Thèse n° 7380

On Smart-Buildings and their Integration into the Smart-Grid

Présentée le 17 janvier 2020

à la Faculté des sciences et techniques de l'ingénieur
Groupe Kayal
Programme doctoral en microsystèmes et microélectronique

pour l'obtention du grade de Docteur ès Sciences

par

Olivier Valentin Henri VAN CUTSEM

Acceptée sur proposition du jury

Dr J.-M. Sallese, président du jury
Prof. M. Kayal, directeur de thèse
Dr M. Pritoni, rapporteur
Prof. B. Cornélusse, rapporteur
Dr R. Cherkaoui, rapporteur

École
polytechnique
fédérale
de Lausanne

2020

Abstract

Today's electrical grid is undergoing deep changes, resulting from the large integration of distributed Renewable Energy Sources (RES) in an effort to decarbonize the generation of electrical energy. In addition to the emergence of this volatile electricity production, the worldwide demand for electricity increases due to a growing population and the intensified electrification of buildings. Smart-buildings represent promising assets for supporting the electrical grid in balancing demand with a supply based on non-dispatchable RES. A smart-building denotes a building equipped with sensor/actuator hardware connected to a federating Building Data Management System (BDMS) which enables high-level applications and services.

Tapping into the flexibility inherent to its various entities (load, storage, and generation), a smart-building can provide Demand Response (DR) functionality through the optimization of its energy profile in response to varying electricity prices or commands from the grid. This PhD thesis provides a set of tools, algorithms, and frameworks, revolving around the notion of smart-buildings that foster an enhanced Building-to-Grid (BtG) integration. The tools developed here aim to fill the gap encountered in the literature created by the recent rollout of BDMSs and the ubiquitous Internet of Things (IoT). Furthermore, the mismatch between current DR and the future RES-based smart-grid opens the way to the development of innovative algorithms and frameworks to manage the flexibility offered by smart-buildings for grid-side agents.

Built upon BDMSs, two open-source tools have been developed. Firstly, an integrated high-speed emulation and simulation software, dubbed Virtualization Engine (vEngine), allows the simulation of non-existing components of a building directly on-site. The multi-threaded, light architecture of vEngine permits efficient simulations, in a modular environment conceived for developers. Secondly, we describe Open Energy Management System (OpenEMS), a platform that seamlessly connects to any existing BDMS and provides its users with an environment to create their own energy management algorithms, with a focus on Model Predictive Control (MPC). Simulations using a realistic Swiss residential building model demonstrate the effectiveness and modularity of both tools. Additionally, we propose a multi-state load profile identification algorithm tailored to Non-Intrusive Load Monitoring (NILM). Applied to energy disaggregation, it shows promising results for enhanced energy feedbacks to the occupants.

To attain daily energy balance within the smart-grid, we propose several algorithms and energy management frameworks, using smart-buildings. An incremental MPC formulation is derived to better balance monthly costs associated to energy and peak demand of large commercial buildings. Simulations data show substantial benefits, for both the building's owner and the grid. Furthermore, we present a decentralized framework for autonomously managing the energy in a community of smart-buildings, with RES. Based on blockchain technology and smart-contracts, the framework optimizes an objective common to the whole community without the need for a central agent. A group of Swiss residential buildings are simulated; the results prove that the cooperative behavior enabled by the framework reduces the peak demand of the community which better uses local resources, compared to individual optimizations. Finally, we suggest a unified BtG model which could benefit grid-side aggregators in both microgrids and electricity markets. Leveraging state-of-the-art models, a second-order battery equation has been used to encapsulate the thermal behavior of buildings, along with data structures that represent deferrable loads and electric vehicles.

Keywords: smart-building, energy management system, model predictive control, integrated simulator, energy disaggregation, smart-grid, demand response, blockchain-based algorithm, smart-community, building-to-grid

Résumé

Le réseau électrique subit actuellement de profonds changements, principalement dus à l'intégration distribuée de Sources d'Energie Renouvelable (RES) visant à décarboniser la génération d'énergie électrique. En plus de l'émergence de cette production volatile, la demande mondiale en électricité augmente progressivement avec une population grandissante et une intensification de l'électrification des bâtiments. Les bâtiments intelligents présentent un potentiel significatif pour assister le réseau électrique à équilibrer la demande avec une offre basée sur des RES non-contrôlables. Un bâtiment intelligent se réfère à un bâtiment équipé de capteurs et actuateurs, fédérés par un Système de Gestion de Données du Bâtiment (BDMS) qui offre un service à des applications tierces.

En exploitant la flexibilité inhérente à ses différentes entités (charges, stockage et production), un bâtiment intelligent peut offrir une capacité de Réponse à la Demande (DR) via l'optimisation de son profil d'énergie en réponse à des prix d'électricité variables ou des commandes venant du réseau intelligent. Cette thèse de doctorat fournit un ensemble d'outils, d'algorithmes et de structures gravitant autour de la notion de bâtiment intelligent dans un but d'améliorer l'intégration bâtiment-vers-réseau (BtG). D'une part, les outils développés visent à combler un manque constaté dans la littérature, créé récemment par le développement des BDMSs et de l'Internet des Choses (IoT). D'autre part, l'incompatibilité entre le DR actuel et le future réseau intelligent riche en RES a ouvert la voie vers la conception d'algorithmes et structures innovants afin de gérer la flexibilité que peuvent offrir les bâtiments intelligent aux agent du réseau électrique.

Construits sur base des BDMSs, deux outils open-source ont été développés. Premièrement, un logiciel haute-vitesse intégré de simulation et émulation, baptisé Moteur de Virtualisation (vEngine), permet la simulation de composants virtuels directement au sein du bâtiment. Son architecture multiprocessus et légère mène à des simulations efficaces, dans un environnement modulaire favorable aux développeurs. Ensuite, nous décrivons notre Système Libre de Gestion d'Energie (OpenEMS), une plateforme qui se connecte de manière transparente au BDMS pour fournir un environnement de développement pour prototyper des algorithmes de gestion de l'énergie, en particulier de la Commande Prédictive (MPC). Des simulations réalisées sur des modèles de bâtiments résidentiels suisses démontrent l'efficacité et la modularité des deux outils. De plus, nous proposons un algorithme d'identification de profil de charges à états multiples, conçu spécialement pour de la Surveillance Non Intrusive de Charges (NILM).

Appliqué à la désagrégation d'énergie, l'algorithme montre des résultats intéressants visant un feedback amélioré pour les occupants du bâtiments.

Afin d'atteindre une balance énergétique quotidienne dans le réseau électrique intelligent, nous proposons plusieurs algorithmes et structures de gestion énergétiques impliquant les bâtiments intelligents. Une formulation incrémentale de MPC est dérivée dans l'objectif de mieux balancer les coûts associés à l'énergie et aux pics de demande des bâtiments commerciaux. Des simulations montrent un bénéfice non-négligeable de cette méthode, autant pour le propriétaire que pour le réseau électrique. En outre, nous présentons un cadre décentralisé pour la gestion automatisée de l'énergie d'une communauté de bâtiments, en présence de RES. Basé sur la technologie blockchain et des contrats intelligents, ce cadre de gestion optimise un objectif commun à l'entièreté de la communauté sans l'aide d'un agent central. Un groupe de bâtiments résidentiels suisses est simulé, prouvant que le comportement coopératif mis en place permet la réduction des pics de demande et une meilleure utilisation des ressources locales, en comparaison aux optimisations individuelles. Finalement, nous suggérons un modèle unifié de BtG qui pourrait être bénéfique aux organismes d'agrégations impliqués dans les microgrid ou les marchés de l'énergie. En utilisant des modèles de l'état de l'art, une équation de batterie de second ordre permet de capturer la dynamique thermique des bâtiments, accompagnée de structures de données représentant les charges déférrables et les véhicules électriques.

Mots-clés : bâtiment intelligent, système de gestion de l'énergie, commande prédictive, simulateur intégré, désagrégation de l'énergie, réseau intelligent, réponse de la demande, blockchain, algorithme décentralisé, communauté intelligente, bâtiment-vers-réseau

Acknowledgements

This PhD project has been a 4-year journey that would never have been possible without the help of the people that kept pushing me forward. First of all, I would like to thank Pr. Maher Kayal, my thesis advisor, who offered me the opportunity to carry out my research. He gave me a total freedom that I could leverage to roll out the numerous research ideas I had throughout my thesis. Thanks are also due to QEERI who funded this PhD project.

During the first two years of my thesis, I got the chance to collaborate with Dr. Georgios Lilis who greatly helped me finding my research path. I would like to express my gratitude for the time he spent guiding and advising me about smart-buildings. I also wish to acknowledge the help provided by Dr. Rachid Cherkaoui concerning the smart-grid and energy markets. As a PhD candidate, I designed and supervised many master/semester projects (11 to be precise) and met many students from which I learned a lot. In particular, I would like to offer my special thanks to David Ho Dac, Pol Boudou, Mélanie Gaillet-Tournier, Max Chevron, Marina Dorokhova, and Tobia Wyss; working with them went beyond traditional assistantship.

In the never-aging building ELB, I would like to thank the people I spent my working days with. I shared my office with François Gaugaz for almost my entire PhD studies and I could not have dreamed of a better officemate; thank you for all the good times, the jokes, and your friendship. A warm thank to Bakul Vinchhi for the numerous extended coffee breaks/lunches, his positive attitude in any circumstance, and the amazing discussions we had. To complete the gang, I would like to thank Evgenia Voulgari and Aristea Grammoustianou who brought the sunshine of Greece in the lab. I wish to thank the co-founders of ThinkEE, Johann Bigler and Jean-Charles Fosse, for their support, the fruitful collaboration, and the intense sessions of running under the rain. I would also like to extend my thanks to Raymond Sutter, Isabelle Buzzi, Karin Jaymes, Jean-Michel Sallese, Adil Koukab, Cédric Meinen, Guillaume Lanz, Nisrina Abdo, and Denis Sallin.

My experience in Berkeley in 2018 and 2019 would not have been possible without Mary-Ann Piette and Rich Brown, to whom I am particularly thankful. At the Lawrence Berkeley National Laboratory, joining the incredible summer-team of Dr. Marco Pritoni was a big leap forward in my PhD studies, as he gave me the opportunity to work on real-life exciting Californian projects. I would like to express him my very great appreciation for the constructive discussions and critics on my research, and in general for the time he always managed to find in his busy

agenda. In the frame of the Solar+ project, I also wish to acknowledge the unconditional help provided by Anand Prakash, and the interesting discussions with Dr. David Blum and Dr. Peter Alstone about the future of smart-buildings. Finally, I would also like to extend my thanks to the LBNL summer-team members, Callie Clark, Pranav Gupta, and, of course, Paula Ashley.

'You Can Choose Your Friends, but Not Your Family' as the saying goes; sharing a home with people sometimes does not follow that rule, and it has been an incredible fruitful experience to me. In the Atrium and Coloc de Montoie in Lausanne, and in the Birdhouse in Berkeley, I shared my daily life with people from which I learned a lot and who were always there to listen to my endless list of PhD-related problems. In addition, sport has been an important part of my PhD life, helping me to release the stress and frustration. I would like to thank the Squash gang and Ornella De Gasperin for giving me a hard time on the courts, Germain Hugon and the EtapeDuTour gang for cycling with me for hours in Lausanne and its surrounding mountains, Bérénice Charrez, a.k.a. Bernie, Daniel Raymundo, a.k.a Raymundade, and Lucas Fuentes, a.k.a. PGM, for the intensive biking & running sessions in the Bay Area, and Colin Darbelay for the jogging around EPFL whatever the weather. If PD was a sport, thank you Pierre, Margaux, and Robin for the weekly sessions.

Despite the distance, I feel blessed for keeping the friendship with my friends in Belgium. A warm thank to Les Mecs Cools and our incredible WKMCs (special thanks to Valentin Fontaine, Joffrey Coheur, and Thibaud Ernotte for helping me to improve this manuscript) and the AirPur gang. Hopefully, I could count on the help of many people to bring the spirit and bonnardise of my dear beer-loving country in the cheese-loving one. First and foremost, I would like to express my gratitude to my dear friend Dr. Simon Schaerlaeken. I would need an entire manuscript to thank you for all the great moments we spent together, from the amazing "Excepte Une Fois Au Chalet" birthday parties we organized, to the simplest space exploration journeys. A very special gratitude goes out to Dr. Jonathan Narinx and (soon-to-be-Dr.) Valentine Magnin, a.k.a. Les Biscuits, YoungJoo Seo a.k.a. Crazy Korean, Heather Carlson, a.k.a. H, Patricia Cernadas, a.k.a. Patoche, Paul-Antoine Spies, a.k.a. PA ArgileMan, Coralie Debracque, a.k.a. SopranoFan, Bernardo Zan, a.k.a. Kung Fury, Hubert Bonisseur de La Bath, a.k.a. Noël Flantier, François Rivest, a.k.a. Franz, Caroline Contat, a.k.a. Caro, and many more. It was fantastic to have met you and shared the adventure in Switzerland with you all.

Finally, I would like express my deep gratitude to Alba, who was always present to cheer me up during the harsh moments and demonstrated an infinite patience with me. Thank you for your unconditional love. Last but not least, I would like to thank my family: my brothers Nicolas and François, their partners Joanne and Heidi, and my amazing parents.

Olivier Van Cutsem
Lausanne, November 11, 2019

Contents

Abstract **iii**

Acknowledgements **vii**

List of figures **xi**

List of tables **xvi**

Acronyms **xix**

Introduction **1**

I Smart-Building Structure **7**

1 State-of-the-art Building Energy Modeling and Simulation **9**
1.1 Building Components Modeling 9
1.1.1 Thermal loads 9
1.1.2 Deferrable loads 16
1.1.3 Non-controllable loads 17
1.1.4 Energy storage systems 18
1.1.5 Local renewable energy production 21
1.2 Existing Building Simulation Tools 22

2 An Integrated Simulator for Smart-Buildings **25**
2.1 Introduction and Motivation 26
2.2 Virtualization Engine: an Integrated Building Simulator 27
2.2.1 Virtual entity concept 29
2.2.2 Simulation coordinator 31
2.2.3 Simulated entity models 36
2.3 Case study: a Minergie Building 41
2.3.1 Minergie building model 41
2.3.2 Energy simulation 46
2.4 Conclusion and Outlook 49

3 Multi-State Load Modeling for Energy Disaggregation 53
3.1 Introduction and Motivation 54
3.2 State-of-the-art Load Modeling and Energy Disaggregation 55
3.2.1 Load modeling 55
3.2.2 Energy disaggregation 56
3.2.3 Factorial hidden markov model applied to NILM 58
3.3 Multi-State Load Profile Identification Algorithm 59
3.4 Case Study on the ECO Dataset 65
3.4.1 The ECO dataset 65
3.4.2 Load power disaggregation 66
3.4.3 Activity set identification 71
3.5 Conclusion 73

II Advanced Integration of Smart-Buildings into the Smart-Grid 75

4 Smart-Building Energy Management 77
4.1 Energy Management in Buildings 78
4.1.1 Model-predictive control applied to building energy management 81
4.2 MPC as a Means to Provide Grid Services 83
4.2.1 Price responsive smart-buildings 83
4.2.2 Event responsive smart-buildings 84
4.3 OpenEMS: a Generic Open-Source EMS 86
4.3.1 BMS interface 88
4.3.2 Data management module 88
4.3.3 Energy management core module 90
4.4 OpenEMS Case Study: Energy Management of a Minergie Building 91
4.4.1 Baseline: traditional control 91
4.4.2 MPC formulation tailored to a Minergie building 92
4.4.3 Simulation results 96
4.5 Conclusion 99

5 MPC Applied to Commercial Buildings: Balancing Energy and Peak Demand 103
5.1 Commercial Time-Of-Use Rates and MPC 104
5.2 Economic MPC to Tackle Demand Charges 107
5.2.1 Common "naive" approach to Demand Charges 108
5.2.2 Incremental approach to demand charges 109
5.3 MPC Formulations Comparison: Case Studies 111
5.3.1 Case Study 1 - Simplified simulated commercial buildings 111
5.3.2 Case study 2 - Advanced simulated commercial buildings 119
5.4 Conclusion 123

6 Decentralized Demand Response in a Community of Smart-Buildings 125
6.1 Introduction and Motivation 126
6.1.1 Blockchain and smart-contracts 129
6.2 Energy Actors Modeling 131
6.3 Cooperative Decentralized DR Framework 134
6.3.1 Decentralized planning algorithm 135
6.3.2 Online phase: tracking and monitoring 137
6.3.3 Community objective 140
6.4 Case Study and Discussion 141
6.4.1 Simulation platform 141
6.4.2 Scalability analysis 144
6.4.3 Community simulation scenarios 146
6.4.4 Discussion 150
6.5 Conclusion 152

7 Toward a Unified Smart-Building Model 153
7.1 Smart-Grid, Energy Markets, and Microgrids: the Role of Smart-Buildings . . . 155
7.2 Building-to-Grid Data Structures 158
7.2.1 Demand bidding curves 159
7.2.2 Flexibility contractual framework 161
7.2.3 Battery equivalent model 163
7.3 A Generic Building-to-Grid Model Proposition 171
7.4 Conclusion 175

Conclusion and Outlook 177

A Appendix 181
A.1 Building Models 181
A.1.1 Minergie residential model 181
A.1.2 Commercial building model 186
A.1.3 Blue Lake Rancheria test site 187
A.2 Virtualization Engine 192
A.2.1 The vEngine in emulation mode 192
A.2.2 OpenBMS example 194
A.3 ELAB Smart-Building Simulator 195
A.3.1 Open Building Data Management System 196
A.3.2 Open Energy Management System 197
A.4 Non-Controllable Loads Model 197
A.5 Battery Equivalent Model 198

Bibliography 219

Curriculum Vitae 221

List of Figures

1 Time evolution of publications involving Demand Response (DR, in blue) and Smart Building (SB, in red) per type of article. Data from *IEEE Xplore* and *ScienceDirect* using keywords "demand response" and "{smart, intelligent}{building, home, house}", respectively. . . . 2

1.1 Generic zone thermal model 10
1.2 RC equivalent circuit of zone thermal model 11
1.3 Battery KiBaM model 19
1.4 PV one-diode model: equivalent circuit 22

2.1 *vEngine* architecture and connection with the BMS 28
2.2 Generic *vEntity* flowchart 30
2.3 Simulation coordinator logic flow to update internal structures 32
2.4 Cycle removal illustration in *vCoordinator* (left) BMS representation of *vEntity* relationship (right) use of a buffer by the coordinator to remove deadlock cycles 34
2.5 *vEntities* implemented in the current version of the *vEngine* 36
2.6 Step-by-step procedure to derive *vEngine* building model 41
2.7 *vEngine* model of Minergie building 42
2.8 Minergie simulation environmental data 43
2.9 RC model of Minergie building thermal system 44
2.10 Minergie building simulation (top) Non-controllable load consumption (bottom) Controllable load consumption 51
2.11 Minergie building simulation: aggregated data 52
2.12 Minergie building simulation: temperature comfort 52

3.1 FHMM: A graphical interpretation of appliances hidden states and the whole aggregated consumption 58
3.2 Load profile modeling: example of modes and activities 60
3.3 Flowchart of the appliance state identification algorithm 61
3.4 Histogram of an air exhauster's power consumption. In red, the centroid of the identified clusters and in green, the corresponding estimated normal distribution 62
3.5 Air exhauster power consumption: DB clustering index and cost-error function used for states identification 62

3.6 Histogram and automatic states identification for some of the ECO dataset appliances. In red, the centroid of the identified clusters and in green, the corresponding estimated normal distribution. 67
3.7 Snapshot of outlet power consumption and its estimation for the set {Stereo, Laptop, Entertainment}. (Top) On/OFF modeling (Bottom) Multi-state modeling 69
3.8 Computation time (in seconds) of the load disaggregation process as a function of the logarithm of the total number of states of the combined FHMM 70
3.9 Activity clustering by activity duration and (left) starting time or (right) ending time. From top to bottom rows: laptops, entertainment system, and stereo system. 72

4.1 MPC applied to buildings: system overview 82
4.2 Illustration of MPC horizon of optimization. 82
4.3 OpenEMS distributed architecture 86
4.4 OpenBMS-OpenEMS interactions timeline 87
4.5 Smart-building electrical entities categorization 89
4.6 OpenEMS applied on a Minergie building: comparison between traditional control and MPC (top) grid consumption (bottom) battery state of charge . . . 97
4.7 OpenEMS applied on a Minergie building: focus on MPC 98
4.8 OpenEMS applied on a Minergie building: temperature comfort (top) traditional control (bottom) MPC 101

5.1 Monthly Time-Of-Use Demand Charges: example of on- and mid- peaks on a 6 days power consumption signal 105
5.2 Illustration of the "Naive" MPC approach to tackle demand charges 110
5.3 Illustration of the "Incremental" MPC approach to tackle demand charges . . . 110
5.4 Case Study 1: Building RC thermal model 112
5.5 Case Study 1: Uncontrollable load profile (left) Retail store (right) Secondary school. Maximum AC capacity in green: (filled) Warm and Hot environment, (dashed) Mild environment. 114
5.6 Case Study 1: Outside temperatures in July, for 3 environments. 114
5.7 Case Study 1: Relative *decrease* of the maximum power demand of the month, compared to control Method A. For each control method: (left bar) Retail store, (right bar) Secondary school. 116
5.8 Case Study 1: Relative *decrease* of the monthly bill, compared to control Method A. For each control method: (left bar) Retail store, (right bar) Secondary school. 117
5.9 Case Study 1: Relative *loss* of energy shifting potential during CPP events, compared to control Method A. For each control method: (left bar) Retail store, (right bar) Secondary school. 118
5.10 Case Study 1: Timeseries analysis of "secondary school" control methods (left) hourly mean of power consumption, energy price and outside temperature throughout all simulations (right) 15-min power consumption on the 3rd CPP event. 119

5.11 Case Study 2: environmental conditions 120
5.12 Case Study 2: (left) MPC cost function (right) Power consumption 121
5.13 Case Study 2: zones temperature comfort 122
5.14 Case Study 2: comparison of MPC computational time 123

6.1 Blockchain is a list of *blocks*, linked to one another by the unique previous block hash. Each block i contains N_i transactions. 130
6.2 Smart-Buildings community with local RES: structure, communication, and power flows: centralized aggregator approach 134
6.3 Smart-Buildings community with local RES: structure, communication, and power flows: autonomous decentralized approach 134
6.4 Blockchain-based decentralized simulation setup 142
6.5 Outside temperature and irradiance used for simulation 143
6.6 Planning phase: algorithm iterations as the the community size grows 145
6.7 Planning phase: steady-state cost function disparity as a function of the community size 145
6.8 Planning phase convergence for a community of 8 buildings. Individual curves refer to a given day-ahead algorithm sequence. 146
6.9 Load profiles resulting from the Cooperative Planning Phase in the community of 8 Minergie buildings, in the presence of **Market prices**, and **with RES** (dashed green): (top left) non-cooperative load profiles (top right) cooperative load profiles (bottom) aggregated profiles. In the top row graphs, each colored dashed-line corresponds to a community participant power profile, either a RES or a smart-building profile, and their aggregation leads to the bottom graphs. 147
6.10 Load profiles resulting from the Cooperative Planning Phase in the community of 8 Minergie buildings, in the presence of **Quadratic prices**, and **with RES** (dashed green): (top left) non-cooperative load profiles (top right) cooperative load profiles (bottom) aggregated profiles. In the top row graphs, each colored dashed-line corresponds to a community participant power profile, either a RES or a smart-building profile, and their aggregation leads to the bottom graphs. . 148
6.11 Load profiles resulting from the Cooperative Planning Phase in the community of 8 Minergie buildings, in the presence of **Quadratic prices**, with **behind-the-meter PV**, and no RES: (top left) non-cooperative load profiles (top right) cooperative load profiles (bottom) aggregated profiles. In the top row graphs, each colored dashed-line corresponds to a community participant power profile, either a RES or a smart-building profile, and their aggregation leads to the bottom graphs. 149

7.1 Smart-grid actors representation. 155
7.2 Energy market actors 156
7.3 Microgrid actors representation 156
7.4 Demand bid illustration 159

7.5 The evolution of commercial building demand bids over one day 160
7.6 Flexibility contract illustration 161
7.7 Battery Equivalent Model illustration 164
7.8 BEM excitation signals at t=12h: effect on the room temperature (top) Room 1 (bottom) Room 2 167
7.9 BEM capacity 168
7.10 BEM self-discharge 168
7.11 Building total power consumption under BEM-based MPC (red) and perfect MPC (blue) 169
7.12 Zones temperature evolution under BEM-based MPC (red) and perfect MPC (blue); (top-left) Zone 1 (top-right) Zone 2 (bottom) Zone 3 171
7.13 A generic building-to-grid data structure proposition 172
7.14 Generic KiBaM model for BtG application (left) Zone conditioned through the air (right) Zone conditioned through the mass/water 172

A.1 3-zone segmentation model 182
A.2 Air-to-water heat pump Coefficient Of Performance (COP) model (Indoor temperature = supply water temperature) 185
A.3 BLR building and casino microgrid 187
A.4 *Solar+* project: PV installation on gas station and convenience store 189
A.5 *Solar+ Optimizer* software architecture and its connection with microgrid entities 191
A.6 *Solar+ Optimizer* software: Modelica bloc diagram representation of (left) system model (right) thermal building model 192
A.7 Building Data Management System (BDMS) example of virtual entity encoding (top) Principal characteristics (bottom) Dependencies 195
A.8 GUI of the ELAB Smart-Building simulation project 196
A.9 The BMS architecture used in this thesis, schema from [1] 197
A.10 UML class diagrams of OpenEMS (top) Energy-related classes (bottom) Zone-related classes 199
A.11 Activity clustering: duration time versus (left) starting time (right) ending time. The top figures refer to a living room lamp, and the bottom figures to the TV in the ECO dataset 200

List of Tables

2.1 Minergie building energy simulation: simulator statistics 48

3.1 Automatic states identification of appliances in Household 2 of ECO dataset . . 66
3.2 Comparison between binary and multi-state FHMM disaggregation results, trained for 30 days 68
3.3 Comparison between binary and multi-state FHMM disaggregation results, trained for 60 days 70

5.1 Nomenclature used in Chapter 5: parameters 107
5.2 Nomenclature used in Chapter 5: variables and notations 107
5.3 Case Study 1: Simulated buildings equivalent parameters 113
5.4 Case Study 1: description of the implemented MPC methods. 115
5.5 Case Study 1: PG&E summer tariff rates - off-peak = week-days 10pm - 8am, week-ends & holidays, mid-peak = week-days 8am-12pm & 6pm-10pm, on-peak = week-days 12pm-6pm. 115

6.1 Nomenclature used in Chapter 6: parameters 126
6.2 Nomenclature used in Chapter 6: variables 127
6.3 Nomenclature used in Chapter 6: conventions and notations 127
6.4 Cooperative algorithm applied to *Minergie* community - Scenario *Greeen community with Market-price.* Scenarios (1) includes an independent RES, while scenarios (2) and (3) only involve buildings. 150
6.5 Cooperative algorithm applied to *Minergie* community - Scenario *Grid-services with quadratic price.* Scenarios (1) includes an independent RES, while scenarios (2) and (3) only involve buildings. 150

7.1 Nomenclature used in Chapter 7: Parameters 154
7.2 Nomenclature used in Chapter 7: Variables 154
7.3 Nomenclature used in Chapter 7 - Conventions and notations 154
7.4 Summary of existing and proposed building-to-grid data structures 175

A.1 Minergie building - Resistance-Capacitance (RC) equivalent parameters 182
A.2 Minergie building zones area 183

A.3 Minergie building wall properties 183
A.4 Minergie building floor properties 184
A.5 Blue Lake *Solar*+ demonstrator: new installed hardware 190

Acronyms

uTread micro-thread.

vCoordinator Simulation Coordinator.

vEngine Virtualization Engine.

vEntity Virtual Entity.

vMid Virtual Middleware.

