, {87, 142}}, color = {0, 0, 127}));
  connect(damperController.y, vawDam.y) annotation(
    Line(points = {{-9, 122}, {-14, 122}, {-14, 138}}, color = {0, 0, 127}));
  connect(temperatureSensor.T, damperController.u_s) annotation(
    Line(points = {{13, 122}, {5, 122}}, color = {0, 0, 127}));
  connect(zon.heatPort, temperatureSensHeat.port) annotation(
    Line(points = {{38, 72}, {18, 72}, {18, -112}, {-12, -112}}, color = {191, 0,
     ↪  0}));
  connect(zon.heatPort, temperatureSensor.port) annotation(
    Line(points = {{38, 72}, {32, 72}, {32, 122}, {28, 122}}, color = {191, 0, 0}));
  connect(heaWat.Q_flow, heatingCoolingDemand.u2) annotation(
    Line(points = {{-7, -126}, {10, -126}, {10, -118}, {78, -118}, {78, 63}, {87,
     ↪  63}}, color = {0, 0, 127}));
```

```
  connect(conResistor.port_a, zon.heatPort) annotation(
    Line(points = {{-18, 62}, {-18, 72}, {38, 72}}, color = {191, 0, 0}));
  connect(absCooling.y, coolingInKW.u) annotation(
    Line(points = {{103, 142}, {108.7, 142}, {108.7, 128}, {80.7, 128}, {80.7, 116},
      {87, 116}}, color = {0, 0, 127}));
  connect(coolAir.Q_flow, heatingCoolingDemand.u1) annotation(
    Line(points = {{-28, 142}, {78, 142}, {78, 71}, {87, 71}}, color = {0, 0, 127}));
  connect(heatingCoolingDemand.y, housePower.u1) annotation(
    Line(points = {{102.7, 67}, {107.7, 67}, {107.7, 42}, {81.7, 42}, {81.7, 24},
      {85, 24}}, color = {0, 0, 127}));
  connect(pVSystem.DCOutputPower, housePower.u2) annotation(
    Line(points = {{-110, -52}, {-90, -52}, {-90, -94}, {82, -94}, {82, 16}, {85,
      16}}, color = {0, 0, 127}));
  connect(constPump.y, pump.m_flow_in) annotation(
    Line(points = {{-55, -156}, {-16, -156}, {-16, -168}}, color = {0, 0, 127}));
  connect(heatingTable.y, integerToReal.u) annotation(
    Line(points = {{-125, -122}, {-117, -122}}, color = {255, 127, 0}));
  connect(coolingTable.y, integerToReal1.u) annotation(
    Line(points = {{-49, 106}, {-40, 106}}, color = {255, 127, 0}));
  connect(damperController1.y, heaWat.u) annotation(
    Line(points = {{-46, -126}, {-30, -126}}, color = {0, 0, 127}));
  connect(temperatureSensHeat.T, damperController1.u_m) annotation(
    Line(points = {{-26, -112}, {-53, -112}, {-53, -119}}, color = {0, 0, 127}));
  connect(integerToReal.y, addHeating.u1) annotation(
    Line(points = {{-101, -122}, {-87, -122}}, color = {0, 0, 127}));
  connect(addHeating.y, damperController1.u_s) annotation(
    Line(points = {{-74, -126}, {-60, -126}}, color = {0, 0, 127}));
  connect(constKelvin.y, addHeating.u2) annotation(
    Line(points = {{-97, -144}, {-94, -144}, {-94, -130}, {-88, -130}}, color = {0,
      0, 127}));
  connect(constKelvin1.y, addHeating1.u2) annotation(
    Line(points = {{-23, 87}, {-20, 87}, {-20, 98}, {-16, 98}}, color = {0, 0,
      127}));
  connect(integerToReal1.y, addHeating1.u1) annotation(
    Line(points = {{-22, 106}, {-16, 106}}, color = {0, 0, 127}));
  connect(addHeating1.y, damperController.u_m) annotation(
    Line(points = {{-2, 102}, {-2, 114}}, color = {0, 0, 127}));
  annotation(
    uses(AixLib(version = "2.1.1"), Modelica(version = "4.0.0")),
    Diagram(graphics = {Rectangle(origin = {-141, 35}, lineColor = {85, 170, 127},
        fillColor = {230, 230, 230}, fillPattern = FillPattern.Solid, lineThickness =
        0.75, borderPattern = BorderPattern.Raised, extent = {{-47, 29}, {47, -29}},
        radius = 4), Text(origin = {-125, 58}, extent = {{-23, 6}, {23, -6}},
        textString = "Weather inputs"), Rectangle(origin = {-11, -1}, lineColor = {0,
      41, 122}, fillColor = {239, 239, 239}, fillPattern = FillPattern.Solid,
      lineThickness = 0.75, borderPattern = BorderPattern.Raised, extent = {{-76,
      89}, {76, -89}}, radius = 4), Rectangle(origin = {-33, -37}, lineColor = {0,
      75, 113}, fillColor = {230, 230, 230}, fillPattern = FillPattern.Solid,
      lineThickness = 0.75, borderPattern = BorderPattern.Raised, extent = {{-48,
      49}, {48, -49}}, radius = 4), Rectangle(origin = {-42, -149}, lineColor =
      {255, 0, 127}, fillColor = {230, 230, 230}, fillPattern = FillPattern.Solid,
      lineThickness = 0.75, borderPattern = BorderPattern.Raised, extent = {{-108,
      50}, {108, -50}}, radius = 4), Text(origin = {-56, 80}, extent = {{-23, 6},
      {23, -6}}, textString = "House"), Text(origin = {-118, -186}, textColor =
      {255, 0, 127}, extent = {{-23, 6}, {23, -6}}, textString = "Heating"),
      Text(origin = {-34, 3}, extent = {{-32, 7}, {32, -7}}, textString = "Windows
      solar irradiation"), Rectangle(origin = {-62, 143}, lineColor = {0, 85, 255},
      fillColor = {230, 230, 230}, fillPattern = FillPattern.Solid, lineThickness =
      0.75, borderPattern = BorderPattern.Raised, extent = {{-128, 49}, {128,
      -49}}, radius = 4), Text(origin = {-166, 185}, textColor = {85, 0, 255},
      extent = {{-23, 6}, {23, -6}}, textString = "Cooling"), Rectangle(origin =
      {95, -4}, lineColor = {0, 85, 255}, fillColor = {230, 230, 230}, fillPattern
      = FillPattern.Solid, lineThickness = 0.75, borderPattern =
      BorderPattern.Raised, extent = {{-21, 195}, {21, -195}}, radius = 4),
      Text(origin = {97, 180}, extent = {{-23, 6}, {23, -6}}, textString =
      "Results")}, coordinateSystem(extent = {{-200, -200}, {200, 200}})),
    Icon(coordinateSystem(extent = {{-200, -200}, {200, 200}}), graphics = ... ),
    version = "",
    __OpenModelica_simulationFlags(lv = "LOG_STDOUT,LOG_ASSERT,LOG_STATS", s =
      "dassl", variableFilter = ".*"),
    experiment(StartTime = 0, StopTime = 3.1536e+07, Tolerance = 1e-06, Interval =
      902.55));
end House;
```

3 – Java code snippets

ChargingSite.java