AC Air Conditioner.

ANN Artificial Neural Network.

API Application Programming Interface.

AWHP Air-source to Water Heat-Pump.

BDMS Building Data Management System.

BEM Battery Equivalent Model.

BLR Blue-Lake Rancheria.

BtG Building-to-Grid.

CAISO California Independent System Operator.

COP Coefficient Of Performance.

CPP Critical Peak Pricing.

CPU Central Processing Unit.

DAM Day-Ahead Market.

DB Davies-Bouldin.

DLC Direct-Load Control.

DOE Department of Energy.

DR Demand Response.

DSM Demand Side Management.

ECO Electricity Consumption and Occupancy.

EE Energy Efficiency.

EM Expectation Maximization.

EMS Energy Management System.

EPRI Electric Power Research Institute.

ESS Energy Storage System.

EV Electric Vehicle.

EWH Electrical Water Heater.

FHMM Factorial Hidden Markov Model.

FMI Functional Mock-up Interface.

FN False Negative.

FP False Positive.

FSM Finite State Machine.

GT Game-Theory.

GUI Graphical User Interface.

HMM Hidden Markov Model.

HP Heat-Pump.

HVAC Heating, Ventilation and Air Conditioning.

ICT Information and Communication Technology.

IoT Internet of Things.

IOU Investor-Owned Utilities.

ISO Independent System Operator.

KiBaM Kinetic Battery Model.

LBNL Lawrence Berkeley National Laboratory.

Li-Ion Lithium-Ion.

LP Load Profile.

MAS Multi-Agent Simulation.

MILP Mixed-Integer Linear Programming.

MLE Maximum Likelihood Estimation.

MPC Model Predictive Control.

MPPT Maximum Power Point Tracker.

NILM Non-Intrusive Appliance Load Monitoring.

NREL National Renewable Energy Laboratory.

OpenADR Open Automated Demand Response Standard.

OpenBMS Open Building Data Management System.

OpenEMS Open Energy Management System.

OS Operating System.

P2P Peer-to-Peer.

PAR Peak-to-Average Ratio.

PG&E Pacific Gas & Electricity.

PI Proportional Integral.

PV Photovoltaic.

RC Resistance-Capacitance.

RES Renewable Energy System.

RMSE Root Mean Square Error.

RTM Real-Time Market.

RTP Real-Time Pricing.

RTU Roof-Top Unit.

SoC State-of-Charge.

TCL Thermostatically Controlled Load.

TOU Time-of-Use.

TP True Positive.

UML Unified Modeling Language.

VB Virtual Battery.

VPP Virtual Power Plant.

XBOS eXtensible Building Operating System.

Introduction

In recent decades, electrical power systems have experienced unprecedented changes in the way energy supply and demand are continuously balanced. Worldwide, countries and states governments are increasingly promoting the deployment of Renewable Energy System (RES) to face climate change and to provide clean energy to a growing population. For instance, by 2050 the European Union (EU) aims to reduce its emissions by 85-90% compared to its 1990 levels and California, the world's fifth-largest economy were it a sovereign nation, pledged to be carbon-free by 2045 [2, 3]. However, the intrinsic volatility of wind and solar energy completely redefines the paradigms of the electrical grid. Initially designed as a unidirectional system, that provided energy for end-consumers from a set of large dispatchable power plants, the unpredictability of RES leads to the need for a system able to follow a non-controllable supply [4, 5]. Moreover, the distributed nature of RES calls for a restructuring of grid architecture to enable adequate monitoring and control.

Residential and commercial buildings hold the largest shares in the demand in electricity, accounting for about 70-75% of the electricity consumption in the U.S. [6]. Increasingly, they become more equipped with ambient sensors, actuators, and connected devices, accelerated by the adoption of Information and Communication Technology (ICT) and Internet of Things (IoT). This led to the emergence of efficient BDMSs, backbones of connected buildings, which gather information from sensors, communicate with actuators, and offer high-level services to third party applications [7]. When supported by a BDMS, the resulting cyber-physical system is referred to as *intelligent building* or a *smart-building*. Rendered aware of its own energy consumption and flexibility potential, the smart-building presents an ideal candidate to assist the *smart-grid* in its everyday task of balancing the power system [8].

Definition. *Smart* - The use of the "smart-" prefix with existing entities refers to their enhancement by sensor/actuator hardware, telecommunication links, and computer-based algorithms, enabling them to sense, monitor, and automatically control their states.

Demand Response (DR) regroups the methods and strategies to dynamically shape end-users' consumption profiles according to the needs of the smart-grid [9]. Introduced in the 1980's by Electric Power Research Institute (EPRI), current DR programs use electricity prices and incentive-based contracts to enforce changes in the consumption of loads, with respect to their normal operations. From the grid perspective, DR can be used to provide energy

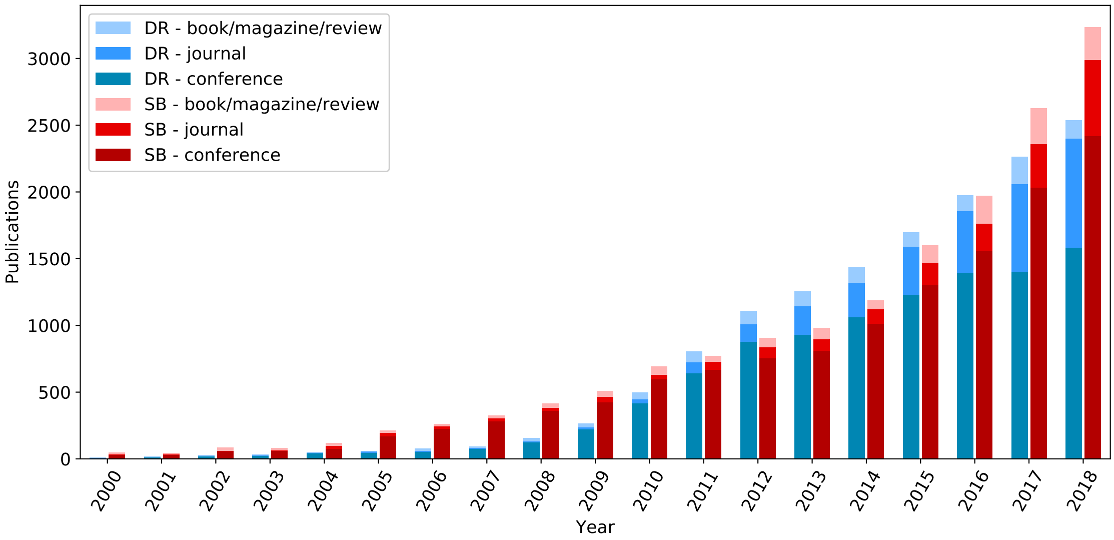


Figure 1 – Time evolution of publications involving Demand Response (DR, in blue) and Smart Building (SB, in red) per type of article. Data from *IEEE Xplore* and *ScienceDirect* using keywords "demand response" and "{smart, intelligent}{building, home, house}", respectively.

services such as energy arbitrage or peak shaving, capacity reserves, and Ancillary Services (AS) such as frequency regulation. Therefore, a considerable amount of the new generation and storage facilities, which would normally be required to provide these services, could be saved through the deployment of DR. From the standpoint of a building, providing DR means changing the normal operations of its electrical entities; often referred to as flexible demand. Inertial loads, typically large Heating, Ventilation and Air Conditioning (HVAC) systems and domestic Electrical Water Heaters (EWHs), can be temporarily disrupted without impacting user comfort. In addition, Heat Pumps (HPs), Electric Vehicles (EVs), deferrable appliances, secondary Energy Storage Systems (ESS), and behind-the-meter Photovoltaic (PV) systems are also typically encountered in flexible smart-buildings.

Yet, DR faces many barriers on its way to large scale integration into the smart-grid and its energy markets. In general, there is a considerable uncertainty around the value of DR and hence, around its potential revenue for participants [10, 11]. While many DR programs exist around the world, they lack normalization and unification in their implementations. In addition, their current measurement methods and corresponding remunerations have severe limitations. Finally, the inclusion of numerous, small RES at the distribution level of grids, requires loads to continuously react to varying supplies, calling for an update in DR definition and structure.

Smart-buildings have the potential to enable the flexibility management of energy demands in systematic and structured ways and hence, can greatly benefit DR. As shown in Figure 1, both domains have seen extensive interest due to the push to find carbon-free solutions for our society and the rise of ubiquitous, affordable ICT. However, most of the existing solutions are designed for specific problems and individual test cases; there is a clear lack of a generic and widely accepted approach to creating smart-buildings and connecting them to the smart-grid. Furthermore, the traditional DR programs do not match the envisioned needs of the future RES-based smart-grid and electrified houses.

Thesis Contribution

The aforementioned issues and drawbacks around DR motivated the research project described in this Ph.D. dissertation. It aims to fill the gap between grid-level DR and its efficient implementation at the building level. To do so, we provide in this thesis a set of tools and frameworks revolving around the notion of the smart-building and the management of its energy. We focus mainly on daily energy arbitrage and day-ahead optimization of the energy consumptions of residential and commercial buildings[1].

The growing adoption of BDMS in residential and commercial buildings paves the way toward a structured entity, able to efficiently federate a large panoply of sensors, actuators, and other IoT-based hardware, working with different technologies [13]. Leveraging building metadata and time-series data stored in the BDMS, new kinds of building simulations can be envisioned, more closely resembling the environment they emulate. This motivated our efforts to create an innovative high-speed integrated building simulator that could seamlessly plug itself into any existing BDMS. In addition to designing of the simulation engine, we also developed an open-source version with a set of useful libraries which can be used by the research community.
Still based on the notion of BDMS, we designed an algorithm that systematically identified the various active power states of a load from load power consumption data stored in time-series databases. The resulting model has useful applications in energy disaggregation.

Zooming out and considering smart-buildings as active components of a smart-grid, their connections with the latter are made through building Energy Management Systems (EMSs). We designed and developed an open-source EMS tool that automatically connects to existing BDMSs, hybridizing them to form grid-responsive smart-buildings. Then, we derived a method combining an economic Model Predictive Control (MPC) formulation with a day-ahead deferrable load dispatch algorithm, integrating them into the EMS tool.
Dealing with commercial buildings subject to peak demand charges, a new MPC formulation was engineered for better management of both the owner's bill and the grid flexibility.

Considering a communities of smart-buildings, we further designed a two-phase blockchain-based decentralized framework, able to reduce the overall community peak and optimize local resources. Solidity smart-contracts were derived for an effective deployment of the framework, and the corresponding code was rendered available to the research community.

---

[1] We have not tackled long-term provision of energy/capacity through smart-buildings nor the sizing of power-system elements such as RES components. In addition, we did not investigate the potential for frequency/voltage regulation [12], happening at a much shorter time scale (seconds) than the one considered in this thesis (5-15 minutes). It's also worth noting that we solely considered the active part of the electrical power.

Finally, in the context of aggregators, we compiled a thorough review and discussion of Building-to-Grid (BtG) data structures, to subsequently derive a unified smart-building flexibility model.

Thesis Organization

This thesis manuscript is divided into two main parts. In the first part, the smart-building is studied as an individual entity, from an electrical consumption standpoint. The second part deals with the algorithms, frameworks, and data structures needed for the active integration of these smart-buildings into the electrical grid, providing energy flexibility. The chapters are organized as follows:

PART I - SMART-BUILDING STRUCTURE

- **Chapter 1** reviews the state-of-the-art building models and existing energy simulators. The presented models cover electrical loads and appliances, energy storage systems, and local energy generation. We also focus on thermal models of zones in buildings, mainly the RC model, as their coupling with HVAC systems holds a large potential for energy flexibility.

- **Chapter 2** presents an innovative integrated smart-building simulator, dubbed Virtualization Engine, that emulates the presence of common buildings' entities. The open-source software plugs seamlessly into any BDMS and has the advantages of being lightweight, modular, and multi-threaded. We illustrate the efficiency of the tool through simulations of an entire Swiss house, with a floor-heating system, connected to a HP that can be controlled in addition to an EWH, an EV, and deferrable loads.

- **Chapter 3** proposes an algorithm for automatically identifying the power states of electrical loads from historical outlet-level data. Then, we leverage the resulting multi-state load modeling for energy disaggregation using Hidden Markov Model (HMM). Simulations based on a fine-grained building electricity consumption dataset show that modeling loads, with multiple power states, better identifies individual appliances participation, when compared to ON/OFF modeling.

PART II - ADVANCED INTEGRATION OF SMART-BUILDINGS INTO THE SMART-GRID

- **Chapter 4** reviews common building EMS strategies and presents an open-source building EMS platform tailored for model-based algorithm development, named OpenEMS. After a thorough description of the use of MPC for providing grid services, we illustrate the modularity and user-friendly features of the OpenEMS tool by controlling the above-mentioned simulated Swiss house. The Economic MPC is proven to be more energy efficient and grid-responsive compared to traditional control.

- **Chapter 5** introduces an innovative Economic MPC formulation for better balancing the various electricity costs charged to commercial buildings in USA. Simulations of HVAC-driven commercial buildings prove that the new MPC framework reduces overall bills while greatly improving the flexibility provided to the grid.
- **Chapter 6** describes a blockchain-based framework for managing the energy of a community of smart-buildings, in the presence of local RES. We derive a set of distributed algorithms coded in Solidity that together enable fully decentralized, cooperative, autonomous two-phase energy arbitrage. Combining the tools presented in Chapters 2 and 4, sees fleets of smart-buildings and RES efficiently emulated in the Ethereum-based framework; we show that the cooperative algorithm reduces individual bills in addition to enabling community-level grid services.
- **Chapter 7** reviews existing state-of-the-art BtG data structures which can be used by aggregators. A total of three distinct models are studied in detail and simulations of a commercial building highlight their features. Finally, we suggest a generic BtG data structure which has the potential to incorporate multiple models of buildings' flexibility entities.

Finally, a **Conclusion** chapter summarizes the findings of this thesis and further discusses the future outlook of the role of a smart-building within the smart-grid.

Smart-Building Structure Part I

1 State-of-the-art Building Energy Modeling and Simulation

1.1 Building Components Modeling

A building consumes or provides electrical power, $e[h]$, to the grid, depending on the internal load consumption, energy storage, and local production. At any time period h, it can be expressed as:

$$e[h] = \overbrace{\sum_{n=1}^{N_l} p_l^n[h]}^{\geq 0} + \underbrace{p_b[h]}_{\leq 0 \text{ or } \geq 0} + \overbrace{p_g[h]}^{\leq 0} \tag{1.1}$$

where p_l^n denotes the power consumption (kW) of the n^{th} load, p_b, the charging/discharging power (kW) of the ESS, and p_g, the generated power (kW) of the local RES. In this discrete form, power quantities represent average power consumption/production over the given sampling period dt.

The parameter N_l denotes the total amount of *loads* in the building. The main drivers of building consumption, i.e., building loads draw electricity in order to perform a specific task serving the user.

Definition 1.1.1. Load - A load is an electrical entity that consumes a positive (or null) amount of power.

1.1.1 Thermal loads

Definition 1.1.2. Thermal load - A thermal load denotes any load that impacts the thermal comfort of a given space.

Thermal comfort is one of the most important metrics in buildings. The corresponding loads strive to ensure that the temperature of the air in occupied building zones or water in tanks remains within comfort bounds. HVAC refers to the set of thermal loads and technology commonly deployed in buildings to regulate indoor air temperature and air quality. They

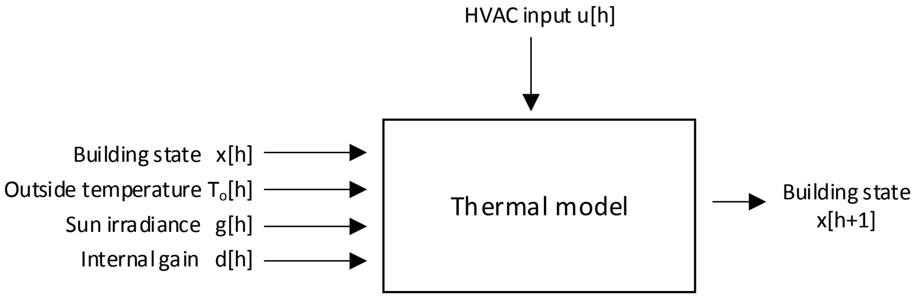


Figure 1.1 – Generic zone thermal model

have been thoroughly studied in the literature [14], with models emerging, both simple and complex ones, at the intersection of many engineering fields. Hot water tanks are entities that store hot water to be delivered to the user or used in water pipes. A heating system generally provides heat to the water tank, the more common being EWH and HP.

Fig. 1.1 illustrates the generic model that maps the updated zone state $x[h+1]$ to the control HVAC signal $u[h]$, the current zone state $x[h]$, and influencing parameters. The latter comprises outside temperature, sun irradiation, and internal gain, such as loads and human-generated heat.

Formally, the state update of the entire building can be expressed through the thermal model function f:

$$\boldsymbol{x}[h+1] = f(\boldsymbol{x}[h], u[h], g[h], d[h]) \tag{1.2}$$

$$T_z[h] = C\, \boldsymbol{x}[h] \tag{1.3}$$

where $x[h]$ stands for the building thermal state, $u[h]$ the control input of the HVAC, $g[h]$ the incoming sun irradiation (W/m^2), $d(t)$ the internal gain (W), and C is a vector mapping the building state to the bound temperatures. The vector $x[h]$ may have a direct physical meaning (e.g., temperature nodes) or may be the result of an advanced data-driven modeling. The nature of the input $u[h]$ depends on the controllability of the HVAC at hand: electrical power consumption set point, air fan speed, or simply temperature set point.

Many methods exist in the literature to identify the thermal model function f in Eq. (1.2). They can be categorized in three main different families [6]:

- White-box model: Physical-based equations describe the detailed thermodynamics of the building. They include radiant and convective effects to model surface temperature and condensation, combined heat and mass transfer to account for air movement between zones, solar energy absorption depending on window properties, and many other effects. White-box models generally require extensive physical analysis by experts to derive the corresponding equations and parameters.

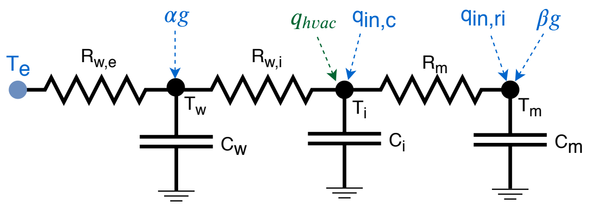


Figure 1.2 – RC equivalent circuit of zone thermal model

- Black-box model: It is a data-driven model agnostic of the underlying physics of the system it represents. Machine Learning algorithms are used to learn the relationship between output zone state (temperature) and influencing parameters (ambient condition, HVAC input, etc.). Black-box models require a large amount of on-site measurements for training, and are generally dependent on the operating conditions that lead to the collected data.

- Gray-box model: It is a simplified physical-based model using equations with physical meanings but without direct relationships to the actual building infrastructure. While gray-box models are less complex than white-box models, they require data to train the equivalent model parameters, which may depend on the training conditions as in the case of black-box models.

Resistance-Capacitance equivalent model

The most widespread gray-box model used to describe thermal building dynamics is the RC equivalent circuit, both for simulation and control [6, 15, 16, 17, 18, 19]. In this analogy, the electrical current (A) and the potential (V) are replaced by heat flow (W) and temperature (K), respectively. Similarly, electric resistance (Ω or V/A) and capacitance (F or C/V) are substituted by thermal heat resistance (K/W) and thermal heat capacitance (J/K), respectively. The zones constituting the building can therefore be modeled as a set of equivalent RC components, holding a physical meaning. Yet, as these parameters are generally an equivalent representation of an aggregation of multiple effects and sub-components (many layers and different materials in walls, air renewal and relative humidity, uneven geometry, furniture, heterogeneous air temperature repartition, etc.), they do not directly represent the building physically.

Fig. 1.2 shows a typical RC equivalent model used to represent a zone temperature evolution [19]. The model considers the zone temperature to be uniform (a single point), which is a common assumption [20]. The corresponding equations can be derived [1]:

---

[1] Eq. (1.5) considers that the zone is solely connected to one other zone (interior or exterior) at temperature T_e. There may be more zones connected to it: the first term on the right in Eq. (1.5) should be duplicated for each of them.

$$C_w \frac{dT_w}{dt} = \frac{(T_e - T_w)}{R_{w,e}} + \frac{(T_i - T_w)}{R_{w,i}} + \alpha G \tag{1.4}$$

$$C_i \frac{dT_i}{dt} = \frac{(T_w - T_i)}{R_{w,i}} + \frac{(T_m - T_i)}{R_m} + q_{hvac} + q_{in,c} \tag{1.5}$$

$$C_m \frac{dT_m}{dt} = \frac{(T_i - T_m)}{R_m} + q_{in,ri} + \beta G \tag{1.6}$$

where T_e, T_w, T_i, and T_m represent the external, wall, zone air, and zone mass temperatures, respectively. The external temperature can either be the outside air temperature or another node temperature (adjacent room, water pipes from heating system, etc.). C_w, C_i, and C_m are the equivalent capacitance of the wall, zone air, and zone internal mass, respectively. $R_{w,e}$ represents the equivalent resistance between the wall temperature node and the external temperature node, $R_{w,i}$ represents the equivalent resistance between the wall temperature node and the zone air, and R_m represents the equivalent resistance between the zone air and the zone internal mass. The HVAC input acts on the zone air temperature node [2] as a controllable source of current. Disturbances g and q_{in} stand for sun irradiance and internal heat gains coming from loads and occupants, respectively. The internal heat gain is split into conductive heat gain $q_{in,c}$ influencing the air temperature directly and the radiated heat gain $q_{in,ri}$ acting on the internal mass. Sun radiation influences the wall temperature through coefficient α and the internal mass through coefficient β.

The presented R3C3 model captures most of the dynamics of the conditioned space and surrounding effect and is generally a good candidate for a thermal model. Simpler and more advanced models have also been investigated in the literature [6, 15, 16, 17, 18, 19].

Commercial versus residential HVAC

HVAC systems designed for commercial buildings differ from the ones used in residential buildings. Commercial HVAC systems are generally large modular systems made of many sub-components to ensure proper air temperature and quality for all the conditioned spaces it controls. Most of them are set on the roof of commercial buildings, hence forming Roof-Top Units (RTUs). Residential HVAC system are mostly sold as a self-contained package that cannot be upgraded. This simpler system also leverages the fact that residential buildings have windows to naturally circulate the air.

Many possible configurations exist in commercial HVAC systems, although many large buildings share the same features [21, 22]. The basic principle of commercial HVAC is to provide conditioned air to target zones, by exchanging heat through a chiller or heater plant (cooling or heating), circulating the air using fans and to improve the quality of air through humidifiers and filters. A multitude of distributed controllers enable local regulation of water temperature, water flow, air flow, air temperature, and air pressure. We consider a common HVAC design made of a chilled water loop and a conditioned loop as in [21]. The general thermal Eq. (1.5)

---

[2] This modeling assumption holds for HVAC systems that exchange heat with the air directly. This might not be the case for the heating system acting on the internal mass such as the floor, instead of the air mass.

can therefore be written as:

$$C_i \frac{dT_i}{dt} = \frac{(T_w - T_i)}{R_{w,i}} + \frac{(T_m - T_i)}{R_m} + c_p \dot{m}(T_c - T_i) + q_{in,c} \tag{1.7}$$

$$P_{el}^c = \frac{q_{cooling}}{COP} = \frac{c_p \dot{m}(T_{a,s} - T_c)}{COP} \tag{1.8}$$

$$P_{el}^f = \alpha_3 \dot{m}^3 + \alpha_2 \dot{m}^2 + \alpha_1 \dot{m} + \alpha_0 \tag{1.9}$$

Eq. (1.7) describes the commercial zone heat exchanges, where T_c is the cooling coil discharge air temperature (K), $\dot{m}$ is the air mass flow (kg/s), and c_p is the specific heat capacity of air (kJ/kg.K). Eq. (1.8) represents the electrical consumption of the cooling system working at a specific COP to cool down supply air whose temperature is $T_{a,s}$. Eq. (1.9) denotes the electrical consumption of the fan where $\alpha_3, \alpha_2, \alpha_1, \alpha_0$ are fixed parameters. The sum of P_{el}^c and P_{el}^f lead to the global HVAC power.

The supply air results from the mix between outside air and air returning from all the zones:

$$T_{a,s} = \theta \frac{\sum_z^{N_z} \dot{m}_z T_z}{\sum_z^{N_z} \dot{m}_z} + (1 - \theta) T_{a,o} \tag{1.10}$$

where N_z is the number of conditioned zones, T_z is the internal air temperature of zone z, $T_{a,o}$ is the outside air temperature, and θ is the mixing factor.

Residential HVAC loads mainly include HP and electrical heater. Here, HP refers to a machine that uses electricity to extract heat from a source to release it to a sink of heat. It therefore encompasses common air-to-air Air Conditioner (AC) (cooling mode), air-to-water HP (heating mode), water-to-water HP (heating mode), and geothermal ground-to-water HP (heating mode). The ratio between the thermal energy transferred and electrical energy used by the HP is called COP (>= 1), and generally depends on working conditions such as outside and inside temperatures. Alternatively, an electrical heater converts electricity into heat (in air or water) through direct dissipation in electrical resistance, hence displaying poorer efficiency (<= 1) than HPs.

In case of air heating/cooling HVAC, general thermal Eq. (1.5) can be written as:

$$C_i \frac{dT_i}{dt} = \frac{(T_w - T_i)}{R_{w,i}} + \frac{(T_m - T_i)}{R_m} + u_{hvac}\, P_{hvac}^{cap} + q_{in,c} \tag{1.11}$$

where P_{hvac}^{cap} (kW) is the thermal capacity of the residential HVAC that determines the maximum amount of heat transfer, and u_{hvac} is the HVAC control input variable such that $0 \leq u_{hvac} \leq 1$.

In case of underfloor-heating, the heat from the heating system is first transferred to the water pipes, then from the pipes to the floor, and finally from the floor to the air zone. The general thermal Eq. (1.5) must therefore be changed accordingly:

$$C_i \frac{dT_i}{dt} = \frac{(T_w - T_i)}{R_{w,i}} + \frac{(T_m - T_i)}{R_m} + \frac{(T_f - T_i)}{R_f} + q_{in,c} \tag{1.12}$$

$$C_f \frac{dT_f}{dt} = \frac{(T_i - T_f)}{R_f} + \frac{(T_{pr} - T_f)}{R_{pf}} \tag{1.13}$$

$$C_{pr} \frac{dT_{pr}}{dt} = \frac{(T_f - T_{pr})}{R_{pf}} + \frac{(T_{ps} - T_{pr})}{R_{rs}} \tag{1.14}$$

$$C_{ps} \frac{dT_{ps}}{dt} = \frac{(T_{pr} - T_{ps})}{R_{rs}} + u_{hvac}\, P_{hvac}^{cap} \tag{1.15}$$

where T_f, T_{pr}, T_{ps} represent the temperature of the floor, return water in pipe, and supply water in pipe, respectively. The distinction between return and supply section means that the heat transfer from the water pipe to the floor is supposed to be done at the return section, while the heating system acts on the supply section. C_f, C_{pr}, C_{ps} represent the equivalent thermal capacitance of the floor, the return water, and the supply water, respectively, while R_f, R_{pf}, R_{rs} represent the equivalent thermal resistance of the interfaces zone-floor, floor-pipe, and return-supply, respectively. It is also assumed that the water-closed circuit does not suffer from any heat loss and therefore transfers entirely the HVAC heat from the source toward the zone.

The electrical energy consumption of the residential HVAC can therefore be derived as:

$$P_{el} = \frac{u_{hvac}\, P_{hvac}^{cap}}{\eta} \tag{1.16}$$

where η refers to the efficiency of the system: $\eta = COP$ for HP and $0 < \eta \leq 1$ for electrical heater.

An important distinction between small residential equipment and large commercial installation is the nature of the HVAC control signal u_{hvac}. While large commercial systems consider a continuous range of values for the control signal, smaller residential and commercial installations only permit on/off operations on the HVAC, leading to:

$$u_{hvac} = \begin{cases} 1 & \text{on state} \\ 0 & \text{off state} \end{cases} \tag{1.17}$$

The COP represents a key element in modeling a HP, and multiple approaches exist to assess its value depending on ambient conditions [23], both for commercial and residential loads. The simplest solution consists in assuming a constant COP value, regardless of the surrounding temperature conditions. However, this method under- or over-estimates the instantaneous COP when both outdoor and indoor temperatures vary over time. Moreover, the relative humidity also impacts the thermal performance of the HP. The following linear model therefore better estimates instantaneous COP:

$$\text{cop}[t] = \text{cop}_0 + \alpha T_i[t] + \beta T_o[t] \tag{1.18}$$

where cop_0, α, and β are coefficients derived from manufacturer datasheets, and T_i, T_o are wetbulb indoor and outdoor temperatures ($°C$), respectively. Practically, wetbulb temperatures can be assessed from dry air temperature, relative humidity, and air pressure [14]. Advanced constraints on the HP compressors and other specific parts of the HVAC system have been modeled in the literature. However, the chapters relying on building thermal models do not consider these advanced modeling, for reasons explained.

Hot water tanks

Apart from the HVAC system, EWH and HP for domestic-use water heating represent important thermal loads. A hot water tank provides hot water to its user and replaces the used hot water by incoming cold water. The incoming cold water is mixed with the stored hot water and subsequently heated to ensure that the output water remains hot enough. A typical EWH contains two resistive elements: a lower heating element and an upper heating element, both connected to a controller that acts on their electrical power. Outlet water is usually taken from the top of the tank, while incoming water is pushed into its bottom.

Extensive work has been performed to model water temperature evolution within the tank with a smart-grid context [24, 25, 26, 27, 28, 29, 12]. Models presented in [28, 29, 12] slice the water tank into N_l multiple layers, each having uniform temperature and interacting with the adjacent layers. These models are referred to as the stratified model and compute the change in temperature of layer k as follows [29]:

$$\dot{T}_k(t) = \dot{T}_k(t)\Big|_{diff} + \dot{T}_k(t)\Big|_{draw} + \dot{T}_k(t)\Big|_{heat} + \dot{T}_k(t)\Big|_{loss} \tag{1.19}$$

where:

- $\dot{T}_k(t)\Big|_{diff} = \frac{u_w}{\rho c_w d^2}(T_{k-1}(t)+T_{k+1}(t)-2T_k(t))$ represents the change in layer temperature due to heat diffusion. u_w is the heat conductivity of water $(K/W.m)$, ρ is the water density (kg/m^3), c_w is the specific heat capacity of water $(J/K.kg)$, T_{k-1} is the temperature of the layer below, T_{k+1} the temperature of the layer above. This equation is modified for the top and bottom layer.

- $\dot{T}_k(t)\Big|_{draw} = \frac{\dot{m}(t)}{m_k}(T_{k-1}(t) - T_k(t))$ represents the change in layer temperature due to drawing of hot water, when occupants use hot water. $\dot{m}$ is the water draw rate and m_k is the mass of water in the k^{th} layer. This term is null when no hot water is used. The bottom layer replaces $T_{k-1}(t)$ by $T_{in}(t)$, the incoming water temperature.

- $\dot{T}_k(t)\Big|_{heat} = \frac{1}{c_w m_k}\frac{Q_h(t)}{N_h}$ represents the change in layer temperature due to electric heating. $Q_h(t)$ represents the thermal heat added to the system, and N_h the amount of layer in contact with the electrical heater. This term is null for layers that are not in contact with any heating element.

- $\dot{T}_k(t)\Big|_{loss} = \frac{A_k U}{m_k u_w}(T_o(t)-T_k(t))$ represents the change in layer temperature due to losses with tank environment. U is the thermal conductivity of the tank structure, $T_o(t)$ is the ambient temperature, and A_k is the tank surface of contact of layer k. The latter is higher for top and bottom layers.

In contrast to the stratified one, models in [24, 25, 26, 27] consider the water temperature to be homogeneous throughout the whole tank. This leads to the following simpler thermal equation:

$$C_w \dot{T_w}(t) = Q_h(t) - \dot{m}(t) c_w (T_w(t) - T_{in}(t)) - AU(T_w(t) - T_o(t)) \tag{1.20}$$

where T_w is the tank water temperature, A is the total tank surface area, C_w is the thermal capacity of the total water mass, and all the other parameters are the same as in the stratified model. Eq. (1.20) can be solved to get the temperature of the hot water after dt seconds:

$$T_w(t+dt) = T_w(t) e^{-\frac{dt}{R^* C}} + (1 - e^{-\frac{dt}{R^* C}})(\frac{AU}{R^*} T_o(t) + c_w \dot{m} R^* T_{in} + R^* Q(t)) \tag{1.21}$$

where $R^* = (AU + c_w \dot{m})^{-1}$ (K/W) is the equivalent thermal resistance modeling the losses due to both the outside environment temperature and the inlet cold water, when a water draw occurs. This simplified model does not accurately capture the complex dynamics of heat propagation due to differences in temperature within the tank itself when incoming cold water enters by the bottom or when heating elements provide heat to the layers around the center of the tank. Nevertheless, Eq. (1.21) allows to catch the main behavior of the tank and is computationally simpler than solving (1.19) for each layer. Moreover, Eq. (1.19) requires more parameters that may not be available from the manufacturer.

1.1.2 Deferrable loads

Definition 1.1.3. Deferrable load - A deferrable load (or shiftable load) refers to a load whose starting time can be postponed, with respect to the trigger time decided by the user, without affecting her/his comfort.

Practically, the user specifies a minimum starting time $\underline{t}_{d,s}$ and maximum ending time $\overline{t}_{d,e}$ for each deferrable load. An external controller/automation system will then decide an appropriate starting time t_d^*. The power profile $\mathscr{P}_d$ of the deferrable is supposed to be given, leading to the following definition of load power consumption:

$$p_d[h] = \begin{cases} \mathscr{P}_d[h - t_d^*] & \text{if } t_d^* \leq h \leq t_d^* + |\mathscr{P}_d| \\ 0 & \text{otherwise} \end{cases} \tag{1.22}$$

where $|\mathscr{P}_d|$ refers to the duration of the deferrable load. The constraints fixed by the user can therefore be expressed as follows:

$$\underline{t}_{d,s} \leq t_d^* \leq \overline{t}_{d,e} - |\mathscr{P}_d| \tag{1.23}$$

Typical residential deferrable loads include washing machine, washer, and dryer.

1.1.3 Non-controllable loads

Definition 1.1.4. Non-controllable load - A non-controllable load (also referred to as user-driven load or uncontrollable load) is a load whose power consumption is driven by the user, and no automation system/local controller may influence its state.

Most of the non-controllable loads are primarily made to directly interact with the user, in order to provide a specific service. Any intervention of an automation system would therefore disturb the main purpose of such loads. The common non-controllable loads encountered in residential buildings are: computer and laptop [3], entertainment load (e.g., sound system, television, video game console), kitchen appliances, light, and other miscellaneous loads (e.g., vacuum cleaner, chargers).

We present here a mathematical framework to model the non-controllable load power consumption. A non-controllable load power profile $\mathscr{P}_{nc}$ is made of the following three components:

- A set $\mathscr{M}_{nc} = \{m_i\}$ of modes m_i, describing the possible power consumption level that the user-driven load can take. The basic information are considered given by the manufacturer, or externally identified, as:
$$m_i = \begin{cases} \text{label} & \text{(text)} \\ (\underline{P}_i, \overline{P}^i) & \text{Watt} \end{cases} \tag{1.24}$$
where $\underline{P}_i$ and $\overline{P}_i$ represent the minimum and maximum power consumption, respectively.

- A set $\mathscr{S}_{nc} = \{s_{ij}\}$ of sequences s_{ij}, representing a succession of modes the load experiences in the state $i \to j$. If $i = j$, then s_{ij} holds the information about the modes that can be activated in that steady-state [4], via the transition matrix A_i. The elements $A_{i,jk}$ matrix A_i define the probability of transitioning from mode m_j to mode m_k. Otherwise, when $i \neq j$, s_{ij} describes the modes encountered to transition from state i to state j. This can be summarized as:
$$s_{ij} = \begin{cases} A_i & i = j \\ ((m_{ij,1}, p_{ij,1}, d_{ij,1}), \ldots, (m_{ij,n}, p_{ij,n}, d_{ij,n})) & i \neq j \end{cases} \tag{1.25}$$
where a triplet $(m_{ij,k}, p_{ij,k}, d_{ij,k})$ contains the k^{th} mode $m_{ij,k}$ in the transition sequence and its corresponding probability $p_{ij,k}$ and duration $d_{ij,k}$.

- An ordered list $\mathscr{A}_{nc} = \{a_k\}$ of activities a_k that define instances of load utilization by its user:
$$a_k = (s_k, t_k, d_k, p_k, param_k) \tag{1.26}$$

---

[3] Laptop systems generally draw power from a portable battery that is connected to the outlet. Even though this could constitute a form of storage, they are considered to be non-controllable in this thesis.

[4] Common steady-state sequences are on, off, and standby.

where s_k is a sequence defined by Eq. (1.25), t_k is the statistical distribution of starting time (seconds), d_k is the statistical distribution of activity duration (seconds), p_k is the probability that the activity happens, and $param_k$ are additional parameters to the activity.

Both $\mathscr{M}$ and $\mathscr{S}$ define the load power consumption independently of its use, while the list $\mathscr{A}$ describes the way the user acts on it. A load profile $\mathscr{P}_{nc}$ is therefore linked to a specific load and user, and this mathematical formulation can directly be leveraged to model and predict the uncontrollable load power consumption. Chapter 3 will present techniques to extract load profiles $\mathscr{P}_{nc}$ from historical data, by identifying the various parameters of the three sets $\mathscr{M}_{nc}$, $\mathscr{S}_{nc}$, and $\mathscr{A}_{nc}$. More details about non-controllable load modeling can therefore be found in that chapter.

1.1.4 Energy storage systems

Definition 1.1.5. Energy storage system - An Energy Storage System (ESS) is an entity capable of absorbing electrical energy, storing it in any form, and delivering back electrical energy later on. Charging and discharging powers are considered to be positive (or null) and negative (or null), respectively.

An ESS is an important component to deal with the mismatch between production and consumption power. Unlike air/mass thermal storage [5] presented in Section 1.1.1, an ESS stores electrical energy in a certain form and part of it can later be retrieved as electrical energy. The most common ESS include:

- Electrochemical battery (rechargeable battery): electrical energy is used to move ions from one electrode to another (charging), and the opposite operation generates electricity (discharging). Lead-acid batteries have a huge share of the automotive market due to their high power-to-weight ratio. Lithium-Ion (Li-Ion) batteries hold a higher energy-to-weight and a slower self-discharge and are used for portable electronics and electric vehicles. The typical round-trip efficiency is 80-90% for Li-Ion batteries, against 70-75% for lead-acid.
- Hydrogen fuel cell: the conversion of chemical energy contained in the hydrogen and oxygen is performed through a pair of redox reactions to generate electricity (discharge). Reversely, in charging mode, electricity is used to perform the electrolysis of water produced during the discharge, leading to hydrogen. Typical round-trip efficiency is in the range 30-50%.
- Pumped hydroelectric energy storage: electrical energy is transformed into gravitational potential energy of water by pumping it to a higher altitude (charging). Alternatively,

---

[5] Ice storage is also found in large buildings, acting as an efficient means for cooling; it is not covered in this thesis.

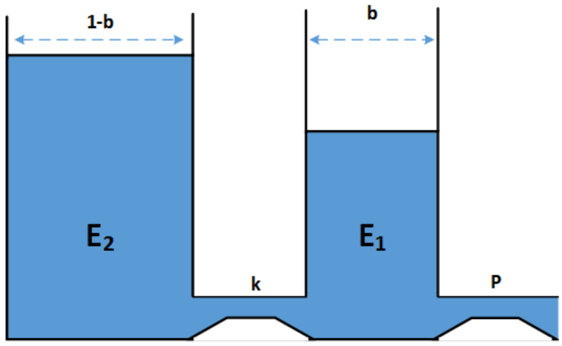


Figure 1.3 – Battery KiBaM model

electricity can be generated by lowering that potential energy (discharging). Typical round-trip efficiency is in the range 70-80%.

- Flywheel system: electrical energy is converted into rotational energy by accelerating a rotor enclosed in vacuum at high-speed (charging). The deceleration of the rotor generates electricity (discharging). Latest technology using magnetic bearings and carbon-fiber composite rotor allows a round-trip efficiency of 85%.

The simplest model used in the literature describes the State-of-Charge (SoC) evolution in a first-order integrative equation [6, 30, 31]:

$$x_b[h+1] = (\alpha\ x_b[h] + \eta_c\ u_c[h] + \frac{1}{\eta_d}\ u_d[h])\ dt \tag{1.27}$$

$$u_b[h] = u_c[h] + u_d[h] \tag{1.28}$$

where $x_b[h]$ represents the SoC of the battery (kWh) at time h, $u_c[h]$ and $u_d[h]$, the charging and discharging power (kW) at time h, respectively, α_b (h^{-1}), the self-discharge coefficient, η_c and η_d, the charging and discharging efficiency, respectively.

Battery charging/discharging power and SoC are physically bound:

$$\underline{C} \leq x_b[h] \leq \overline{C} \tag{1.29}$$

$$0 \leq u_c[h] \leq \overline{P} \tag{1.30}$$

$$0 \geq u_d[h] \geq \underline{P} \tag{1.31}$$

where $\underline{C}$ and $\overline{C}$ represent the minimum and maximum SoC (kWh), $\underline{P}$ and $\overline{P}$ the maximum charging and minimum discharging power (kW), respectively.

Advanced modeling of ESS

The Kinetic Battery Model (KiBaM) model [32, 33] provides further equations to better catch advanced phenomena happening in batteries, through chemical kinetics processes. It uses two wells to represent the total storage capabilities C: the available-charge capacity bC and the bound-charge capacity $(1-b)C$. As depicted in Fig. 1.3, the rate at which the charges flow between the two wells depends on the conductance k (s^{-1}) and the height difference of the two wells. The discharging/charging power P influences only the charges $E1$ in the available-charge well. Authors in [32] further proved that the KiBaM model is actually the first-order simplification of the continuous diffusion model.

From the KiBaM basic equations, the discrete expressions of the two wells of energy can be derived as [33]:

$$E1[h+1] = E1[h] \cdot e^{-k \cdot dt} + C \cdot b \cdot (1 - e^{-k \cdot dt}) + \eta \cdot P[h] \cdot \frac{(1 - e^{-k \cdot dt}) + b \cdot (k \cdot dt + e^{-k \cdot dt} - 1)}{k} \tag{1.32}$$

$$E2[h+1] = E2[h] \cdot e^{-k \cdot dt} + C \cdot (1-b) \cdot (1 - e^{-k \cdot dt}) + \eta \cdot P[h] \cdot \frac{(1-b) \cdot (k \cdot dt + e^{-k \cdot dt} - 1)}{k} \tag{1.33}$$

$$E[h] = E1[h] + E2[h] \tag{1.34}$$

where dt is the discretization period and η is either the charging efficiency η_c or discharging efficiency $\frac{1}{\eta_d}$, depending on the sign of $P[h]$.

From the KiBaM model the maximum charging and discharging powers can be derived that depend on the current state of the system:

$$P^{disch}_{max}[h] = \frac{k \cdot E1[h] \cdot e^{-k \cdot dt} + E[t] \cdot k \cdot b \cdot (1 - e^{-k \cdot dt})}{1 - e^{-k \cdot dt} + b \cdot (k \cdot dt - 1 + e^{-k \cdot dt})} \tag{1.35}$$

$$P^{ch}_{max}[h] = \frac{-k \cdot b \cdot C + k \cdot E1[h] \cdot e^{-k \cdot dt} + E[t] \cdot k \cdot b \cdot (1 - e^{-k \cdot dt})}{1 - e^{-k \cdot dt} + b \cdot (k \cdot dt - 1 + e^{-k \cdot dt})} \tag{1.36}$$

Equations (1.32) and (1.33), together with equations (1.35) and (1.36), define the energy-oriented KiBaM model of the building battery.

Lead-acid battery represents the main target of the model. When analyzing the current and voltage, the model cannot be applied to Li-Ion batteries. However, the KiBaM model is valid for that kind of battery if one looks solely at the energy exchange occurring in the system [33].

Electric vehicles

EVs rely on a large electrochemical rechargeable battery to power the mechanical system. They require a substantial amount of electrical power to charge the battery in a relative short amount of time and must therefore be appropriately coordinated. It is assumed that the EV battery can be used in charging and discharging mode, when plugged-in to the building. In addition to the dynamic equations modeling the SoC, the following constraints are added:

$$u_b[h] = 0 \qquad \forall h \in \mathcal{T}_{out} \tag{1.37}$$

$$x_b[t_{b,a}] = C_{b,a} \tag{1.38}$$

Eq. (1.37) prevents the battery from being used when it is not physically present, in which $\mathcal{T}_{out}$ represents the time periods when the EV is not available at the building site. When the EV arrives at the building charging facility, the corresponding SoC is modelled in Eq. (1.38) by forcing x_b to the arrival SoC $C_{b,a}$ at the arrival instant $t_{b,a}$. Models and statistical distributions like the ones used in [34] can statistically represent arrival time $t_{b,a}$, arrival SoC $C_{b,a}$, and disconnected periods $\mathcal{T}_{out}$.

1.1.5 Local renewable energy production

Definition 1.1.6. Local renewable energy system - A local renewable energy system refers to a small-scale system converting a source of renewable energy into electrical energy.

Local RES sources commonly encompass solar and wind at the building/community scale [6]. In this thesis, we focused mainly on PV modeling from incoming solar irradiation and ambient temperature.

Photovoltaic energy production

Electrical energy generated from solar radiations is the most common source of renewable energy encountered at the building level. It can easily be harnessed on rooftops of dwellings or commercial buildings. This energy source is also commonly exploited in electrical grids and microgrids, concentrated in one point to inject power in the grid infrastructure.

The PV cell is the basic element of the PV system. Its crystalline silicon material reacts to high-energy photons coming from solar radiation, hence allowing electrons to flow in the semi-conductor and through an external circuit. The combined serial and parallel connection of multiple PV cells produces a PV array (or module) that can practically be used in the aforementioned electrical systems.

The PV array is a two-terminal active electrical component whose output voltage V dictates the output current I, depending on the connected load and incoming radiation intensity. The corresponding non-linear I-V curves define the characteristics of the PV array, from which the instantaneous output DC power of the module can be derived. At any instant, it is suitable to always harness as much power as possible from the PV system, defined as the maximum product between I and V. The Maximum Power Point Tracker (MPPT) controller aims to fulfill this task, and ensures an optimal couple (I,V) against any input/output conditions. Most of the MPPT systems are integrated within the PV DC/AC inverter.

PV array modeling has received much attention, leading to the establishment of the well-

---

[6] Hydraulic, geothermal, and biomass are not considered as they generally require a larger plant to exploit them.

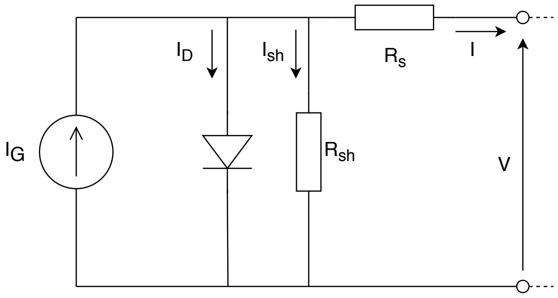


Figure 1.4 – PV one-diode model: equivalent circuit

known one-diode and two-diode models [35, 36, 37]. Fig. 1.4 depicts the one-diode model equivalent electrical circuit, involving a current source driven by solar radiation, a diode, and two resistances R_{sh} (Ω) and R_s (Ω). The diode models the semi-conductor nature of the module, and the resistances account for the losses:

$$I = I_G - I_0(e^{-\frac{q(V+R_s \cdot I)}{nkT_c}} - 1) - \frac{V + R_s \cdot I}{R_{sh}} \tag{1.39}$$

where I_G (A) is the solar-driven current, I_0 (A), the reverse saturation current, q (C), the charge of the electron, n, the ideality factor, k (J/K), the Boltzmann constant, and T_c the cell temperature. The use of well established (I,V) curves by the PV array manufacturer enables the identification of the parameters in Eq. (1.39).