```java
package chargingInfra;

import java.util.ArrayList;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import data_monitoring.monitors.SiteMonitor;
import exceptions.SitePowerExceededException;
import simulationParameters.SiteParameters;

/**
 * Represents a charging site with multiple charging points.
 * It manages the charging process, balances the power across charging points,
 * and monitors the charging site.
 */
public class ChargingSite {
    private static final Logger logger = LoggerFactory.getLogger(ChargingSite.class);

    private List<ChargingPoint> chargingPoints = new ArrayList<>();
    private List<ElectricVehicle> electricVehicles = new ArrayList<>();

    private final QueueingSystem queueingSystem;
    private final SiteParameters config;

    private double dynamicPowerLimit;

    public ChargingSite(SiteParameters config,
            QueueingSystem queueingSystem,
            List<ChargingPoint> chargingPoints,
            List<ElectricVehicle> electricVehicles) {

        this.config = config;
        this.queueingSystem = queueingSystem;

        this.chargingPoints = chargingPoints;
        for (var point : chargingPoints) {
            point.setChargingSite(this);
        }

        this.electricVehicles = electricVehicles;
        dynamicPowerLimit = config.getMaxSitePower();
    }

    public void processCharging(double deltaTime) {
        chargingPoints.stream()
            .forEach(cp -> cp.processCharging(deltaTime));
    }

    public void balanceChargingPower() {
        double power = getTotalPower();
        if (power > getDynamicPowerLimit()) {
            double scale = getDynamicPowerLimit() / power;
            power = chargingPoints.stream()
                    .mapToDouble(cp -> cp.scaleChargingPower(scale))
                    .sum();

            checkSitePower(power);
        }
    }

    public void scaleDynamicPowerLimit(double scale) {

        dynamicPowerLimit *= scale;
        if (dynamicPowerLimit > config.getMaxSitePower()) {
            dynamicPowerLimit = config.getMaxSitePower();
        }
        else if(dynamicPowerLimit < config.getMinSitePower()) {
            dynamicPowerLimit = config.getMinSitePower();
        }

        logger.info("Scaling site power {} by factor {}. Site power {}",
        ↪  this.hashCode(), scale, dynamicPowerLimit);
```