As one looks solely at the output power model, a simplified model can be derived directly from the knowledge of MPPT maximum power at nominal conditions [36]:

$$p_g[h] = n_{cells}\, P_n\, \frac{G[h]}{G_n}(1 + \alpha_i\, \Delta T[h])(1 + \alpha_u\, \Delta T[h]) \tag{1.40}$$

where $\Delta T[h]$ is the cell temperature difference from standard condition, G_n is the nominal radiation (W/m^2), α_i and α_u are the temperature sensitivity ($\%/°C$) of the PV output current and voltage, respectively, n_{cells} is the number of modules, and P_n (W) is the nominal MPPT output power.

Such a model neglects the non-linearities of Eq. (1.39) as well as losses. Yet, it is not accurate enough to be used in a MPPT controller or module characterization; it is suitable for the purpose of simulating building PV energy production at the system level.

1.2 Existing Building Simulation Tools

Modeling and simulating (smart-)buildings in an appropriate way is of paramount importance to characterize and analyze the complex relationships between its stakeholders. Many tools exist to simulate energy in buildings, notably EnergyPlus [38], DOE-2 [39], TRNSYS [40], and ESP-r [41]. Primarily tailored to accurately simulate the complex thermodynamics of buildings,

they constitute many engines to model an individual building energy consumption analysis. EnergyPlus is an open-source whole building simulation program developed by a collaboration between National Renewable Energy Laboratory (NREL) and Department of Energy (DOE) national laboratories. It takes input from the user files on the building's geometry, construction materials, user actions and schedules on internal loads, weather, and other thermal characteristics. Based on these data, the tool then calculates the necessary heating and cooling loads to ensure thermal comfort as well as other useful design information. The software is based on many advanced models, such as heat balance of radiant and convective effects, detailed HVAC system, layer-by-layer heat balances in windows, and even performs atmospheric pollutant calculations. It produces a set of proprietary files that an expert can use for a complete building energy consumption analysis. Similar to EnergyPlus, the software DOE-2 developed at Lawrence Berkeley National Laboratory (LBNL) is the core kernel for other widely used user-friendly software like Home Energy Saver [42] and MULTEA [43].
ESP-r is an open-source performance modeling software for energy in buildings, created by the University of Strathclyde, widely used in consultancy and research. It solves conservation equations on finite volumes discretizing the whole system and follows a holistic approach to precisely model the building (heat, air, moisture, light, electrical power flows, ...). While it offers a range of interesting features, the tool requires an expert user for complex tasks.
TRNSYS is a commercial, flexible software environment designed at the University of Wisconsin to simulate transient systems. Thermal and electrical energy systems being the main target, the applications can go beyond thereof. The core is coded in Fortran, but the particularity of the tool lies in the modular components that can be added (typically in C, C++). Many plug-ins gravitate around the TRNSYS kernel, such as a Graphical User Interface and a module for 3D modeling.

Despite their wide acceptance and proven accuracy in single building energy simulation, these tools are not directly suitable for co-simulation in a building-to-grid simulation context. They were not primarily designed to connect to an external controller or energy management system; instead, they were built as a monolithic software tailored solely for building energy analysis. Hence, the research community developed modules like BCVTB [44], MLE+ [45], and OpenBuild [46] that allow to jointly use advanced simulation tools like EnergyPlus and user-friendly modeling languages such as MATLAB or Modelica. Authors in [47] also developed a co-simulation interface to bridge the TRNSYS software with Simulink. A general-purpose mathematical environment like Matlab offers librairies like Simscape Block [48], a user-friendly hierarchical set of blocks to simulate the building thermal behavior. Likewise, Modelica language comes with an open-source suite of tools and packages like [49] that enable advanced continuous building modeling. Compared to the aforementioned single-purpose solutions, environments like Matlab and Modelica are already widely spread in the engineering field, with their users familiar to the scripting and modeling languages. In addition, the well-documented libraries and the active community promote their use and adoption.

Multi-Agent Simulation (MAS) has recently been applied to smart-buildings to simulate complex systems through the decentralized simulation of their individual actors [50]. Authors in [51] modeled the building as a set of distributed devices that work together to serve the purpose of its users. They developed a multiagent system called CASA to simulate the complex relationships between loads, sensors, actuators, user actions, and user comfort/security. Similarly, the UMASS project [52] leverages MAS to simulate coordination of entities and control logic in an intelligent home. In [53], multiple agents implement equivalent thermal RC model, PV generation, and battery storage taking into account the uncertainty on electricity prices. The Nottingham Multi-Agent Stochastic Simulation platform (No-MASS) was presented in [54], along with a thorough review of MAS applied to occupant behavior modeling. Using Functional Mock-up Interface (FMI), the tool provides EnergyPlus with user behavior schedules that result from internal agent-based simulation. Beyond MAS-based solutions, it is also worth citing projects and works like CASS [55], HomeSim [56], and HybridSim [57], allowing innovative architecture to implement offline building simulation.

2 An Integrated Simulator for Smart-Buildings

Simulation of building energy consumption represents the cornerstone in building-to-grid research. Therefore, a suitable building simulation tool is required to appropriately capture their main features and model their flexibility potential, with a reasonable complexity and computational time. As Building Data Management Systems (BDMS) become increasingly popular, leveraging their infrastructure leads to the design of a new kind of building simulation environment.

The main **highlights** and **contributions** of this chapter are:

- The building Virtualization Engine: a modular simulation and emulation tool closely integrated within the widespread BDMS. Distributed micro-threads making up the core of the engine interact with each other to reproduce physical building behavior, by implementing common building component models. The resulting tool is open-source for the research community.
- A detailed realistic test case of an instance of the Virtualization Engine, based on the Swiss Minergie standard for energy efficient buildings. It emphasizes the modularity of the tool through the simulation of loads, user behavior, storage, and generation.

Related **publications**:

[58] G. Lilis, O. Van Cutsem, and M. Kayal, "Building Virtualization Engine: a Novel Approach Based on Discrete Event Simulation," *2nd International Conference on Event-Based Control, Communication, and Signal Processing*, 2016.

[59] G. Lilis*, O. Van Cutsem*, and M. Kayal, "A High-Speed Integrated building emulation engine based on discrete event simulation," *Journal of Systems Architecture*, vol. 92, pp. 53-65, 2018
**the authors contributed equally.*

2.1 Introduction and Motivation

Carrying out offline simulations of building energy consumption is a time and resource consuming task. It requires expert knowledge, to model the building geometry, wall & window properties, HVAC system, loads, occupant behavior, and to identify the corresponding parameters. Many smart-buildings host a BDMS, a software that collects and dispatches data among all the smart-devices in the building to expose them to high-level applications [1, 60]. Building metadata and historical data collected over time is therefore readily available for a subsequent identification of model parameters used in classic offline simulations. The data extraction, processing & analysis, and formatting for a later offline simulation is an expensive process. The external simulator generally also requires advanced expert knowledge and/or a license.

In this chapter, we present an innovative simulation engine that seamlessly plugs into existing BDMS software, hence avoiding the aforementioned drawbacks of off-line simulations. The developed tool, called Virtualization Engine (*vEngine*), consists of a module that replicates building components behavior that are not physically present, through the use of a set of distributed (micro-)processes. Completely agnostic to the virtual nature of the simulated virtual entities, the BDMS handles their data and acts on it the same way it does on their physical counterparts. Two different purposes of the *vEngine* exist: emulation or simulation of building components. In emulation mode, the simulated entities run in real-time along with the physical infrastructure it may interact with. A full description of the emulation mode mechanisms can be found in [59, 13]. In simulation mode - the focus of this chapter -, the simulated entities are decoupled from the real infrastructure. The tool has been enhanced towards an autonomous simulation platform, that takes advantage of building metadata in existing BDMS and the connection of existing advanced applicationq such as building EMS. Freely accessible to the community, the corresponding open-source code can be found at [61]. The main advantages/innovations of the tool can be summarized as follows:

- BDMS integration: the simulation tool directly integrates the BDMS, thus removing the need for an external tool. As described above, these tools are generally proprietary, non-modular, monolithic, or a combination of these features. On the contrary, the close connection with the BDMS allows to use local entity metadata and timeseries data and to better characterize occupant behavior.

- Light and open source: a light and modular engine architecture was sought. The resulting tool is light, ideal for low-resources hardware, and its modularity eases the update or addition of new models. Open source, it is entirely developed in Python that enjoys a large community. This removes the burden of learning a new language solely for the use of the simulation.

- Building-to-Grid research: the engine and its models do not intend to compete with state-of-the-art monolithic energy simulators like EnergyPlus. Instead, it suits behavioral analysis and multi-buildings simulations in which lower accuracy is acceptable,

> while being fast thanks to its light distributed architecture. Yet, the user is free to develop more complex models for advanced building simulation.

Sharing similar architecture and purpose as the *vEngine*, authors in [62] presented a service-oriented simulation architecture for intelligent building management. Their distinctive feature is the modularity of the open platform which allows different users to participate in and contribute to the development. The service-like architecture permits an effective and simplified introduction of new services without entangling the developers with the complexity of the BDMS. Their simulation design integrates both real and simulated devices like the architecture presented in this paper. However, they only do so at the web service level. In addition, that work does not study the simulation models but solely defines the service-oriented framework, leaving the model design to the developers. On the contrary, this work proposes an optimized, yet scalable and expandable solution for integrating real devices along with simulated ones at the network, instead of the web layer. Additionally, it not only defines the models' architecture but also implements and validates a comprehensive list of models through a realistic test case.

The rest of this chapter is organized as follow. Section 2.2 presents the innovative simulation engine developed in this thesis, along with the various mechanisms making up its logic. As an illustration, Section 2.3 details a test case based of the Swiss Minergie Standard, highlighting how the tool can be used in practice. Finally, Section 2.4 concludes the chapter and discusses further applications of the engine.

2.2 Virtualization Engine: an Integrated Building Simulator

The overall architecture of the developed *vEngine* is depicted in Fig. 2.1. Seamlessly connected to the BDMS, the engine acts as a virtual middleware by emulating building components behavior, like their physical counterparts. Two different parts make up the engine, namely the Virtual Middleware (*vMid*) and the pool of Virtual Entities (*vEntities*). On the one hand, the *vMid* handles the startup of the building components simulation, ensures their proper behavior and intercommunication, and connects it to the existing BDMS. It plays the same role as a physical middleware, apart from the virtual nature of the entities it communicates with. On the other hand, the pool of *vEntities* is a set of distributed elements that individually implement specific models of building components. Their combined actions lead to the simulation of a virtual part added to the existing building or the whole building itself.

Generally speaking, a middleware can be seen as a delegate of the underlying hardware (e.g. measurement sensors, actuators), bridging low level components with higher level software and services. In this context, the *vMid* is entirely interoperable with their physical homologues at the BDMS premise. Instead of dealing with a dedicated hardware, the *vMid* collects data from software-emulated entities - the *vEntities* - and transmits them upward to the BDMS or internally to targeted *vEntities*. Alternatively, commands coming from the BDMS are passed to the pool of *vEntities* through the *vMid*. The routing and message forwarding processed is

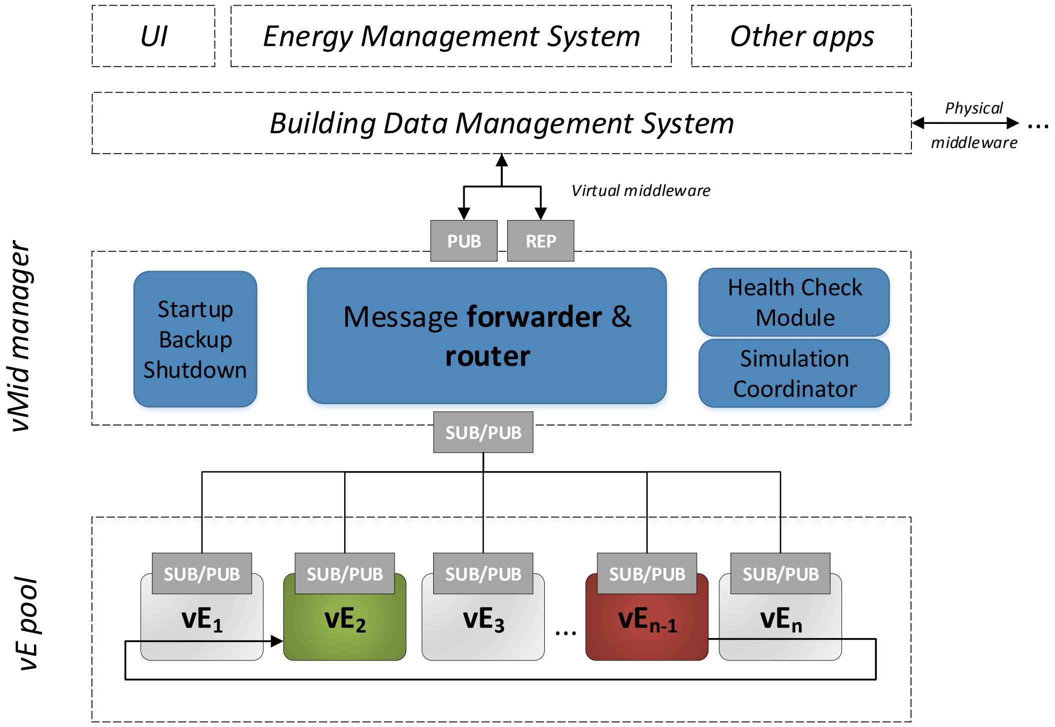


Figure 2.1 – *vEngine* architecture and connection with the BMS

ensured by the sub-module *Message forwarder & router*.

Upon creation of the *vMid*, the *Startup, Backup & Shutdown* module spawns all the *vEntities* linked to that specific *vMid*. The metadata and model parameters necessary for the simulation of building entities are provided by the BDMS database, stored in the usual entry linked to the virtual entity. As the module is fully set up and running, the performance of the pool is continuously monitored by the *Health Check* module, especially important in emulation mode. In simulation mode, the Simulation Coordinator (*vCoordinator*) module ensures the correct orchestration of the distributed model simulations.

From an implementation standpoint, the aforementioned modules and the pool of simulated entities run in separated processes. This allows a better parallelism when the host hardware permits it as well as enhanced scalability. For communication, sets of SUB/PUB sockets are used to enable two-way communication between the *vMid* and the pool of *vEntities*, as well as among the *vMid* sub-modules. The library ZeroMQ practically enables a fast communication [63]. Although perfectly compatible with the external BDMS protocol, an internal messaging protocol has been created for structured communication between the *vMid* and the *vEntities*. The exchanged messages build on a type of multiple-frame packet, called multi-part message, that allows message forwarder and router to be agnostic to the actual content of the message.

Despite sharing some similarity with multi-agent systems, the presented simulator is slightly different. Indeed, single virtual entities are more basic, generally passive and their light implementation facilitates a simple, fast individual model. In agent-based simulation, an agent normally holds intelligence such as rule-based decisions or local optimization algorithms [64]. Moreover, the overall evolution of the system solely depends on the interaction between various agents. They are also more computationally demanding and exchange more messages. On the contrary, the *vEntities* are preprogrammed input-to-output objects, free of methods specific to agents. Instead, an external application leverages their flexibility to control them and the responses of all the entities form the overall simulation.

From a practical aspect, the use of Python as the main language to implement the engine does not add a significant overhead. Indeed, most of the sensitive parts and bottlenecks are coded in C++, meaning that Python solely binds them all. This therefore does not undermine the many advantages of the language, namely its simplicity, dynamic typing, and the large community around it. They lead to a developing time greatly reduced and easier use of the tool.

2.2.1 Virtual entity concept

Definition 2.2.1. Virtual Entity - A virtual entity (*vEntity*) is a basic block of the Virtualization Engine, simulating the behavior of a building's component through the use of a model and data coming from others *vEntities*.

Whilst the *vMid* connects the *vEngine* to the external BDMS and ensures communication with the existing infrastructure, the pool of *vEntities* simulates the behavior of elements that are not physically present. They constitute the fundamental elements of the tool, and are similar to the notion of virtual sensors presented in [55]. The simulation is done by implementing models described in Section 1.1 through a generic process. Fig. 2.2 depicts this generic flowchart that every virtual entity experiences during its lifetime. All going through the same generic flowchart, the *vEntities* differentiate themselves by their input/output signals, equations linking these signals, and their parameters (model or simulation). This flowchart applies specifically to the behavior in simulation mode. Interested reader can further read about the behavior of a *vEntity* in emulation mode in [13].

Right after being spawned, the *vEntity* reads the parameters that were given to it and sets up its model structures. This includes intermediate state variables instantiation, initial value assignation, and parametrization of useful function for a faster online utilization. Then, the *vEntity* waits until it receives a specific signal with simulation context parameters, such as simulation time step or the virtual starting timestamp of the simulation. This completes the steps '0' in Fig.2.2. The entity then reaches the first step '1' of the main infinite loop, that consists in sleeping until reception of a signal of type "simulation sync ." Once received, it evaluates the parameter to decide whether to stay alive. In that case, it processes the data messages that were encapsulated with the "simulation sync" one to update internal state

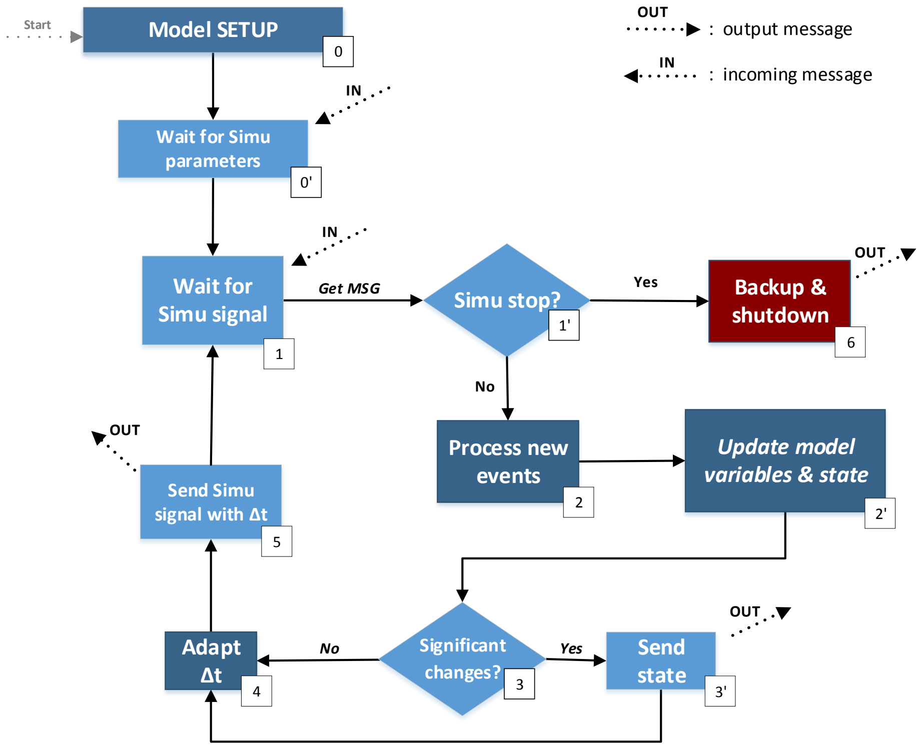


Figure 2.2 – Generic *vEntity* flowchart

variables/parameters. Subsequently, in the second phase of step '2', the entity updates internal state based on its constituting equation, intermediate variables, and incoming data.

Among the internal state variables, some values are meant to be shared with other entities (e.g., power state, stored energy, occupancy binary value). Whether such quantities must be communicated up to the *vMid* is decided at step '3', by comparing output variable values with given thresholds. If a new value must be emitted, a packet is created that encapsulates the data along with the type of information it represents. Two types of data signal can be sent up to the middleware from any virtual entity:

- Sensor value: contains the *vMid* ID as message recipient, the type of the measured quantity (e.g., power, water flow), and the data itself.
- Actuator command: contains the *vEntity* ID it aims to target, the type of actuation it intends to enforce (e.g., switch on appliance, set temperature setpoint, reduce water flow), and actuation parameters. The actuation parameters depend on the type of action (e.g., water flow value or valve state, temperature set point value).

This packet structure is similar to the one sent by physical entities connected to the BDMS.

One exception is the action of the virtual user, who must act on the surrounding environment to simulate reality. If she/he interacts with physical components, the signal sent has to comply with the specific component, but there is no restriction on the virtual recipient. In that case, actuation parameters may include simulation-specific data that would not exist in their physical counterparts. For instance, a non-controllable load does not have physical actuators, but their virtual version requires one to receive on/off commands from the (virtual) user.

Upon completion of sending events to upper-level *vMid* (or after internal state update), the *vEntity* prepares itself to go back to sleep mode. Before doing so, it computes the time interval with the next time instant it will have to wake up (step ’4’). Given the threshold on the output data change for simulating measurement precision, a virtual entity might not need to be waken up too often, hence wasting Central Processing Unit (CPU) resources. It can therefore specify this time interval to the *vCoordinator* when sending the synchronization signal up to the *vMid*, at step ’5’, specifying that it has completed its steps.

Lastly, the *vEntity* might be asked to stop and shutdown (leading to step ’6’). Instead of being externally killed, the responsibility lies on the distributed entities to give them the opportunity to save the internal state that might be useful for a later reincarnation.

Practically, a *vEntity* is an instance as a micro-thread (*uTread*), from the Python library ’greenlet’ [1]. The main advantage of using low-level *uTread* instead of Operating System (OS) threads lies in the faster context switching from one *uTread* to another [65]. As many *vEntities* coexist, this is a key feature to increase simulation speed. However, dealing with *uTread* requires manual management of the proper switching between entities, which is either ensured by the *Health-check* module (see Appendix A.2.1) in emulation mode, or the *vCoordinator* in simulation mode.

Every *vEntity* inherits from the same root (abstract) class, called "vEntity", that implements the aforementioned generic step flow. Some functions are generic, e.g. the definition of output variables, change threshold, and outgoing event generation based on internal value change. Mandatory simulation and model parameters are therefore required for all sub-classes of root one. The specialization of each *vEntity* (sub-classes) to give them their identity is done through the definition of a set of (abstract) functions in the inheriting classes, called in the generic "run" function of the root class. These functions include the initialization step, the incoming message decoding & processing, the internal state update logic (generally implementing equations with a physical meaning), and adaptation of sleeping time.

2.2.2 Simulation coordinator

Beyond the emulation purpose of the engine [59, 13], the tool holds significant values for carrying offline building simulations. When dealing with simulation instead of real-time emulation,

---

[1] A ’greenlet’ is actually a more primitive version of a micro-thread, but is referred to as thereof for the sake of clarity.

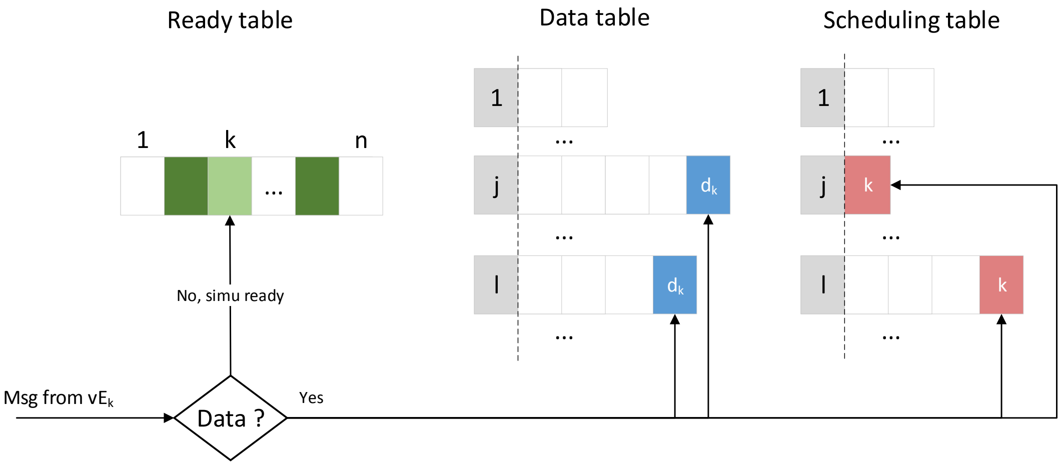


Figure 2.3 – Simulation coordinator logic flow to update internal structures

the notion of time changes as the simulated entities perceive a virtual time step, generally larger than the real one. The *vEngine* must therefore be adapted to integrate simulation capability: this is ensured by the *vCoordinator*, a module integrated into the *vMidManager*. The resulting simulator is closely coupled with the BDMS and can directly leverage building metadata and historical timeseries data. This constitutes a valuable advantage over a traditional offline simulator because it removes the burden of exporting existing data and formatting them to make them compatible with the external simulator.

In practice, the *vCoordinator* communicates with the pool of *uTread* via a specific type of signal to appropriately orchestrate simulation operations and the exchange of data within the pool. These signals have already been mentioned in the previous section, at step '0', '1', and '5', to start the *vEntities*, synchronize their wake-up instants, and allow them to indicate they are ready for the *vCoordinator*, respectively. As the developed classes must be compatible for both emulation and simulation mode to avoid unnecessary double work, one must carefully add the layer of simulation mechanisms at the *vEntity* premise with the help of the *vCoordinator*. For instance, any incoming event containing data would wake-up the *vEntity*. Although desirable in emulation mode, this behavior would wrongly wake-up an entity in simulation mode and make it skip one simulation step, leading to the erroneous data ahead of time for other entities.

All the emitted events must therefore go through the *vCoordinator* that will orchestrate the way they will be dispatched. A specific order must be respected, as most of *vEntities* models depend on data coming from other *vEntities*: the *vCoordinator* will therefore trigger the *vEntities* according to a *dependency* table, in order to collect data from some of them before launching the ones that need it. That table is reconstructed from the dependencies of all the *vEntities* in the pool, an information stored in the BDMS.

From the *dependency* table, the *vCoordinator* will start constructing a *scheduling table*, that maps each *vEntity* to the *vEntities* they still depend on to be triggered. A simulation step therefore starts by triggering all the *vEntities* that do not depend on any other (data provider). Then, the rest of the simulation steps are dictated by the logic depicted on Fig. 2.3. Whenever a new message is received from a *vEntity* k, it first checks the nature on the message:

- Message type "data" means the *vEntity* k just emitted a new event that might interest other entities. This event is stored in the *data* table, repeated for all the *vEntities* that subscribed to it (blue boxes for entities i and j) [2]. In addition, the incoming data will remove the ID k in the *scheduling table* (red boxes). The *vCoordinator* will then trigger all the *vEntities* that have empty remaining dependencies (entity j in the example) in the *scheduling table*.
- Message type "simu ready" means that the *vEntity* k just completed its simulation round and is ready for the next one. This is stored in the *entity ready* table. If the latter reaches its full size n, this means the simulation round is finished: the scheduling table is reset, and the same logic is started over again, with simulated time incremented.

A newly triggered entity receives the special signal of type "start simu-step", along with the data events accumulated in the *data* table, from all the *vEntities* it subscribes to. The *vEntity* will therefore process all these incoming events to update its internal states, and subsequently follows the steps explained in the previous section.

The presented logic does not take into account external actors in the simulation, such as applications leveraging the collected BDMS data to act on the pool of *vEntities*. A typical example involves the building automation system that could schedule deferrable loads at appropriate low prices moment or switch off unused loads in unoccupied rooms. The *vCoordinator* must therefore have a list of these external applications, and will wait for a signal coming from them just before starting a new simulation step in the pool of *vEntities*. Practically, they will be inserted in the *scheduling* table as negative IDs and subscribe to all the actors in the pool. This ensures that external applications wait for the simulation step to be entirely completed and will naturally block the next simulation step until all the external applications have sent their "simu ready" signal, along with their control data signal

Blocking dependency removal

The creation of the *dependency* table from individual *vEntities* dependencies might lead to unwanted deadlock cycles, as illustrated in Fig. 2.4 (left). If the *dependency* table is used

---

[2] A simulated entity might send multiple events or send targeted data to a specific entity, in which case the newly emitted data must be stored either in a unique row in the *data* table or with the corresponding type of data. In the latter case, the receiving entities will filter out the data, as they did not subscribe to that specific type. For instance, a single virtual sensor might send humidity and occupancy data, and another virtual temperature sensor might just be interested in the presence of the user.

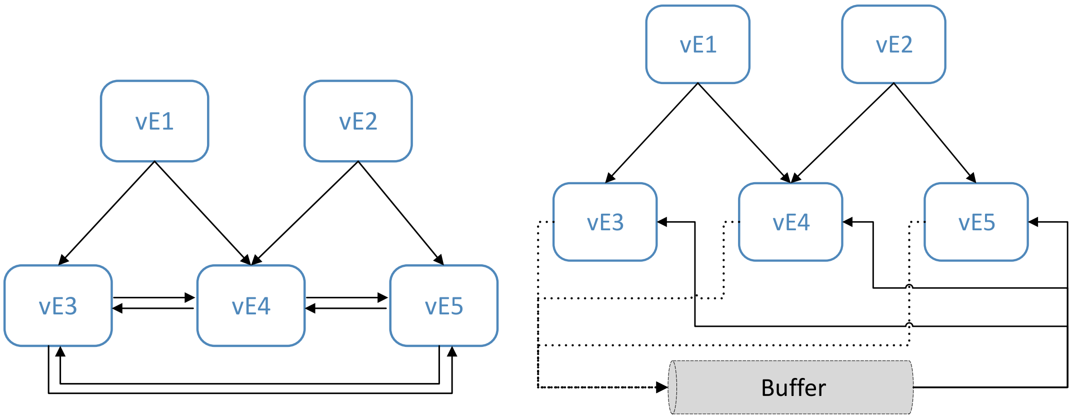


Figure 2.4 – Cycle removal illustration in *vCoordinator* (left) BMS representation of *vEntity* relationship (right) use of a buffer by the coordinator to remove deadlock cycles

directly as such in the algorithm of *vCoordinator* previously described, this would cause the vE_3, vE_4, vE_5 to never be triggered. This phenomena happens in models with mutually coupled equations, such as the ones used to describe the output of temperature sensors of adjacent rooms.

An additional structure called *non-blocking dependency* table is therefore added in the description of each *vEntity* relationships, stored in the BDMS. This table describes the data that an entity subscribed to without blocking its execution. Upon reception of newly emitted data coming from *vEntity* k, the *non-blocking dependency* is used to store the emitted data to an auxiliary buffer, sharing the same structure as the *data* table. When an entity is triggered, it only receives the *blocking* data during its current execution, and will receive the content of the buffer at the next time step. Fig. 2.4 (right) illustrates the use of this buffer to prevent cycles in the *dependency* table.

Event-based simulation mode

Fig. 2.3 and the corresponding description of *vCoordinator* logic depict a generic synchronized simulator in which every simulation participant is woken up at each simulation time step. However, a *vEntity* does not necessarily need to wake up with a specific frequency, as they might have no new event to produce. A too frequent wake-up schedule might therefore lead to too many useless CPU acquisitions and simulation message exchanges, thus slowing down the overall simulation.

To deal with this issue, *vCoordinator* also allows for an event-based simulation, a means by which *vEntities* are just solicited whenever a new event is produced. Following [57], the idea consists of leveraging the Δt information sent by every *vEntity* at step ’4’ and ’5’ of their

Algorithm 1 *vCoordinator*: event-driven targeted scheduling

```
$t \leftarrow 0$
$t^s \leftarrow [t, t, ..., t]$
procedure UPDATE_SCHEDULE(id $k$, interval $\Delta t$)
    $t^s[k] \leftarrow t + \Delta t$
end procedure
procedure PICK_NEXT_VE
    $\mathscr{S}_{passed} \leftarrow \{k\}\ \forall k$ s.t. $t \geq t^s[k]$
    if $passed\_set$ is empty then
        $\mathscr{S}_{trig} \leftarrow \underset{k}{\operatorname{argmin}}(t^s)$
        $t \leftarrow t^s[k]$ for any $k \in \mathscr{S}_{trig}$
        return CONSTRUCT_SCHEDULE_TABLE($\mathscr{S}_{trig}$)
    else
        return CONSTRUCT_SCHEDULE_TABLE($\mathscr{S}_{passed}$)
    end if
end procedure
```

flowchart (cf. Fig. 2.2), at the end of their CPU use. Each *uTread* computes this Δt as the time interval from the current simulation time to the next one when they need to get the CPU again. Such a time interval depends on internal models, the sensitivity threshold to emit output events, and predefined timeseries data. For instance, a virtual ESS of capacity C (kWh) that has a resolution of 1% on its output sensed SoC and a current charging power P (kW) will compute its time interval to the next wake-up time as follows:

$$\Delta t = \frac{3600 * C}{P} * 0.01$$

Yet, a virtual entity can be woken up before their specified dt whenever they receive a message from another entity/BDMS from which they subscribe to. For instance, in the previous example involving the ESS, a new input power might be enforced by the BDMS, in which case the *vEntity* would have to acquire the CPU to process it.

Picking the next entities to run is decided by Algorithm 1, in function $pick\_next\_ve()$. It returns a reduced *scheduling* table that will be used the same way as it was presented through Fig. 2.3. The vector t^s contains the desired scheduled instant of all the entities, which is updated through $update\_schedule()$ whenever an entity k - in the simulation pool or external application - sends a "simu ready" signal along with its Δt. Method $pick\_next\_ve()$ starts by looking at entities that might have a scheduled time inferior to the current time t, in which case they must be scheduled immediately. Even though this case does not arise in a normal situation, this is a robust technique to account for unexpected behaviors. The corresponding entities are stored in $\mathscr{S}_{passed}$. In normal execution involving $\mathscr{S}_{passed}$ to be empty, the entities with the lowest trigger instant are selected and stored in $\mathscr{S}_{trig}$. The next simulation time t instant jumps directly to the scheduling time of this selected set of entities. Finally, the

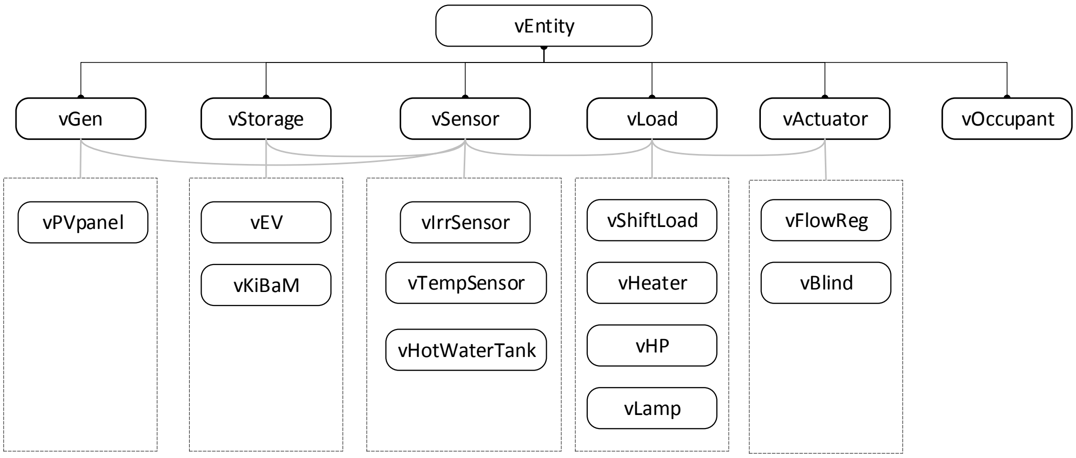


Figure 2.5 – *vEntities* implemented in the current version of the *vEngine*

scheduling table is constructed from the selected $\mathscr{S}_{trig}$, by looking at all the actors interested in events emitted by entities in $\mathscr{S}_{trig}$ and adding them to the table. Alternatively, if $\mathscr{S}_{passed}$ contains values, they must be triggered without updating the current time.

In this configuration, the current simulation time t is therefore driven by the scheduled time t^s of the next picked entities, instead of being periodically incremented. This information must be transmitted to every actor involved in the simulation, whenever they are woken up by a "start simu-step" signal.

2.2.3 Simulated entity models

Many *vEntities* have been developed and can be instantiated to emulate/simulate building components commonly found in a building. They all inherit from the root Python class "vEntity", and other inheritances are found in some sub-classes. Fig. 2.5 depicts the hierarchical classification of the developed virtual entities. Top-level categories have already been described briefly in [59, 13], along with their pseudo-code logic. They represent the core pieces of a generic building emulation/simulation and can be categorized as follows:

- *vSensor* (Virtual Sensor): data measurement constitutes the most fundamental action found in a smart-building. A *vSensor* represents the virtualization of this functionality and generically maps input variables $\mathscr{I}$ to output variables $\mathscr{O}$. By default, a linear mapping $\{x_1, ..., x_n \in \mathscr{I}, y \in \mathscr{O} : f(x_1, ..., x_n) \rightarrow y\}$ can be given to the class *vSensor*. Alternatively, a filename can be provided when spawning a *vSensor* instance to produce specific data to other entities from a static file. To each of the output in $\mathscr{O}$ is mapped a sensitivity threshold that characterizes the sensor resolution. The virtual sensor therefore stores incremental changes and emits event only when the change becomes significant.

- *vActuator* (Virtual Actuator) - actuating allows a remote control of building component state (load power, blind position, temperature setpoint, etc.). A *vActuator* listens specifically to commands that modify an output state. The generic form of the *vActuator* takes as parameters a set of possible commands $\mathscr{I}$ and a set of output variable $\mathscr{O}$. The mapping between $\mathscr{I}$ and $\mathscr{O}$ is left to the user.
- *vLoad* (Virtual Load) - the load is the main power consumption actor in the building, and *vLoad* models a load as a combination of a *vSensor* to emitting power data (W) and a *vActuator* to receive the ON/OFF/DIMMING command (as represented by the gray lines from vLoad in Fig. 2.5). The generic *vLoad* reads as parameter a load profile structures $\mathscr{P}$ defined in Section 1.1.3, to drive the emitted power and the incoming commands.
- *vStorage* (Virtual Energy Storage System) - the second actor that consumes power - or provides it - at the building premise is the ESS, modeled by the *vStorage*. It uses a *vSensor* to sense both charging/discharging power (W) and the SoC (%), and the generic *vStorage* class implements the simple integrative model to update sensed battery SoC.
- *vGenerator* (Virtual Local Energy Generator) - any power generated locally is model through a *vGenerator* that implements a *vSensor* to sense the generated power (W). The generic *vGenerator* class is therefore no more than a generic *vSensor* with a linear mapping between outputs and inputs, but the user can model advanced energy generators.
- *vUser* (Virtual Building Occupant) - the building and its components all serve its occupants, modeled as a *vUser*. The generic *vUser* class collects all the loads/actuators profiles $\mathscr{P}$ it acts on in order to schedule the corresponding activities $\mathscr{A}$. In its simpler form, the *vUser* is therefore an event scheduler, sending ON/OFF signals and dimming values throughout the simulation time.

Beyond these high-level families of entities, many important advanced sub-entities have been further developed to refine building simulation. They are briefly reviewed in this section, and interested readers can get further details in the documented code available at [61].

Advanced virtual sensors and actuator

The following advanced virtual sensors have been developed in the *vEngine*:

- **Virtual zone temperature sensor: *vTempSensor***

  The virtual temperature sensor implements equations of the RC equivalent model presented in Section 1.1.1 in order to output a zone temperature. A zone can either be a room occupied by humans or a hydronic system filled with water. At initialization, two data structures need to be transferred to the *vTempSensor* instance: the RC network with each neighbor node and the geometry of the zone. The latter will be used to compute the equivalent thermal capacity of the zone. At each simulation time step, the virtual entity expects to receive temperature information of connected nodes and the heat power gain injected by other *vEntities* they subscribe to in a non-blocking fashion. Eq. (1.11) updates the zone temperature, given all the stored information.

- **Virtual hot water tank temperature sensor: *vHotWaterTank***

  Similar to *vTempSensor* logic, a *vHotWaterTank* leverages equations described in Section 1.1.1 to update the output temperature of the outlet water of a hot water tank. The implementation considers a simplified model, taking into account dynamic water draw and losses with the environment. It therefore subscribes to event coming from the outside temperature and the actuator governing the output water flow. Like the *vTempSensor*, the dimensions of the tank are given at initialization to compute the equivalent thermal capacity of the tank.

- **Virtual solar irradiance sensor:*vIrrSensor***

  Solar data may not always be available, or the simulation manager might prefer standard solar scenarios. To this end, a virtual irradiance sensor allows to get solar irradiation at the building site. Emitted events are the direct solar irradiance (W/m^2) and diffuse irradiance (W/m^2). They depend on day of the year and time of the day as well as latitude and longitude coordinates. Additional parameters include the ground reflectance coefficient, the altitude, and optional cloud coverage. Used models are based on [66]. An instance of the class "vIrrSensor" therefore solely requires the simulated timestamp and will provide irradiance data to interested entities.

The following advanced virtual actuators have been developed in the *vEngine*:

- **Virtual window blind: *vBlind***

  Blinds are common building components to prevent solar beams from entering and heating the building mass through a window. A virtual blind can be instantiated from the class "vBlind" to model both the glassing effect and the blind effect. Such an instance therefore subscribes to a virtual irradiance sensor - such as a *vIrrSensor* - to subsequently output an equivalent internal heat gain for interested entities - such as a *vTempSensor*. As an actuator, the *vEntity* also listens to incoming commands, either from the *vUser* or the BDMS. A first dimming effect is performed by the glassing of the window as a function of the incidence angle of the solar beams that depends on the same variables/parameters as a *vIrrSensor*. Additional parameters are required to characterize the glassing that influences the reflection/refraction of part of the beam intensity according to [67]. The second step consists in modeling the blind effect. In this simple model, the closing angle is directly related to the amount of heat going through it.

- **Virtual flow regulator: *vFLowReg***

  Many fluids (water) or gases (air) are used in the building infrastructure, either in close or open circuit. An instance of the "vFlowReg" class aims to regulate the flux of the fluid/gas. The *vEntity* expects ON/OFF commands from the BDMS or the *vUser* and a dimming coefficient is typically provided along with the ON command. In addition, the entity can simulate either an abrupt (instantaneous) or linear (delay) transition between states. A typical example is the modeling of hot water use through the instantiation of a *vFLowReg*, dimming the output water flow (l/s) or switching it off.

Advanced virtual loads

Most of the loads in the building aim to provide a given service through the use of electrical power. Beyond entertainment or other services that cannot be easily modeled, the following advanced virtual loads have been developed in the *vEngine*:

- **Virtual deferrable load: *vShiftLoad***

  The definition of a deferrable load was provided in Section 1.1.2. An instance of the class "vShiftLoad" is therefore a load that can be scheduled by an external application through the BDMS. The virtual user first specifies the starting time and the maximum ending time, and the *vEntity* then transmits this information up to the BDMS. It then waits for a command signal that will trigger its start and force it to consume power according to its predefined load profile $\mathscr{P}$.

- **Virtual heat-pump: *vHP***

  Thermal loads emit thermal heat flux (W) in addition to electrical power consumption (W). The class "vHP" implements a general heater or HP model to compute the output heat flux. For virtual HP (heating or cooling), parameters include $\text{cop}_0, \alpha, \beta$ to compute instantaneous COP according to Eq. (1.18):

  $$\text{cop} = \text{cop}_0 + \alpha T_i + \beta T_0 \tag{1.18}$$

  Eq. 1.18 uses the data coming from internal and external temperature sensors to get T_i and T_0, respectively. A simpler model allows for a constant COP, and the electrical heater model requires an efficiency coefficient. The output heat gain can then be emitted to virtual temperature sensors.

- **Virtual light: *vLamp***

  A virtual lamp can be instantiated from the class "vLamp". It directly maps electrical power (W) to an output light intensity (Lux), useful for humans. The *vEntity* only requires mapping from electricity power to light intensity.

Advanced virtual storage and generation

From the *vStorage*, a *vEV* (virtual electric vehicle) can be derived. An advanced KiBaM model is also available in the engine as well as a PV panel model.

- **Virtual KiBaM-based battery: *vKiBaM***

  The KiBaM method to model a secondary battery was presented in Section 1.1.4. Two connected capacities are linked via Eq. (1.32) and Eq. (1.33) to express the overall battery state. The virtual KiBaM battery can be instantiated from the class "vKiBaM", with the parameters k and b. The interest of this model compared to the integrative one lies in the modeling of the maximum charging/discharging power that depends on battery SoC according to Eq. (1.35) and Eq. (1.36). The power command of the BDMS may therefore be impossible to hold, and the sensed power will be an important quantity.

- **Virtual electric vehicle: *vEV***

  A virtual electric vehicle from the class "vEV" simulates a *vStorage*, with an additional layer of user actions. This *vEntity* is idle most of the time (input commands are ignored) and becomes active under the impulsion of a signal coming from the *vUser*. The latter provides starting SoC information at triggering time and may also specify the expected SoC at leaving time t_l. Both information are transmitted up to the BDMS applications for a later control of battery state.

- **Virtual PV array: *vPVpanel***

  Directly implementing Eq. (1.40), the class "vPVpanel" models the generated power of a PV module depending on outside air temperature (at the roof level) and incoming irradiance. It therefore subscribes to the corresponding two *vEntities*.

Advanced virtual user

The virtual user is a special *vEntity*, as it holds a specific intelligence and does not have any sensor nor actuator attached to it. It acts on the spawned *vEntities*, especially the virtual sensors, actuators, and loads, by sending them commands or information.

These entities evolving in the virtual pool have an ID unique to their middleware, much like in physical networks. However, a virtual user, spawned like other entities in a given *vMid*, must know these IDs to be able to target specific entities. Hence, the *vUser* will be provided with the unique IDs of the components it acts on, as being the BDMS key IDs. At the *vMid* premise, a table is used to map this unique BDMS ID to a couple (middleware ID, entity ID) in order to send the command to the right place.

In the current incarnation of the *vEngine*, the virtual user is simply an event scheduler able to decode activity profiles of loads and actuators to subsequently produce commands. These profiles are static and predefined, with optional probability to induce uncertainty. Advanced

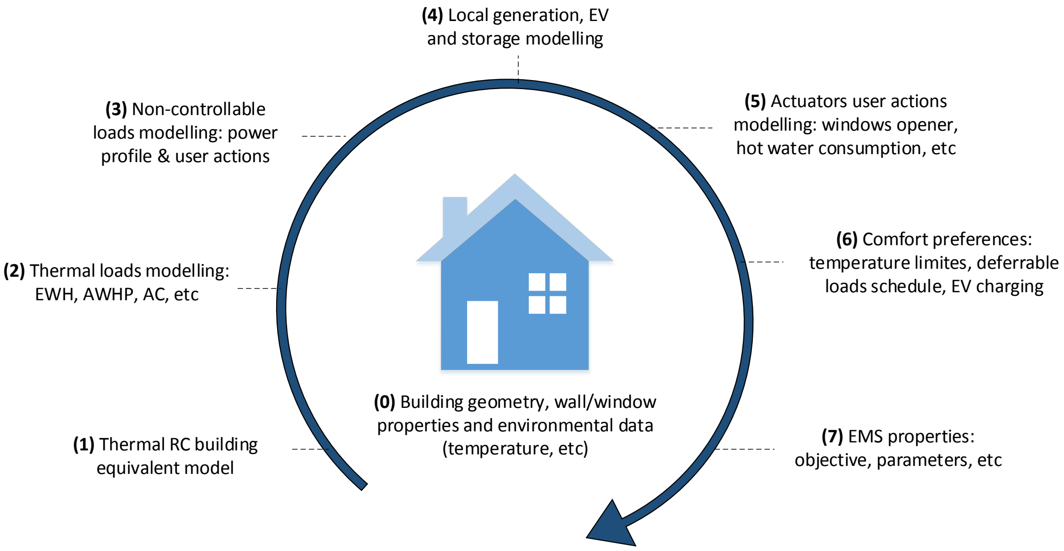

(4) Local generation, EV and storage modelling
(3) Non-controllable loads modelling: power profile & user actions
(5) Actuators user actions modelling: windows opener, hot water consumption, etc
(2) Thermal loads modelling: EWH, AWHP, AC, etc
(6) Comfort preferences: temperature limites, deferrable loads schedule, EV charging
(0) Building geometry, wall/window properties and environmental data (temperature, etc)
(1) Thermal RC building equivalent model
(7) EMS properties: objective, parameters, etc

Figure 2.6 – Step-by-step procedure to derive *vEngine* building model

model can be implemented in the core of this *vUser* to better model the complex behavior of the user that is generally closely coupled with her/his surrounding space and components. The distributed architecture of the tool offers an ideal environment for advanced machine learning techniques to model the virtual user action.

2.3 Case study: a Minergie Building

This section presents a case study to illustrate the use of *vEngine* for building energy simulation. A Swiss building standard called Minergie is used to derive the all-electric building parameters, from thermal envelope to PV sizing, including typical residential loads. The modeling in the BDMS database shows how the various components are practically encoded to represent the building meta-data. Translated into a set of *vEntities*, their interactions simulate the physics of the Minergie building and its components, the behavior of the occupant, and lead to the determination an overall energy consumption profile along with comfort profile.

2.3.1 Minergie building model

Minergie is a voluntary Swiss standard for comfort, efficiency, and value preservation in buildings, created in 1998 [68]. The building constructed under the standard is awarded the corresponding label, ensuring a high-quality envelope, controlled air exchange, and low-energy consumption and maximum use of renewable energies. Various sub-labels can be granted, such as Minergie-P for an enhanced building envelope leading to nearly no use of thermal energy and Minergie-A for the combination of Minergie-P with PV panels, battery, and load management. Compared to other certifications such as LEED, Minergie does not

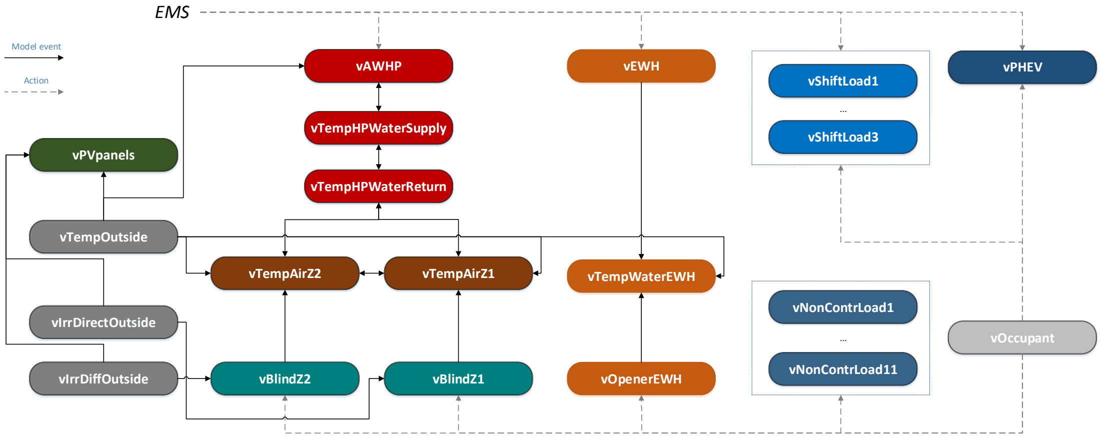


Figure 2.7 – *vEngine* model of Minergie building

work based on a system of score but instead the building must reach requirement thresholds in all key performance indicators. As of 2018, more than 40,000 buildings have already been certified, and an increasing number of new constructions are aiming for it [69].