```
        balanceChargingPower();
    }

    public double getTotalPower() {
        double totalPower = 0.0;
        for (ChargingPoint cp : chargingPoints) {
            totalPower += cp.getPower();
        }
        return totalPower;
    }

    public void reset() {
        chargingPoints.stream().forEach(ChargingPoint::reset);
        electricVehicles.stream().forEach(ElectricVehicle::reset);
        queueingSystem.reset();
        dynamicPowerLimit = config.getMaxSitePower();
    }

    public List<ElectricVehicle> getElectricVehicles() {
        return electricVehicles;
    }

    public List<ChargingPoint> getChargingPoints() {
        return chargingPoints;
    }

    public QueueingSystem getQueueingSystem() {
        return queueingSystem;
    }

    private void checkSitePower(double totalPower) {
        if (totalPower - getDynamicPowerLimit() > 0.0001) {
            throw new SitePowerExceededException("Total power exceeds site
            ↪  capacity");
        }
    }

    public void attachMonitor(SiteMonitor monitor) {
        queueingSystem.attachMonitor(monitor);
    }

    public SiteParameters getParameters() {
        return config;
    }

    public double getDynamicPowerLimit() {
        return dynamicPowerLimit;
    }
}
```

QueueingSystem.java

```
package chargingInfra;

import data_monitoring.monitors.SiteMonitor;
import event_simulation.events.Event;
import simulation_control.SimulationControl;

/**
 * Represents a queueing system for managing clients and servers in a charging site.
 * It processes EV events: arrivals, departures, queuing, blocking, clock events.
 */
public class QueueingSystem {
    private LimitedClientsQueue clients;
    private ServersPool servers;

    private SimulationControl simControl;
    private SiteMonitor chargingSiteMonitor;

    public QueueingSystem(SimulationControl simControl,
            LimitedClientsQueue clients,
            ServersPool servers) {
        this.simControl = simControl;
        this.clients = clients;
        this.servers = servers;
    }
```

```
    public void processArrival(Event event) {
        chargingSiteMonitor.logArrival();

        simControl.scheduleNewArrival(event);

        var client = event.getClient();
        if (servers.connectToServer(client)) {
            client.setQueueingTime(event.getTime());
            client.setStartChargingTime(event.getTime());
            simControl.scheduleNewDeparture(event);
        } else {
            simControl.processNewQueueing(event);
        }
    }

    public void processQueuing(Event event) {
        chargingSiteMonitor.logQueueing();

        if (clients.isFull()) {
            simControl.processNewBlocking(event);
            return;
        }
        event.getClient().setQueueingTime(event.getTime());
        clients.addClient(event.getClient());
    }

    public void processDeparture(Event event) {
        chargingSiteMonitor.logDeparture();

        if (!event.getClient().isReadyToDeparture()) {
            simControl.scheduleNewDeparture(event);
            return;
        }
        event.getClient().saveOnDepartureParameters(event.getTime());
        event.getClient().getElectricVehicle().prepareForNextDay();

        if (clients.isEmpty()) {
            servers.releaseServer(event.getClient());
            return;
        }
        var client = clients.pullClient();
        client.setStartChargingTime(event.getTime());
        servers.reconnectServer(event.getClient(), client);
        event.setClient(client);
        simControl.scheduleNewDeparture(event);
    }

    public void processBlocking(Event event) {
        chargingSiteMonitor.logBlocking();
    }

    public void attachMonitor(SiteMonitor monitor) {
        this.chargingSiteMonitor = monitor;
    }

    public void reset() {
        clients.clear();
        servers.reset();
    }
}
```

ServersPool.java

```
package chargingInfra;

import java.util.ArrayList;
import java.util.List;

/** * Represents a pool of servers managing multiple charging points.
 * It allows connecting clients to available servers, releasing servers,
 * and checking server availability.
 */
public class ServersPool {
    private List<Server> availableServers;
    private List<Server> allServers;
```