We considered an all-electric residential dwelling based on Minergie standard, occupied by a typical Swiss family. For simulation purposes, specifications of the standard help derive building model parameters including building envelope, thermal energy use, maximum energy consumption per area, water use, and many others. The step-by-step process to extract building parameters from the standard is illustrated in Fig. 2.6. The process starts by looking at (0) the building physical properties such as spatial geometry, walls and windows material, number of zones, and ambient conditions where the building evolves. From these data, the (1) RC equivalent thermal model of the interconnected zones can be computed. Walls are modelled using two resistances and one capacitance, and a single capacitance represents a zone. Then, (2) the heating system (thermal loads and water pipes) can be designed accordingly. It includes the Air-source to Water Heat-Pump (AWHP) for space heating and EWH for hot water use. The rest of the (3) electrical loads can subsequently be modeled, according to their load profile $\mathscr{P}$ defined in Section 1.1.3. Along with thermal loads, the deferrable loads and uncontrollable loads will lead to (4) the sizing of PV array and battery, if installed. An optional EV can be added at that step. The step (5) models the various actuators that are needed to automatize the resulting smart-building (e.g., blind/window opener, hydronic system) and the actions taken by occupants on building facilities that define how resources are used (e.g. hot water consumption). Comfort and preferences (6) constrain the system on specific zones temperature, starting/ending time of deferrable loads, EV SoC when arriving/leaving, and water temperature. Finally, in case the simulated building is plugged to an (7) EMS, the optimization parameters should be specified (e.g., price of energy, maximum peak demand).

The Minergie case study presents an application of this procedure and shows how the building

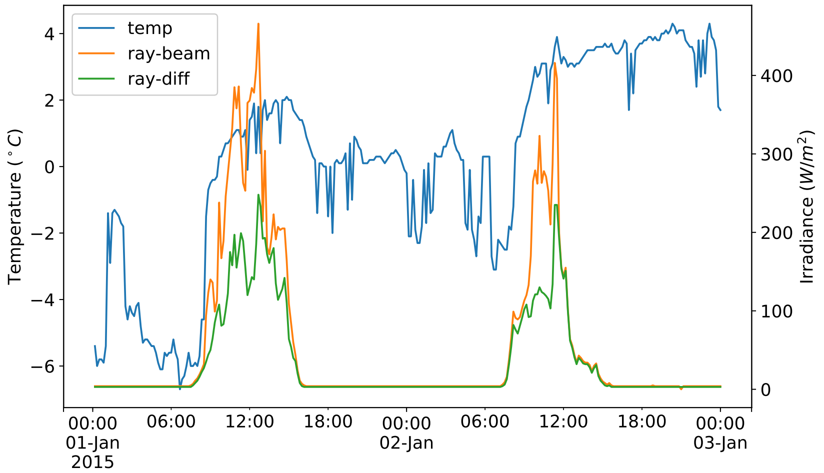


Figure 2.8 – Minergie simulation environmental data

model can be broken down into independent simulated entities that can run in the *vEngine*. The overview of the *vEntities* used in this case study and their interconnection is shown in Fig. 2.7. The rest of this section describes each of these entities, their logic, and the data they exchange.

Environmental data

Environmental data was taken from MeteoSwiss [70] at the weather station in Pully, Vaud, Switzerland (Long 46.51027, Lat 6.66183). The governmental service provides accurate data sampled at a frequency down to 10 minutes. The measures data includes temperatures, solar radiation, relative humidity, wind speed, cloud coverage, and many others. Temperature and solar radiation are shown in Fig. 2.8 for the simulation period (01/01/2015 to 01/03/2015). The ambient air temperature (blue curve, label 'temp') is around $0°C$ on the first day and increases around $2°C$ on the second day, typical of a Swiss winter day. Despite the harsh temperature conditions, the direct (orange curve, label 'ray-beam') solar radiation went up to $400W/m^2$ on the first day and was slightly less intense on the second day.

Three *vEntities* emulate outside environment sensors by reading data from MeteoSwiss historical files and broadcasting corresponding values to the rest of the interested entities. As the simulation goes by, these *vEntities* emit an event containing the data they sense whenever a new read value differs from the current value, given a sensitivity threshold that reflects the resolution of the sensor. The virtual sensor *vTempOutside* emits outside ambient dry air temperature, with a resolution of $0.1°C$. Two other virtual sensors are in charge of reading sun irradiance data: *vIrrDirectOutside* emits direct sun irradiance (W/m^2) measurements and *vIrrDiffOutside* emits diffuse sun irradiance (W/m^2) measurements. They both have a

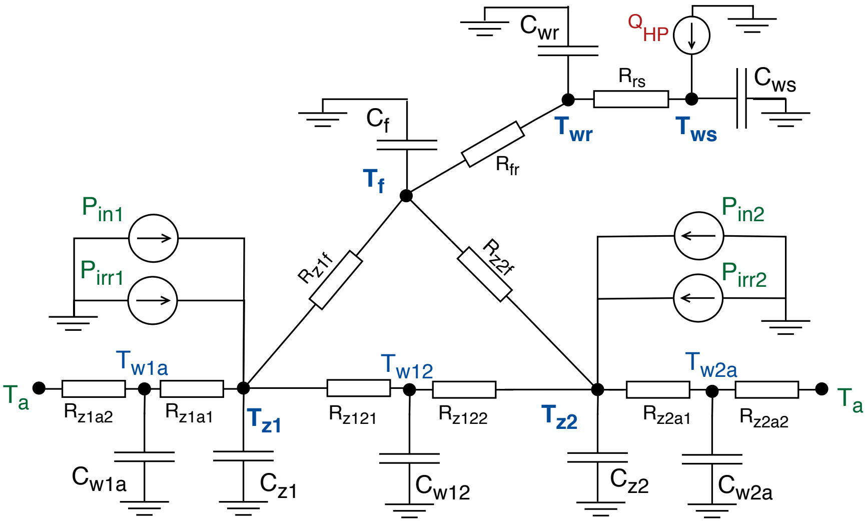


Figure 2.9 – RC model of Minergie building thermal system

resolution of $1W/m^2$. It is assumed that emitted temperature and irradiance values hold for any point in space of the outside environment (roof, ground, wall, window). The run radiation does not affect directly the building interior space: virtual blinds *vBlindZ1* and *vBlindZ2* linked to windows reflect part of them and the rest enters the building zones as disturbance heat.

Thermal model and floor-heating system

The zones the humans occupy in the Minergie building are heated by an underfloor hydronic system. Water in pipes convey the heat from the AWHP source of energy toward the floor where it is slowly released to zone 1 and zone 2. By doing so, the air temperature in each zone can be kept within satisfying bounds, even when facing internal and external disturbance such as sun radiation, internal heat gain, and losses due to the low outside air temperature. Fig. 2.9 schematizes the complete RC thermal model of the building. Appendix A.1.1 provides the simulation values and a complete description how these values have been derived from Minergie specifications.

Heat transfer from the hydronic system to the air zones Z1 and Z2 leverage Eq. (1.12) to Eq. (1.15) presented in Section 1.1.1. The heat pump uses electricity to transfer heat into the supply part of the water pipes, represented by thermal capacitance C_{ws} at temperature T_{ws}. Part of that heat is conducted toward the return water capacitance C_{wr} at temperature T_{wr}, through the resistance R_{rs} that depends on the water flow. The water can then serve to provide energy to the floor, (capacitance C_f at temperature T_f) through resistance R_{fr}, which in turn provides heat to zones Z1 and Z2 through resistances R_{Z1f} and R_{Z2f}, respectively.

The hydronic system and AWHP are simulated in the *vEngine* via three *vEntities*. A *vAWHP* models the HP as a thermal load with COP that depends on both the outside air temperature and supply water temperature, according to Eq. (1.18). The electrical nominal power of the HP is 1.5 kW and COP parameters are provided in Appendix A.1.1. In practice, this *vEntity* subscribes for events coming from the *vTempOutside* and *vTempHPWaterSupply* and receives power setpoint signal from an external controller. Two temperature sensors emulate the return (*vTempHPWaterReturn*) temperature and supply (*vTempHPWaterSupply*) temperature evolution, by using Eq. (1.14) and Eq. (1.15), respectively. These *vEntities* implement the class *vTempSensor* with a precision of $0.1°C$ and include thermal resistances with neighboring interfaces. In addition, *vTempHPWaterReturn* stores the equivalent capacitance of the floor to model its temperature. To simulate the effect of the HP power, *vTempHPWaterSupply* subscribes to events coming from *vAWHP*.

Temperatures of the zones in which occupants lives are virtually measured by *vTempAirZ1* and *vTempAirZ2* for Zone 1 and Zone 2, respectively. These *vEntities* implement the class *vTempSensor*, including thermal resistances with the outside environment, water pipes, and adjacent zone, as well as intermediate capacitance to model heat flow delay in floor tiles and walls. Therefore, each of the two zone subscribes to temperature coming from the virtual temperature sensor of the other zone, the *vTempOutside*, and the *vTempHPWaterReturn*. In addition, incoming heat gain coming from *vBlind* and internal virtual non-controllable loads is received by the virtual zone air temperature sensor (P_{in} and P_{irr} in Fig. 2.9).

Non-controllable loads and occupant

A set of user-driven loads are simulated through the use of *vEntity* class *vLoad*. They contribute to the overall building power consumption and to the release of heat that affects *vTempAirZ1* and *vTempAirZ2*. Two types of *vLoad* subclasses run in the *vEngine*: purely non-controllable loads as *vNonContrLoad* and deferrable loads as *vShiftLoad*. In addition to the virtual components, a virtual human *vOccupant* actuates them by switching them on/off or dimming their controllable power. The human behavior on any load is contained in the metadata of the *vLoad*, as a load profile activity $\mathscr{A}$.

Typical residential loads have been taken from the Electricity Consumption and Occupancy (ECO) dataset [71] and their load profile structure is detailed in Appendix A.4. A total of 11 non-controllable loads run in the simulation: 4 lamps, 1 freezer, 1 fridge, 1 laptop, 1 air-exhauster, 1 stereo sound system, 1 TV, and 1 entertainment station. In addition, 3 deferrable loads can be scheduled by the central controller: 1 dishwasher, 1 washing machine, and 1 dryer.

Hot water tank and EWH

Hot water is provided to the home occupant by a different system than the AWHP: a EWH heats water in a dedicated hot water tank. The virtual entity *vEWH* models a simple electrical heater system with the efficiency of the water tank fixed to 0.98. This *vEWH* influences the virtual sensor *vTempWaterEWH* that leverages Eq. (1.21). Model parameters are provided in

Appendix A.1.1.

The virtual entity *vTempWaterEWH* also requires outside ambient temperature and hot water usage information by subscribing to *vTempOutside* and *vOpenerEWH*, respectively. Modeled as a virtual "flow regulator", the *vOpenerEWH* receives ON/OFF and dimming parameters from the *vOccupant* to output the corresponding water flow usage necessary for the model of the hot water temperature. We considered the water usage to take instantaneous values ranging from 0 to 0.1 l/s, and it is assumed that one water-use event happens hourly, according to a uniform distribution. Such an action lasts on average 240 seconds, with a standard deviation of 60 seconds. Using hot water induces incoming cold water at $14°C$ to flow in.

PV arrays and EV

Solar energy is harnessed to locally produce electricity at the building premise. The power output of the corresponding PV installation is modeled by *vPVpanels*, according to Eq. (1.40), with a nominal value of 5.775 kW. This virtual entity subscribes to data coming from *vTempOutside*, *vIrrDiffOutside*, and *vIrrDirectOutside* to compute the instantaneous power production.

The Minergie building hosts a *Tesla Model S* electric vehicle, with an autonomy of 490 km. The chemical battery has a capacity equal to 75 kWh (soc_{max} = 72.5 kWh) and a maximum charging power of 7 kW. The charging efficiency is taken as 90%. The virtual entity *vPHEV* models this Tesla battery through Eq. (1.32) and Eq. (1.33), using KiBaM parameters provided in Appendix A.1.1. As a residential vehicle is out of the dwelling premise on most weekdays, the battery is out of the building model during that period. It is assumed that the vehicle is available every day during t^{ev}_{start} (h) and t^{ev}_{stop} (h) such that:

$$t^{ev}_{start} \sim \mathcal{N}(6,1) \text{ and } t^{ev}_{stop} \sim \mathcal{N}(18,1)$$

The vehicle arrives at the dwelling premise with an initial SoC at t^{ev}_{start} and should ideally be fully charged at t^{ev}_{stop}. The value $soc(t^{ev}_{start})$ depends on the distance the vehicle traveled during the day [34]. In this simulation, $soc(t^{ev}_{start})$ is uniformly distributed between 0.3 and 0.5. It is then assumed that the dweller has the choice to offer the battery as an ESS or the EV might only be considered as a load to charge.

2.3.2 Energy simulation

A total of 30 *vEntities*, mostly described here above, were spawn in the *vEngine* to simulate the Minergie building. Their corresponding parameters and relationships were encoded in the BDMS and retrieved by the *vMid* at its instantiation. Appendix A.2.2 provides a concrete example of BDMS encoding of a virtual entity. The *vCoordinator* orchestrates the simulation, and a fixed time step has been chosen over an event-based simulation, for the sake of simplicity. An Intel Core i7-4710HQ CPU (2.50GHz × 8) was used to practically perform the simulation.

Algorithm 2 Minergie simulation: flexible entity control logic

```
procedure CONTROL_BUILDING(time $t$)
    if $T_{ewh} \geq T_{ewh}^{mean}$ then
        $u_{ewh}[t] \leftarrow 0$
    else if $T_{ewh} < T_{ewh}^{mean}$ then
        $u_{ewh}[t] \leftarrow u_{ewh}^{max}$
    else
        $u_{ewh}[t] \leftarrow u_{ewh}[t-1]$
    end if
    if $T_Z \leftarrow T_Z^{mean}$ then
        $u_{hp}[t] \leftarrow 0$
    else if $T_Z \leftarrow T_Z^{min}$ then
        $u_{hp}[t] \leftarrow u_{hp}^{max}$
    else
        $u_{hp}[t] \leftarrow u_{hp}[t-1]$
    end if
    if EV connected and EV not full then
        $u_{ev}[t] \leftarrow u_{ev}^{max}$
    else
        $u_{ev}[t] \leftarrow 0$
    end if
    for all $k \in N_d$ do
        if $t \geq t_d^s[k]$ and $k$ not yet scheduled then
            $u_d[k] \leftarrow ON$
        else
            $u_d[k] \leftarrow OFF$
        end if
    end for
    APPLY([$u_{ewh}$, $u_{hp}$, $u_{ev}$, $u_d$])
end procedure
```

Some entities in the simulation wait for an external command to update their internal state. The chosen logic is a rule-based control, described in Algorithm 2. Thermal loads *vEWH* and *vHP* are switched on and off according to a thermostat logic. As for the HP, it is switched on (maximum power) any time the zone temperatures T_Z (either zone 1 or 2) reaches its lower bound $T_Z^{min} = 19°C$. Alternatively, it is switched off whenever the regulated temperatures reach the mean comfort $T_Z^{mean} = 20.5°C$. The mean value is chosen to trigger the switching off in order to account for the large thermal inertia of the floor heating system that will keep heating the zones even when the HP is off. The control rule of the EWH is slightly different to track the mean comfort: it is switched on whenever the water temperature T_{ewh} goes below the mean acceptable temperature $T_{ewh}^{mean} = 60°C$, and is off otherwise. Good insulation of the tank prevents large oscillation of the control signal. Concerning the Tesla EV, the rule consists

Table 2.1 – Minergie building energy simulation: simulator statistics

Time step (minutes)	Tot. Events	vMid Overhead	Simulation time (s)
1	51700	5.56	71.18
5	11296	1.216	11.37
15	4109	0.463	4.39
60	1187	0.133	1.54

in enforcing maximum charging power whenever it is plugged-in to the building and not full yet. As for the N_d deferrable loads, they are simply triggered as soon as possible (similar to the non-controllable loads).

A range of time steps has been used to show the performance of the tool. Table 2.1 displays a few statistics about the distributed simulator under varying time steps. Given a simulation time step of 60 seconds, this simple yet realistic building simulation needed a bit more than 1 minute to perform a daily simulation, through the exchange of a little more that 50,000 messages. As the time step is increased at the expense of accuracy - the simulator behaves linearly, both in term of messages and duration. The *vMid* overhead represents the total number of times the *vMid* used the CPU, staying around 8-10 % of the total simulation time. Fast and numerous message exchanges could be ensured by the optimized ZeroMQ library. Furthermore, these values could be reduced by implementing an event-based simulation, as many *vEntity* were woken up for performing insignificant short increases in entity state, at low time steps.

One advantage of the distributed simulator lies in the analysis of individual entities' behavior. Fig. 2.10 highlights each individual load's power consumption. This allows for both individual and comparative analysis. In this Minergie example, most of the non-controllable loads have their main activities in the late evening, apart from the fridge and freezer that are periodically switched on and off. One can already observe the difference in order of magnitude when comparing non-controllable to controllable loads.

Applications connected to the BDMS post-process the generated data to generated aggregated analysis as shown in Fig. 2.11 and Fig. 2.12. In this simple simulation, the main load is driven by the EV that charges early in the morning. This behavior is obviously sub-optimal, as the controller did not take into account the price of electricity (in red, middle figure) nor the local PV production. To a lower scale, the same statement can be made for two uncontrollable loads (Washing machine and Dryer). Luckily, the HP switched off around the starting time of the EV charging.

Temperature comfort of air zones and hot water could be ensured by Algorithm 2, as proven in Fig. 2.12. One can clearly observe the supply and return water temperature of the hydronic system, with the former being higher than the latter. As for the zone air temperature, the cold outside temperature pushes it down at the same time as the hydronic system pushes it up, also with the help of daily solar heat gain. It is worth noting that the hydronic system has been modeled without any loss. A more realistic case would demonstrate lower temperatures and a higher use of the HP. As hot water is consumed by the virtual user, tank water temperature abruptly decreases, which directly triggers the EWH to switch on and provide heat to ensure the user will have hot water for later use.

2.4 Conclusion and Outlook

A simulation tool tailored to building applications has been described in this chapter. Its unique architecture relies on a set of distributed micro-threads called *vEntities* that implement individual models. Their cooperative interaction leads to a global building behavior simulation, and the natural decentralized feature speeds up the simulation time when the hosting hardware allows it, compared to centralized solutions. The resulting pool of distributed virtual components is seamlessly connected to existing BDMS through the use of a virtual middleware *vMid* that acts exactly as its physical counterparts. In emulation mode, the *vMid* leads to a coexistence of virtual components along with physical infrastructure for integrated analysis. In simulation mode, the *vMid* acts as a simulation coordinator, through the use of the *vCoordinator* module. The mechanisms of the latter permits either a synchronized fixed time step simulation or a time-optimized event-based simulation. A test case on a realistic Swiss Minergie building highlighted the process of creating virtual entities and connecting them together. The resulting simulation was proven to be fairly fast, thanks to the lightweight Python libraries practically used for developing the tool.

By leveraging BDMS metadata, existing building infrastructure can be quickly digitalized without a large time overhead needed to migrate existing data to an offline simulator. In addition, the insertion of new purely virtual entities allows to simulate the effect of non-present technology on an existing building, before proceeding to the expensive retrofitting. The resulting digital twin can therefore continuously learn from the building data updates and can swiftly be upgraded with additional hardware for advanced analysis purpose. This represents a promising tool, directly integrated to the existing BDMS, for real-time evaluation of new component installation (e.g., sizing up solar panels or changing the heat pump), building envelope upgrade, and occupant behavior analysis without the burden of depending on an external expert tool.

Furthermore, the chosen approach is distributed and decentralized, meaning that the simulation designer will model building components individually, in a piece-by-piece process. Compared to centralized, monolithic methods, this leads to a better working load distribution, easier future individual modification, and adaptability. This architecture fully leverages the parallelism that a multi-core hardware host might offer, as many entities can be triggered at the same time from the *scheduling table*. Compatible with both continuous (physical) models and event-based (user-driven) model, the tool is ideal for introducing uncertainty and disturbances in the simulation. When plugging advanced applications to manage the flexible entities, real-life modeled uncertainties are of great value to test the robustness of the application's algorithm.

The tool, through the presented virtual entity classes, was primarily designed for residential buildings simulation. Nevertheless, it may host other building models, thanks to its intrinsic modularity that allows any user to insert her/his own models into the engine. By modeling more advanced a HVAC system, the *vEngine* could be applied to commercial buildings that hold a great potential for retrofitting, energy saving, and advanced energy management.

Beyond the simulation of a single building, the light design of the engine also enables the simulation of a set of smart-buildings, ready to be included in smart-grid frameworks. In this useful case, either a single BDMS can be used to model multiple units (i.e., buildings) or each engine could be plugged to a single instance of a BDMS, to emulate real-life implementation. Chapter 6 takes advantage of this feature to simulate a community of cooperative smart-buildings.

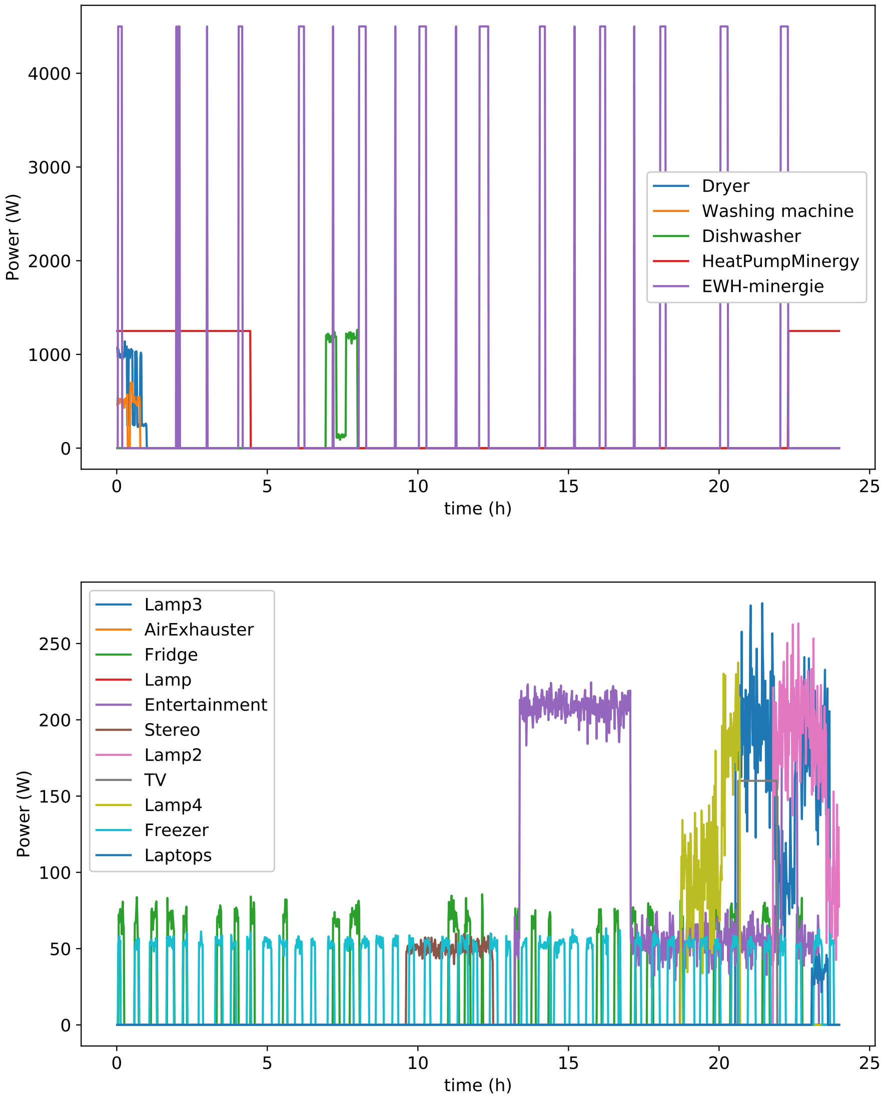


Figure 2.10 – Minergie building simulation (top) Non-controllable load consumption (bottom) Controllable load consumption

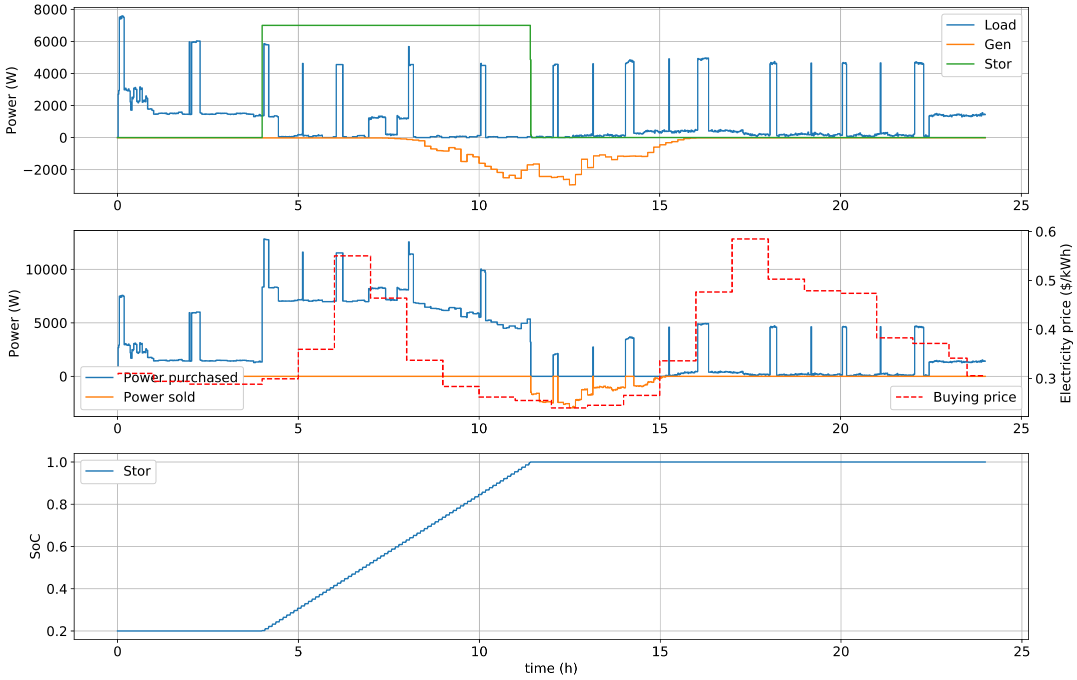


Figure 2.11 – Minergie building simulation: aggregated data

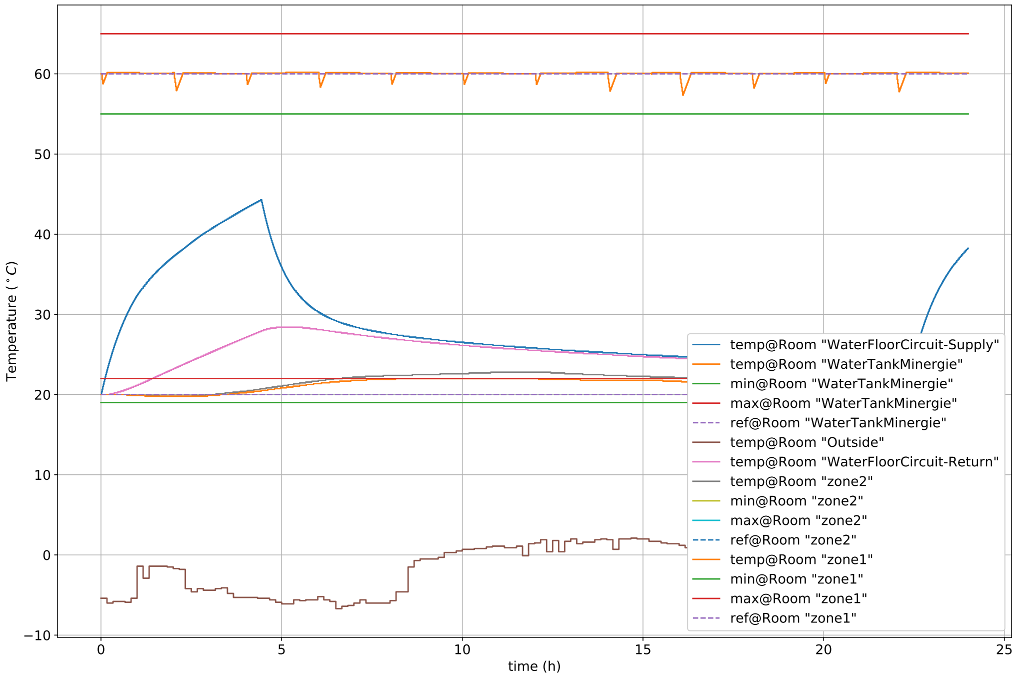


Figure 2.12 – Minergie building simulation: temperature comfort

3 Multi-State Load Modeling for Energy Disaggregation

Beyond the controllable entities present in the smart-buildings, non-controllable loads still represent a large share of total energy demand. Their consumption closely depends on the building occupants' behavior, the state of the building, and its environment. Understanding the way non-controllable loads drive whole building consumption will help enable energy efficiency. Therefore, it is important to equip future smart-buildings with the right technology for valuable energy feedback to the user at the individual load level.

The main **highlights** and **contributions** of this chapter are:

- We present an algorithm based on *k-means* to automatically extract appliance load profile parameters from historical time-series power data. The resulting profile structure contains both the power modes of the load and the user activities on the appliance.
- The identified multi-state load profiles have been used to conduct low-frequency energy disaggregation. Compared to traditional ON/OFF load modeling, this requires an enhanced formulation of the Hidden Markov Model (HMM), which was implemented in the MATLAB *NILM-Eval* toolbox.
- The resulting multi-state energy disaggregation was run against traditional binary load modeling using the Electricity Consumption and Occupancy (ECO) dataset. Results show that enhanced multi-state modeling significantly improves the disaggregation performances, at the expense of a larger computational time.

Related **publications**:

[72] O. Van Cutsem, G. Lilis, and M. Kayal, "Automatic multi-state load profile identification with application to energy disaggregation," *IEEE International Conference on Emerging Technologies and Factory Automation*, 2018.

3.1 Introduction and Motivation

The recent change of paradigms in the energy sectors led to a growing interest in Demand Side Management (DSM) [10]. The smart-building, a building infrastructure enhanced with sensors, actuators, and management systems, fosters the active inclusion of residential and commercial buildings into the smart-grid. Supported by ICT equipments, the smart-building can leverage the flexibility offered by controllable loads, energy storage systems, and local energy production, to further provide services to the smart-grid. Nevertheless, part of the energy demand still solely remains under the control of the dwellers, referred to as *non-controllable* load consumption.

In the context of smart-buildings, energy feedback systems [73] represent a powerful means of bringing the building occupants into the loop of DSM, concerning the user-driven share of energy consumption. These feedback systems generally improve human awareness of the electrical energy repartition within the building, hence understanding how various appliances influence total energy consumption.

Due to the intrusiveness and excessive cost of installing a dedicated sensor for every home appliance, methods fostering Non-Intrusive Appliance Load Monitoring (NILM) have been developed for energy monitoring purposes. NILM algorithms disaggregate the superposition of individual signals, sometimes taking into account appliance parameters or relying on features database. Nowadays, smart meters have largely been deployed at building premises and allow low-cost solutions for monitoring the overall building consumption, based on low-frequency power data The design of the smart-building requires large controllable loads to be equipped with sensing and actuation hardware for active participation in the DSM. In that context, NILM methods can be applied at outlet-level for the remaining non-controllable loads, hence grouping user-driven appliances sharing the same order of magnitude with regards to power consumption.

This chapter presents an automatic method for detecting the multiple power states of any appliance. It is trained on low-frequency historical data of the appliance's active power collected at plug-level. The output multi-state modeling of appliances is then used for power disaggregation purposes. State-based HMM has been chosen in support of disaggregation implementation, allowing the comparison between the standard binary ON/OFF modeling and the proposed multi-state modeling of the appliances. Finally, a clustering-based algorithm has also been developed to extract a representative set of user activities on the building appliances.

An overview of the existing state-of-the-art load models, NILM algorithms, and a theoretical background about HMM modeling is provided in Section 3.2. Then, Section 3.3 describes the multi-state identification algorithm, from the clustering task to the extraction of power modes. The experimental results of multi-state HMM-based power disaggregation from a public dataset are detailed in Section 3.4. Finally, Section 3.5 concludes the chapter.

3.2 State-of-the-art Load Modeling and Energy Disaggregation

3.2.1 Load modeling

Load modeling and power predictions for buildings have received intense attention, essentially at the whole building level and community aggregation [74, 75, 76]. However, models of individual loads have been investigated less due to the lack of granular data that smart meters do not capture. Hart et al. [77] initially presented three different classes of load model: ON/OFF (binary load), Finite State Machine (FSM), and continuous variable. In the ON/OFF model, the load power consumption can only take a single mean value and a standard deviation describes the variation around the mean value. As most of the loads can experience more than one steady-state power consumption mode - generally depending on user-related workload - the FSM allows for an arbitrary set of discrete states and transitions [78]. Finally, continuous variables model dimmable loads, such as lighting and some HVAC systems. Authors in [79] presented a dynamic programming technique for extracting FSM from the aggregated signal, but did not apply it to any dataset. In [80], a mathematical model represents the load profiles of various residential buildings, using a bottom-up approach. They used generic models of household appliances and introduced a saturation level that statistically represents their presence across a large set of building populations. Considering a single nominal power consumption (W) per load, the authors proposed the use of a mean daily starting frequency, along with a fixed time per cycle, to model user actions on the appliance.

Some works have studied the electrical properties of loads to model them from an electrical standpoint. In [81], a voltage-dependent model of appliances is presented, leading to a bottom-up method to retrieve an electric parameters profile (current, active/reactive power, power factor, etc.). This model was implemented in MATLAB to get residential and commercial building energy profiles, and shows results similar to EnergyPlus simulations. A survey is presented in [82], where individual load characteristics (active/reactive power, current, and harmonic content) are used to classify residential loads. They also emphasize the difficulty of capturing individual load electrical characteristics, due to the influence of other loads. Authors of [83] created a high-resolution smart home power demand model that takes into account the activity patterns of individuals, based on the non-homogeneous Markov chain. It generates highly realistic patterns that capture annual and daily features of building consumption, for DSM potential analysis.

It's also worth noticing that advanced building energy simulators like EnergyPlus practically use fixed (hourly) schedules of appliance consumption [38]. The schedules generally define the fraction of the load nominal power use in each period of time and therefore does not use granular data of the appliances' power consumption.

3.2.2 Energy disaggregation

NILM was initially investigated by G. Hart [77] and identified ON/OFF switching events in the aggregated power signal, analyzing them in the P-Q plan, for appliance clustering. With the help of a signature database, he could then retrieve the appliances that caused the switching events. Subsequently, substantial research in load identification and power disaggregation has been carried out, mainly summarized in [84, 85]. With the distinction made between the frequency of data sampling, the reviews classify NILM methods into supervised or unsupervised, stateless or state-based, and dissociate them based on the analyzed features (P, Q, power factor, etc). Most of the unsupervised NILM algorithms require the manual intervention of the user to associate the disaggregated signals/events to the actual appliance.

The stateless methods generally consist of extracting the most useful features of switching events and steady-states, followed by an efficient clustering and classification. In [86], the authors presented two algorithms based on Decision-Tree that stores events and Dynamic Time Warping, working on active power sampled either every 6 seconds or 1 minute. Their main advantage is the required small training period. Researchers in [87] also based their approach on a Decision-Tree that stores pairs of events. Their goal was to investigate the feasibility of load disaggregation for consumer service development, showing the user its energy consumption with an acceptable error of 10%. In [88], authors developed a Non-Intrusive Load Identification, at the outlet level, using a time-series classifier. The resulting generated database holds features ranging from average power, power variance, power extrema, and duty cycle to waveforms of the power consumption. Baranski et al. [89] developed a fuzzy clustering method for detecting main recurrent events. A genetic algorithm was then used for forming a finite state machine, optimized using dynamic programming. Taking into account the infrequent nature of the switching events, the authors in [90] leveraged the sparsity of the on/off switching events matrix, leading to the Sparse Switching Event Recovering algorithm. Furthermore, they developed an optimization algorithm that speeds up the disaggregation process. Wytock et al. [91] presented a contextually supervised source separation, that lies between supervised and unsupervised NILM. A convex optimization problem takes as inputs the features of each source signal for estimating their participation in the global aggregated signal.

State-based NILM was greatly enhanced by the use of HMM for modeling the individual appliance states and their transitions. Kim et al. [92] detailed an unsupervised use of Factorial Hidden Markov Model (FHMM) for low frequency power disaggregation purposes and developed an extension called Conditional Factorial Semi Hidden Markov Model (CFSHMM) that takes into account the ON/OFF duration distribution, the appliances dependency, and the time of the day. The Expectation Maximization (EM) algorithm was used to determine the model parameters while Gibbs sampling replaced the Viterbi algorithm for inferring the hidden states. Authors of [93] presented methodologies for building HMM, aiming for load recognition including multi-state model of appliances. Parson et al. [94] leveraged the prior knowledge of a generic appliance model so that manual labeling could be avoided. The specific model parameters are learned through the EM algorithm in periods during which only one appliance is active.

The inference process in FHMM-based approaches suffers from many issues, namely enumerating an exponential number of states and local optima convergence. In [95], Kolter et al. explained their approximate inference algorithm for energy disaggregation using additive factorial HMM, leading to the AFAMAP convex optimization problem. The latter strives to alleviate the aforementioned hurdles hampering the inference phase. Furthermore, authors in [96] presented a segmented application of the Viterbi algorithm for state decoding of the HMM. In [97] a Hierarchical Dirichlet Process Hidden Semi Markov Model is presented, a method that learns the device models during inference process and takes into account multi-state model of appliances, on the contrary to most of the binary ON/OFF approaches. Unlike common NILM applied to the whole house power consumption, authors in [98] focused on circuit-level monitoring and have proposed both a heuristic-based and a Bayesian approach, where information about step changes in power are added to steady-state power information. Authors of [99] propose a fully unsupervised NILM framework based on non-parametric FHMM and have developed an efficient inference algorithm to detect the number of appliances from data and disaggregate the power signal simultaneously.

Many NILM applications have been developed and practically deployed. AppliSense [100] created by Weiss et al. is based on Hart's approach that uses distortion power in addition to P and Q features. A mobile phone application has been designed to train and acquire new load signatures for the user to specify which appliance has been switched on/off. ViridiScope [101] is a power monitoring system based on ambient data retrieved from inexpensive sensors, such as magnetic or acoustic sensors. As the total power consumption is also taken into account, an explicit optimization problem may be formulated, whose solution provides the individual appliances consumption profile. RECAP [102], a complete real-time solution for recognition and profiling of appliances, uses Artificial Neural Network (ANN) for features training such as P-Q, RMS current/voltage, and peak current/voltage. Through a user-interface, this solution can easily be installed in existing buildings. In [103], NILM found an interest for load-shed verification, such that utilities can check whether a Demand Response subscriber has performed the contracted behavior.

Most of the presented disaggregation approaches, with the exception of [93, 97, 104], focus solely on binary ON/OFF models, averaging power consumption over the active periods. The present chapter investigates the potential of the multi-state modeling of appliances' active power consumption for power disaggregation, compared to the common binary ON/OFF modeling. The supervised approach uses FHMM in which each appliance may have more than one ON state. Those numerous active states are automatically extracted from past time-series data, unlike [93, 104] that manually encoded them. Similar to the context of [98], the disaggregation tasks take place at outlet level, grouping appliances that share the same order of magnitude of power consumption.

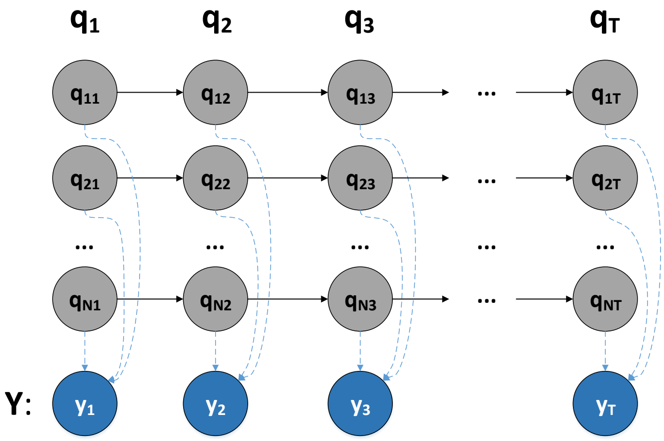


Figure 3.1 – FHMM: A graphical interpretation of appliances hidden states and the whole aggregated consumption

3.2.3 Factorial hidden markov model applied to NILM

In the domain of NILM, HMM has recently become the reference method because of its simplicity for modeling time-series data in a probabilistic way. In a HMM, the observation of sequential data $\boldsymbol{Y} = \{y_1, y_2, ..., y_T\}$ is caused by sequential hidden states $\boldsymbol{Q} = \{q_1, q_2, ..., q_T\}$ that have to be determined. Each of the hidden states takes a value among the finite set $S = \{S_1, ..., S_K\}$, where K is the total number of possible states. The HMM then defines three quantities for modeling the sequential process:

- The initial state probability distribution $\boldsymbol{\pi} = \{\pi_1, ..., \pi_K\}$:

$$\pi_i = p(q_1 = S_i) \ \text{ such that } \sum_{i=1}^{K} \pi_i = 1$$

- The transition matrix $\boldsymbol{A}$ whose element $A_{i,j}$ represents the probability to switch from state i at time t to state j at time $t+1$:

$$A_{i,j} = p(q_{t+1} = j | q_t = i) \ \text{ such that } \sum_{j=1}^{K} A_{i,j} = 1$$

- The emission matrix $\boldsymbol{B}$ whose element $B_{n,j}$ represents the probability of seeing a particular observation n during state j:

$$B_{n,j} = p(\Phi_t = n | S_t = j)$$

As an extension of HMM, FHMM [77] takes multiple appliances into account by modeling each of them with a sequence of hidden states, as illustrated in Figure 3.1. The combination of those hidden states at each time instant leads to the sensed aggregated consumption $\boldsymbol{Y} = \{y_1, y_2, ..., y_T\}$, i.e., $\boldsymbol{Q} = \{\boldsymbol{q_1}, \boldsymbol{q_2}, ..., \boldsymbol{q_T}\}$, in which $\boldsymbol{q_i} = \{q_{1i}, q_{2i}, ..., q_{Ni}\}$, N standing for the total amount of aggregated appliances. In the context of NILM, the elements of the emission matrix $\boldsymbol{B}$ commonly follow a Gaussian distribution, i.e., $p(\Phi_t = n|S_t = j) \sim \mathcal{N}(\mu_j, \sigma_j)$ for each state j. As reviewed in the former section, the set S of a given appliance is generally made of the ON and OFF states. In this chapter, we consider the general case of multiple non-null states.

Disaggregating a power signal consists of leveraging the combined FHMM to retrieve individual appliance participation in the aggregated signal. The first step aims to learn the parameters π_i, $A_{i,j}$, and $B_{n,j}$ for each appliance from historical data. This is the focus of the next section. Then, the disaggregation of a new incoming signal leads to the determination of the hidden states of the appliances in a new aggregated signal, which can be done using Maximum Likelihood Estimation (MLE). Applying MLE in energy disaggregation results in finding hidden states $\hat{\boldsymbol{q}}$ that are the most likely to have triggered the measured aggregated signal:

$$\hat{\boldsymbol{q}} = \underset{\boldsymbol{q}}{\operatorname{argmin}}\, P(\boldsymbol{Y}, \boldsymbol{q}|\lambda) \tag{3.1}$$

where λ is the set of parameters linked to the FHMM. The Viterbi algorithm, popular for efficiently estimating the hidden states of a HMM, cannot practically be used for estimating the FHMM [77]. Instead, methods like Simulated Annealing [105] can be used to infer the hidden states of the FHMM. While this method is unlikely to find a global optimum, it generally finds a very good solution even in the presence of noise.

3.3 Multi-State Load Profile Identification Algorithm

The non-controllable load power profile model was briefly introduced in Chapter 1. It is made up of a set of power modes $\mathscr{M}$, mode sequences $\mathscr{S}$, and user activities $\mathscr{A}$. Figure 3.2 illustrates the sets $\mathscr{M}$ and $\mathscr{A}$ on a 24h load profile example. On the one hand, each mode $\{m_i\}$ describes the possible power consumption of the load in the corresponding state. On the other hand, an activity a_i contains information about how the appliance is used by the user throughout the day and is therefore specific to each user. Within an activity, the structure $\mathscr{S}$ will dictate the transitions from a power mode to another.

This load profile structure solely models the steady states of power consumption, and thus does not catch the short transition modes. In the example of Figure 3.2, the load may be in five distinct states. The off and standby states are represented by M_0 and the state M_4 captures the frequent high peaks of the appliance. Modes M_1, M_2, and M_3 are the more common power states of the appliance at hand. These power consumption modes are observed in activities a_0 and a_1 throughout the day, outside of which the default mode is M_0. An activity is defined by a statistical distribution on its starting/ending time and duration, as well as the sequence of power modes $\mathscr{S}$ it experiences. For instance, the activity a_0 only contains M_1 and M_2, whereas the activity a_1 encompasses higher modes M_3 and M_4. In the rest of this section,

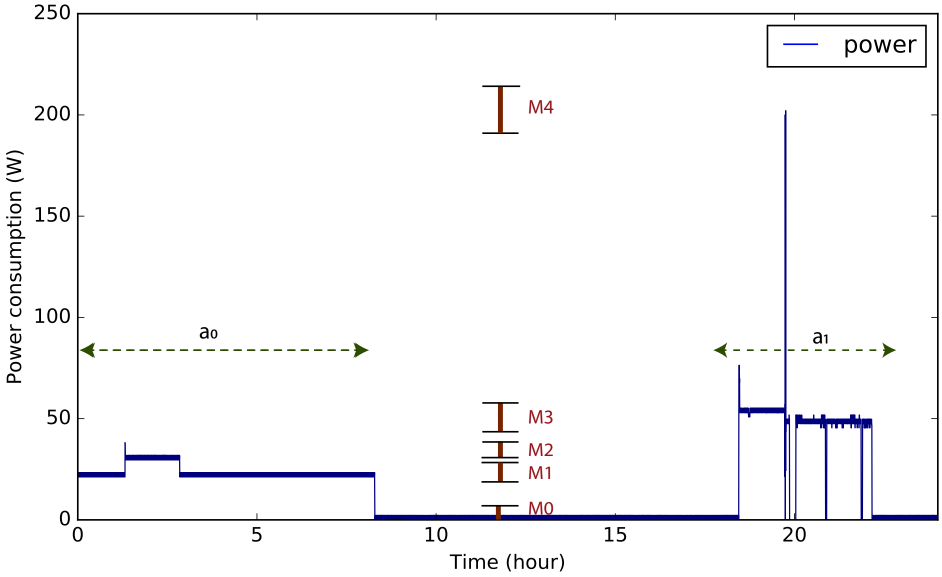


Figure 3.2 – Load profile modeling: example of modes and activities

we present the algorithm to automatically extract the set $\mathscr{M}$, and Section 3.4 will present the application of this algorithm for energy disaggregation, as well as a method for user activities $\mathscr{A}$ extraction.

Algorithm overview

The load profile identification algorithm's goal is to extract a set of distinct power states $\mathscr{M}$ of an appliance, given an incoming time-series vector $\boldsymbol{x}$, containing T samples. Each element m_i of the set $\mathscr{M}$ represents a power consumption state the corresponding appliance can be in. It is composed of the following parameters:

- The average power consumption μ_i in the state
- The variance σ_i^2 of the power consumption in the state
- The extrema $[P_i^{min}, P_i^{max}]$ of the power consumption in the state

The purpose of the algorithm is therefore to estimate the optimal number of states along with the aforementioned parameters characterizing each of them. Figure 3.3 shows the flowchart of the proposed states identification algorithm. A pre-processing step aims at removing undesired outliers that are not representative of the appliance steady states. Then, the clustering phase iteratively gathers power consumption into distinct groups, whose amount increases along with the iterations. Based on the formed groups, the cluster selection step chooses

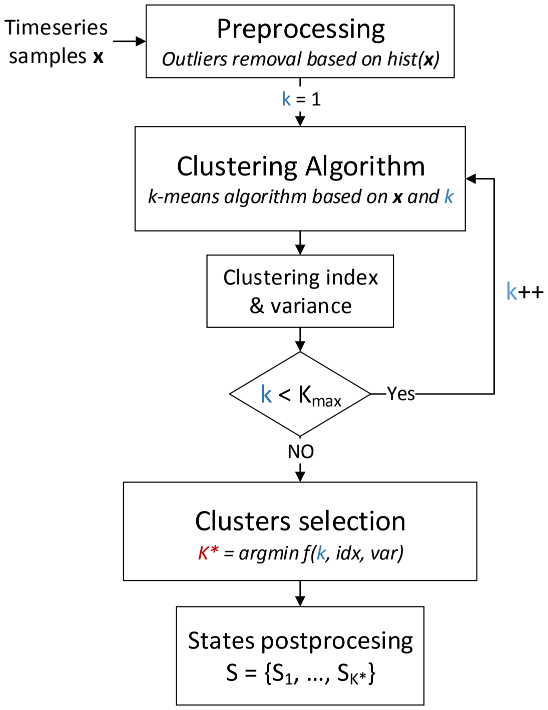


Figure 3.3 – Flowchart of the appliance state identification algorithm

the most optimal combination of groups, a trade-off between clustering performance and a desired low number of states. Finally, similar states that are close enough are merged during the post-processing phase. Taking into account that the resulting multi-state model is further used for power disaggregation purposes, the total amount of states must be reduced as much as possible. An excessive number of states would slow down the disaggregation process and might also degrade the accuracy of the result since different appliances could share the same states.