```
    public ServersPool(List<ChargingPoint> chargingPoints) {
        availableServers = new ArrayList<>();
        allServers = new ArrayList<>();
        for (ChargingPoint chargingPoint : chargingPoints) {
            var server = new Server(chargingPoint);
            availableServers.add(server);
            allServers.add(server);
        }
    }

    public boolean connectToServer(Client client) {
        if (availableServers.isEmpty()) {
            return false;
        }
        var server = availableServers.remove(0);
        server.connectClient(client);
        return true;
    }

    public void releaseServer(Client client) {
        Server connectedServer = client.getConnectedServer();
        connectedServer.disconnectClient();
        availableServers.add(connectedServer);
    }

    public void reconnectServer(Client client, Client newClient) {
        client.getConnectedServer().connectClient(newClient);
    }

    public boolean isServerAvailable() {
        return !availableServers.isEmpty();
    }

    public void reset() {
        availableServers.clear();
        for (Server server : allServers) {
            server.disconnectClient();
            availableServers.add(server);
        }
    }
}
```

SimulationMonitor.java

```
package data_monitoring.monitors;

import java.util.ArrayList;
import java.util.List;

import data_monitoring.statistics.Parameter;
import data_monitoring.statistics.StatisticRecordCollection;
import data_monitoring.statistics.TimeRecordCollection;
import extendedModelling.powerFlow.PowerFlow;
import simulation_control.HouseSiteControl;

/**
 * ChargingMonitor is responsible for monitoring the charging infrastructure.
 * It manages site monitors, point monitors, and vehicle monitors.
 * It provides methods to log events, save step data, and retrieve statistics.
 */
public class SimulationMonitor {

    private List<HouseSiteMonitor> houseSiteMonitors;

    // its the list of monitors that is used for reporting results
    // it can contains all monitors: site, point, vehicle, power flow, house
    private List<SystemMonitor> allMonitors;
    private List<Parameter> monitoredParameters;

    public SimulationMonitor(
            List<HouseSiteControl> houseSiteControls, PowerFlow powerFlow,
            List<Parameter> monitoredParameters) {

        this.houseSiteMonitors = new ArrayList<>();
        this.allMonitors = new ArrayList<>();
        this.monitoredParameters = monitoredParameters;
```

```
        for (int siteId = 0; siteId < houseSiteControls.size(); siteId++) {
            HouseSiteControl houseSiteControl = houseSiteControls.get(siteId);
            var monitor = new HouseSiteMonitor(houseSiteControl, siteId, powerFlow,
             ↪  monitoredParameters);
            houseSiteMonitors.add(monitor);
            allMonitors.addAll(monitor.getAllMonitors());
        }
    }

    public void reconnect(List<HouseSiteControl> houseSiteControls, PowerFlow
     ↪  powerFlow) {
        for (int houseId = 0; houseId < houseSiteControls.size(); houseId++) {
            if(houseId >= houseSiteMonitors.size()) {
                HouseSiteControl houseSiteControl = houseSiteControls.get(houseId);
                var monitor = new HouseSiteMonitor(houseSiteControl, houseId,
                 ↪  powerFlow, monitoredParameters);
                houseSiteMonitors.add(monitor);
                allMonitors.addAll(monitor.getAllMonitors());
            } else {
                HouseSiteControl houseSiteControl = houseSiteControls.get(houseId);
                houseSiteMonitors.get(houseId).reconnect( houseSiteControl,
                 ↪  powerFlow);
            }
        }
    }

    public void logOnClockEvent(double time) {
        for(HouseSiteMonitor houseSiteMonitor : houseSiteMonitors){
            houseSiteMonitor.logOnClockEvent(time);
        }
    }

    public void saveStepData(Boolean archiveData) {
        for (SystemMonitor monitor : allMonitors) {
            monitor.saveStepData(archiveData);
        }
    }

    public List<StatisticRecordCollection> getStatistics(int confidenceLevel) {
        List<StatisticRecordCollection> collections = new ArrayList<>();
        for (HouseSiteMonitor houseSiteMonitor : houseSiteMonitors) {
            collections.addAll( houseSiteMonitor.getStatistics( confidenceLevel));
        }
        return collections;
    }

    public List<TimeRecordCollection> getTimeRecords() {
        List<TimeRecordCollection> collections = new ArrayList<>();
        for (SystemMonitor monitor : allMonitors) {
            collections.add(monitor.getTimeRecords());
        }
        return collections;
    }

    public List<List<TimeRecordCollection>> getArchivedTimeRecords() {
        List<List<TimeRecordCollection>> collections = new ArrayList<>();
        for (SystemMonitor monitor : allMonitors) {
            // List of TimeRecordCollection for one monitor for all steps
            List<TimeRecordCollection> timeCollections =
             ↪  monitor.getArchivedTimeRecords();
            if(collections.size() < timeCollections.size()) {
                // extend collections to have enough lists
                for(int i = collections.size(); i < timeCollections.size(); i++) {
                    collections.add(new ArrayList<>());
                }
            }
            for (int i = 0; i < timeCollections.size(); i++) {
                collections.get(i).add(timeCollections.get(i));
            }
        }
        return collections;
    }
}
```

PandaSample.java

```java
package data_monitoring.statistics;

import java.util.ArrayList;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

/**
 * PandaSample is a class that collects time series data for various parameters.
 * It allows adding values, saving step data.
 */
public class PandaSample {

    private List<List<TimeValueEntry>> data = new ArrayList<>();
    private List<TimeValueEntry> currentStepData = new ArrayList<>();

    public void addValue(double timestamp, double value) {
        currentStepData.add(new TimeValueEntry(value, timestamp));
    }

    public List<TimeValueEntry> getCurrentData() {
        return currentStepData;
    }

    public List<List<TimeValueEntry>> getArchivedData() {
        return data;
    }

    @JsonIgnore
    public List<Double> getValueList() {
        return getValueList(currentStepData);
    }

    public void saveStepData(Parameter parameter, Boolean archiveData) {
        if (!currentStepData.isEmpty()) {
            if(archiveData) {
                data.add(new ArrayList<>(currentStepData));
            }
            currentStepData.clear();
        }
    }

    public StatisticsRecord getCurrentStepStatistics(int confidenceLevel) {
        return new StatisticsRecord(getValueList(currentStepData), confidenceLevel);
    }

    public StatisticsRecord getStepStatistics(int stepIndex, int confidenceLevel) {
        return new StatisticsRecord(getValueList(data.get(stepIndex)),
        ↪  confidenceLevel);
    }

    public StatisticsRecord getTimeStatistics(double time, int confidenceLevel) {
        List<Double> valuesAtTime = new ArrayList<>();
        for (List<TimeValueEntry> step : data) {
            for (TimeValueEntry entry : step) {
                if (entry.getTimestamp() == time) {
                    valuesAtTime.add(entry.getValue());
                }
            }
        }
        return new StatisticsRecord(valuesAtTime, confidenceLevel);
    }

    private List<Double> getValueList(List<TimeValueEntry> entries) {
        return entries.stream()
                .map(TimeValueEntry::getValue)
                .toList();
    }
}
```

Appendix D

University for
Continuing
Education Krems
Department for
Integrated Sensor Systems (DISS)

Letter of intent to use

Issued to the student of group 1AKITP-24м of Vinnytsia National Technical University, Mariia S. Forkaliuk, in that the simulation tool developed by her in the process of performing her master's qualification work on the topic "INTEGRATION OF MODELLING TOOLS FROM ELECTRICITY, MOBILITY AND HEATING SECTORS INTO AN EVENT DRIVEN ENERGY FLOW MONITORING FRAMEWORK", namely the simulation framework developed in the course of her ERASMUS+ student mobility for the Department for Integrated Sensor Systems of the University for Continuing Education Krems, is planned to be used in the national funded research project "ErneuerbaresWald4tel" (FO999922001) as well as in future R&D projects.

License provisions: All results of the work performed can be implemented in the educational process, scientific research, scientific and methodological support and IT infrastructure of the Department for Integrated Sensor Systems at the University for Continuing Education Krems on a free basis without restrictions on the rights to use, copy, edit, supplement, publication and distribution.

Best Regards

Albert Treytl

DI Albert Treytl
Deputy of Department

Department für Integrierte Sensorsysteme
DONAU-UNIVERSITÄT KREMS
Viktor Kaplan Straße 2 E, A-2700 Wiener Neustadt
Tel: +43 (0)2622 23420, Fax: +43 (0)2622 23420-99
www.donau-uni.ac.at/diss

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інтеграція інструментів моделювання з секторів електроенергетики, мобільності та опалення в систему моніторингу потоків енергії, керовану подіями»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 5,35 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- (•) Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

- ( ) У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- ( ) У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Овчинников К.В., доц. каф. АІІТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Маслій Р.В.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Бісікало О.В., проф. каф. АІІТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Форкалюк М.С.
(підпис) (прізвище, ініціали)