Time-series data preprocessing

The clustering algorithm in use, described in the following subsection, is sensitive to outliers in the time-series data $\boldsymbol{x}$, due to the norm it relies on for updating centroids. To remove those outliers while avoiding losing useful information, the histogram $\boldsymbol{h}$ of the vector $\boldsymbol{x}$ is computed, associating an occurrence frequency for each power consumption range. The whole range of power consumption of $\boldsymbol{h}$ is then sliced into N_{bins} bins of equal width, in which the corresponding occurrence frequencies are summed up. An outlier bin is then defined as a bin containing less than a specified number of occurrences, depending on the total amount of values in the histogram. For each detected outlier bin, the corresponding values in the initial

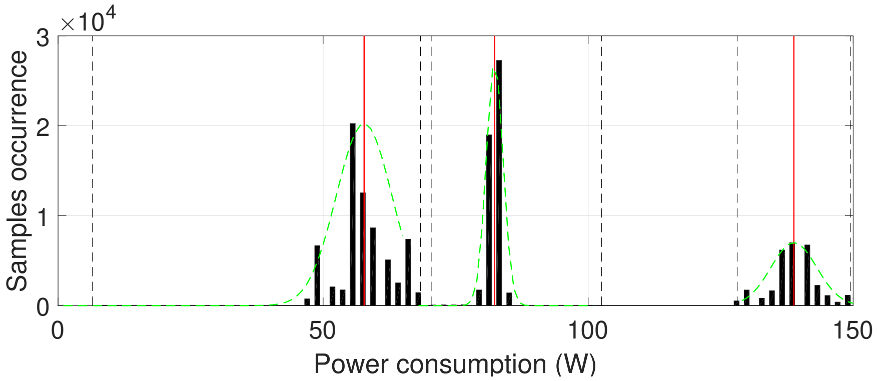


Figure 3.4 – Histogram of an air exhauster's power consumption. In red, the centroid of the identified clusters and in green, the corresponding estimated normal distribution

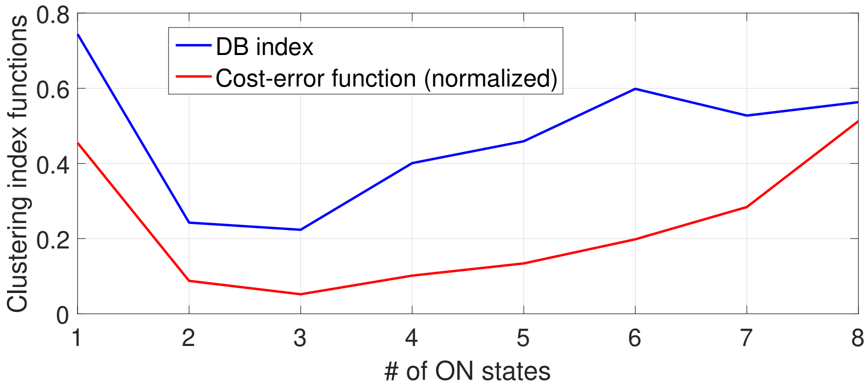


Figure 3.5 – Air exhauster power consumption: DB clustering index and cost-error function used for states identification

time-series data $\boldsymbol{x}$ are removed:

$$x_i = 0 \; \forall x_i \in j \text{ s.t. } h_j \leq \alpha_{out} \cdot T \tag{3.2}$$

where α_{out} is a value in $[0;1]$ and T is the total number of samples. By doing so, the infrequent steady-state or transient power consumption can be removed and won't jeopardise the identification of the main states.

As a last preprocessing step, the values of $\boldsymbol{x}$ that are inferior to a given precision P_{thres} are gathered into the OFF state. This includes the measurement errors due to the sensors as well as a possible standby neglectable power consumption. Hence, the clustering phase only works on $\boldsymbol{x} \geq P_{thres}$ and therefore analyzes the power consumption states when the appliance is switched ON.

Clustering algorithm

The state identification is carried out through pattern recognition in the one-dimensional active power consumption histogram $\boldsymbol{h}$. The k-means algorithm is preferred over the non-parametric Mean-Shift clustering algorithm [106]. Indeed, with the main objective of minimizing the optimal number of clusters while keeping a representative set $\mathscr{M}$, the parameter k - i.e., the number of clusters - should be progressively tuned for estimating the quality of new clusters. Moreover, the simplicity and average good results of k-means motivate its choice for the current application, whereas the Gaussian Mixture Models algorithm and Fuzzy C-means could be useful for solving the overlapping clusters issues [107]. The latter case rarely appears for appliance power state identification since the states to be extracted must be completely distinct for FHMM-based disaggregation application, and can generally be labeled. Furthermore, k-means is suitable for the large time-series dataset used in the current context.

In [108] and [109], authors used the k-means algorithm in an iterative way for selecting the clustering solution that best groups $\{P, Q, STC\}$ features for NILM purposes [1]. Yet quantities Q and STC are not commonly available in most of the BDMS database, unlike low-frequency active power P. Similarly, the preprocessed time-series data $\boldsymbol{x}$ will be grouped in k clusters by the k-means algorithm, with k iteratively varying from 1 to K_{max}. For each cluster number k, the set of cluster centroids $\mathscr{C}_k$ are given by solving the following optimization problem:

$$\mathscr{C}_k = \underset{c_j}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x_i \in \mathscr{C}_{k,j}} \left\| x_i - c_j \right\|^2 \tag{3.3}$$

where c_j is the centroid of $\mathscr{C}_{k,j} \in \mathscr{C}_k$. The k-means formulation actually strives to directly minimize the intra-class variance S_j in every cluster:

$$S_j = \frac{1}{N_j} \sum_{i=1}^{N_s} ||x_i - c_j||^2 \tag{3.4}$$

where N_j is the number of samples in cluster j. The k-means algorithm will therefore start by selecting "random" initial centroids $\mathscr{C}_k$ amongst the power consumption data. These starting centroids, called "seeds", will condition the performance and convergence of the algorithm. Methods such as *k-means++* [110] aim to find the best starting points to converge toward near-optimal clusters. Then, the algorithm iteratively applies an *assignment* step and a centroid *update* step, until convergence. The *assignment* step browses all the points x_i in the dataset to assign them to a unique cluster $c_{k,j}$, such that the following property is respected:

$$\left\| x_i - c_{k,j} \right\|^2 \leq \left\| x_i - c_{k,l} \right\|^2 \ \forall l \neq j \tag{3.5}$$

After all the points have been assigned to their closest cluster, the set of centroids $\mathscr{C}_k$ are *updated* as follows:

$$c_{k,j} = \frac{1}{N_j} \sum_{x_i \in \mathscr{C}_{k,j}} x_i \tag{3.6}$$

---

[1] P stands for the active power consumption, Q for the reactive power consumption, and STC for the switching transient current.

where $\mathscr{C}_{k,j}$ is the set of points assigned to the centroid $c_{k,j}$. These steps are repeated until the set of centroids $\mathscr{C}_k$ does not change between two consecutive iterations, hence reaching convergence.

States selection

Among the created sets $\{\mathscr{C}_k\}$, one has to find the combination of clusters that best suits the multi-state modeling task. Many indices exist for evaluating the quality of a set of clusters [111], such as Bayesian Information Criterion (BIC), Calinski-Harabasz (CH) index, Davies-Bouldin (DB) index or Silhouette (SH) index. The DB index fits for the current application because it penalizes centroids that are too close while fostering the low variances clustering solutions.

For each cluster in $\{\mathscr{C}_k\}$, the quantity D_i can be defined as:

$$D_i = \underset{j \neq i}{max} \frac{S_i + S_j}{|c_i - c_j|} \text{ where } j \in [1, ..., k] \tag{3.7}$$

where S_i, S_j are defined by Eq. (3.4) and c_i, c_j by Eq. (3.6).

Then the DB index can be derived as:

$$\mathrm{DB}_{idx}(k) = \frac{1}{k} \sum_{i=1}^{k} D_i \tag{3.8}$$

A lower value of $\mathrm{DB}_{idx}(k)$ indicates a better DB fitting for the clustering $\mathscr{C}_k$. However, selecting k solely based on the DB index may lead to an excessive number of states for some cases. A cost function f_c is hence used to foster the selection of small values of k:

$$f_c(k) = \begin{cases} a_l \cdot k + b_l & \text{if} \quad 1 \leq k < K_{av} \\ a_e \cdot e^k + b_e & \text{if} \quad K_{av} \leq k < K_{max} \end{cases} \tag{3.9}$$

where a_l, b_l, a_e and b_e are parameters to tune and K_{av} represents the threshold above which the number of states becomes costly[2].

The final *cost-error* function $\epsilon(k)$ is a trade-off between DB_{idx} and f_c, and also takes into account that the total standard deviation has to be minimized:

$$\epsilon(k) = \mathrm{DB}_{idx}(k) \cdot f_c(k) \cdot \sqrt{\sum_{j=1}^{k} S_j} \tag{3.10}$$

Finally, the optimal K^* is taken to be the solution of:

$$k^* = \underset{k}{\operatorname{argmin}}\, \epsilon(k) \tag{3.11}$$

The corresponding set of clusters $\mathscr{C}_{k^*}$ further defines the mean μ_i, standard deviation σ_i, and extrema $\{P_i^{min}, P_i^{max}\}$ of the power consumption of each power mode m_i, $i = 1, ..., k^*$.

---

[2] It is fairly reasonable to fix K_{av} to 4 and K_{max} to 8 for power disaggregation purposes.

Figure 3.4 shows the power histogram of an air exhauster device, along with an estimated Gaussian distribution for each of the three clusters identified by the algorithm. In Figure 3.5, the corresponding cost-error function $\epsilon(k)$ displays a minimum for $k = 3$ that defines the number of active states for this appliance. One can observe that the DB index decreased around the larger values of k, while the function $\epsilon(k)$ is convex.

States merging

The above algorithm may result in the identification of close states that could decrease the quality of the load disaggregation process. The post-processing step ensures that centroids spaced less than α_{dist} percent from the total power range are merged together. A value of $\alpha_{dist} = 5\%$ is a reasonable compromise for preventing close states while ensuring enough granularity.

3.4 Case Study on the ECO Dataset

The presented multi-state modeling algorithm aims at enhancing the automatic detection of an appliance's load profile. Among others, this appliance modeling can be beneficial for FHMM-based power disaggregation techniques. In this section, the experimental results of the multi-state modeling are carried out on a public dataset for outlet-level power load disaggregation tests and are compared to binary ON/OFF modeling. Time-series data collected at plug-level for various appliances is used for the load profile identification algorithm, that outputs the multi-state model of each appliance. The true aggregated signal is then formed by summing up the individual time-series signals.

3.4.1 The ECO dataset

Many datasets are publicly available in support of the research in the NILM domain, such as the REDD [112], BLUED [113], GREEND [114] or the ECO dataset [71]. In the ECO dataset, data has been sensed at the building smart-meter level and also at plug-level for six different houses, over a period of eight months at a granularity of 1Hz. This dataset has been chosen for its many user-driven appliances, particularly in household two, which will be the only household of interest.

The ECO dataset has primarily been designed for occupancy estimation based on whole-house consumption [115]. Nevertheless, the granularity and large data it holds made it a good candidate for NILM evaluation. Table 3.1 shows the loads that are used for the experiments, along with the results of the automatic states identification algorithm. Figure 3.6 displays a histogram for some of the loads and a graphical representation of the state's identification results. A sampling period of 180 days has been used, leading to the identification of more that one active state for most of the tested loads.

Table 3.1 – Automatic states identification of appliances in Household 2 of ECO dataset

Appliance name	K^*	**State means (W)**
Tablet	4	{0 ; 6.4 ; 8.6 ; 10.7}
Dishwasher	3	{0 ; 123.7 ; 2192.3}
Fridge	3	{0, 15.3 ; 70.7}
Entertainment	4	{0 ; 24.1 ; 53.8 ; 208.4}
Freezer	2	{0 ; 53.8}
Water kettle	2	{0 ; 1838.3}
Lamp	3	{0 ; 85.1 ; 185.2}
Laptop	3	{0 ; 31.0 ; 69.6}
TV	2	{0 ; 160}
Stereo	4	{0 ; 24.0 ; 50.2 ; 149.2}

3.4.2 Load power disaggregation

The authors of the ECO dataset have also developed *NILM-Eval*, a combined MATLAB-Python framework for evaluating the most popular disaggregation algorithms on any dataset. The implemented algorithms cover the various approaches of the design space of NILM algorithms: Parson [94], Baranski [89], Weiss [100], Kolter [95], and FHMM-based algorithms.

Basis on *NILM-Eval*, the multi-state identification algorithm could easily be integrated for determining the appropriate number of states for each appliance, along with the means and variances for each of the identified states. The EM algorithm, run by default for FHMM parameter identification based on plug-level data, was used for determining the transition matrix A and the initial state probability distribution π. This corresponds to the set of sequences defined in the proposed Load Profile (LP) structure, presented in Chapter 1. The multi-state identification process and the subsequent EM algorithm constitute the training phase for load disaggregation. Once all the individual FHMM models have been trained, the combined FHMM model is instantiated for power disaggregation purpose. From an input time-series data vector resulting from the aggregation of N_a real consumption signals, the *NILM-Eval* toolbox allows the disaggregation algorithm to be run.

To evaluate the performance of the multi-state FHMM-based power disaggregation, the corresponding binary FHMM-based power disaggregation has been run with the same configurations, also referred to as ON/OFF. For each of the disaggregated signals $\widetilde{\boldsymbol{x}_i}$, the difference with the real one $\boldsymbol{x}_i$ is quantified by three different quantities:

- The F-score, taking into account the number of True Positive (TP), False Positive (FP), and False Negative (FN). Introducing Precision = $\frac{TP}{TP+FP}$ and Recall = $\frac{TP}{TP+FN}$, F-score

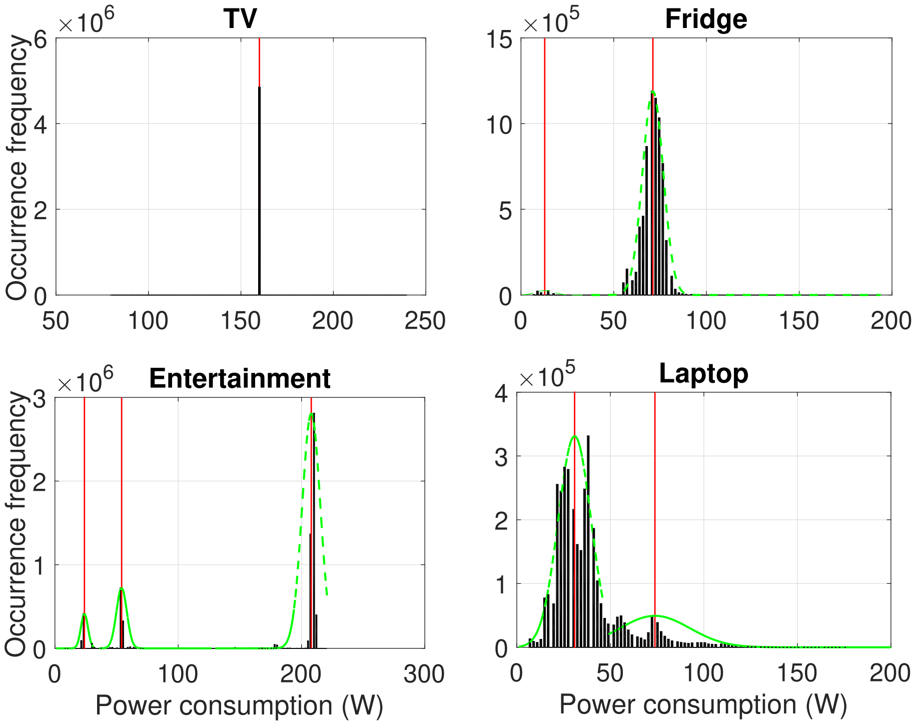


Figure 3.6 – Histogram and automatic states identification for some of the ECO dataset appliances. In red, the centroid of the identified clusters and in green, the corresponding estimated normal distribution.

represents their harmonic mean:

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- The Root Mean Square Error (RMSE) (W), defined as:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\widetilde{x}_i(t) - x_i(t))^2}$$

- The error in the total estimated energy $\frac{\Delta E}{E}$ (%). As the user is more interested in the total energy used by its appliances rather than their instantaneous power, the error in aggregated estimated energy use has been evaluated. This quantity is computed over the N_a appliances that influence the aggregated signal:

$$\frac{\Delta E}{E} = \frac{\sum_{i=1}^{N_a} |\widetilde{E_i} - E_i|}{\sum_{i=1}^{N_a} E_i}$$

where $\widetilde{E_i}$ and E_i stands for the inferred and real energy consumption of appliance i over the disaggregation period, respectively.

A definition of accurate TP and inaccurate TP have been suggested by Kim et al. [92] to make the distinction between the recognition of an active ON state and an accurate power

Table 3.2 – Comparison between binary and multi-state FHMM disaggregation results, trained for 30 days

Appliance set	**ON/OFF**			**Multi-state**		
	$\frac{\Delta E}{E}$	F_a-score	RMS	$\frac{\Delta E}{E}$	F_a-score	RMS
TV	0.9	0.996	4.11	**0.3**	0.999	0.54
Laptop		0.966	7.28		0.984	2.22
TV	2.5	0.995	4.22	**0.4**	0.990	7.84
Lamp		0.972	9.81		0.907	7.68
TV	1.7	0.991	5.90	**1.1**	0.991	5.84
Entertainment		0.836	13.26		0.977	6.91
Laptop	**1.5**	0.892	11.77	2.0	0.889	11.40
Entertainment		0.831	15.28		0.868	9.56
Stereo	17	0.930	13.68	**3.7**	0.802	13.49
Laptop		0.780	16.60		0.770	9.57
Entertainment		0.885	25.44		0.894	8.91
TV	**4.4**	0.978	8.45	5.8	0.994	4.50
Laptop		0.892	11.12		0.857	12.52
Stereo		0.932	10.86		0.722	13.10
TV	2.9	0.944	14.90	**2.1**	0.976	9.81
Laptop		0.884	12.22		0.885	11.80
Entertainment		0.836	13.25		0.866	9.01
TV	53.4	0.984	8.00	**13.9**	0.948	22.93
Lamp		0.698	46.61		0.260	29.54
Entertainment		0.438	94.60		0.933	51.18
Stereo		0.788	17.94		0.915	14.13

estimation. Since each state mean and standard deviation are estimated by the multi-state identification algorithm, a more precise definition can be derived: a TP is said to be accurate if

$$\frac{|\widetilde{x}_i(t) - x_i(t)|}{x_i(t)} \leq 3 \cdot \sigma_j$$

where j is the state corresponding to $\widetilde{x}_i(t)$, otherwise it is classified as an inaccurate TP. The corresponding accurate F-score is used in the experiment results, referred to as F_a-score.

The binary and multi-state FHMM-based power disaggregation methods have been applied to various groups of appliances and various lengths of training/disaggregation periods. Given the same period's length, the lowest value of $\frac{\Delta E}{E}$ of the binary model has been selected among all the results and the corresponding multi-state model has been recorded. By doing so, the advantage of using more than one active power state may be highlighted.

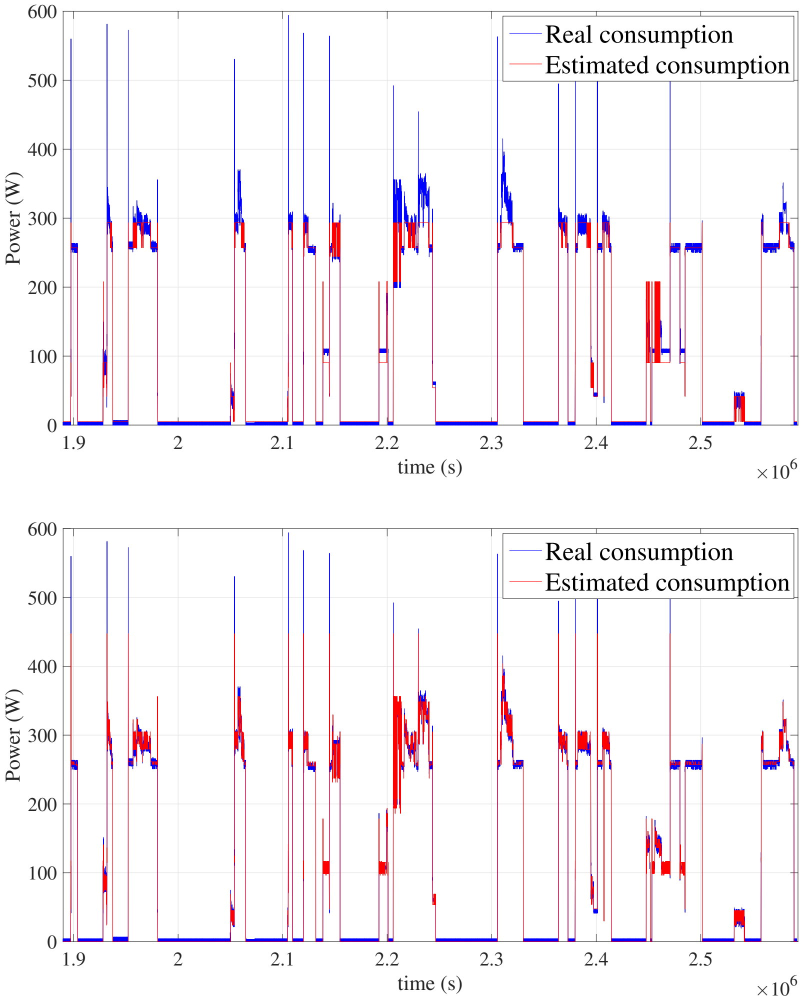


Figure 3.7 – Snapshot of outlet power consumption and its estimation for the set {Stereo, Laptop, Entertainment}. (Top) On/OFF modeling (Bottom) Multi-state modeling

Table 3.2 lists the results for 30 days of training and a disaggregation period of 30 days. A time-series example of the disaggregation result is shown in Figure 3.7. It can be seen that multi-state modeling generally outperforms binary modeling in terms of energy estimation and RMSE. The disaggregated time signal analysis shows better coverage of the multiple states taken by the outlet appliances. The F_a-score is yet not much improved by the multi-state modeling. This is due to the fact that the variance taken by the binary model state is generally larger than the variances in each state of the multi-state model, leading to less inaccurate TP in the binary F_a-score. Extremely low values of the F_a-score highlight the degradation of the disaggregation results based on multi-state modeling as an excessive number of states define the appliances. While the sum of the estimated signals gets closer to the real signal, the large amount of possible state combinations leads to erroneous individual signals. Other

Table 3.3 – Comparison between binary and multi-state FHMM disaggregation results, trained for 60 days

Appliance set	**ON/OFF**			**Multi-state**		
	$\frac{\Delta E}{E}$	F-score	RMS	$\frac{\Delta E}{E}$	F-score	RMS
Stereo	7.8	0.929	12.32	**1.4**	0.884	14.60
Laptop		0.858	11.44		0.898	10.78
Entertainment		0.816	17.47		0.913	10.49
TV	3.2	0.958	13.40	**0.9**	0.972	11.83
Laptop		0.867	11.11		0.884	9.82
Entertainment		0.828	12.67		0.930	7.53
TV	**2.8**	0.979	8.23	4.8	0.988	9.967
Laptop		0.895	9.70		0.681	11.14
Stereo		0.932	11.74		0.893	12.32

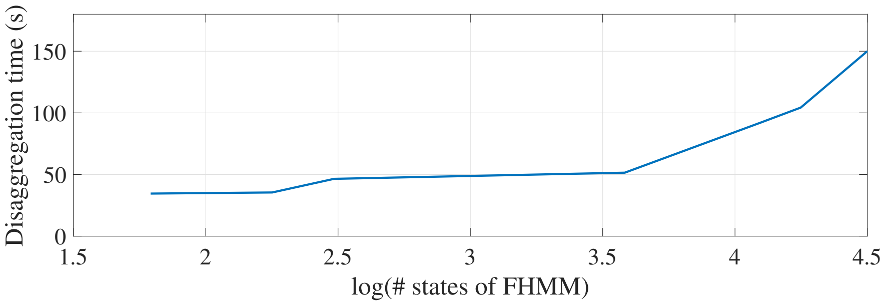


Figure 3.8 – Computation time (in seconds) of the load disaggregation process as a function of the logarithm of the total number of states of the combined FHMM

issues appear when frequent switching peaks populate the histogram, leading to a dedicated state. This short peak state might be erroneously identified in the presence of concurrent high consuming loads. The aforementioned issues could be mitigated by the modeling of the state's duration, especially the peaks, within a Semi-Hidden Markov Model. Moreover, a Conditional Factorial Markov Model could also improve the disaggregation results as correlated appliances are active at the same time [92].

The training period substantially influences the disaggregation efficiency for both the binary and the multi-state models. Table 3.3 shows the disaggregation results for a training period of 60 days and a disaggregation period of 30 days. Compared to Table 3.2 where the training period is two times smaller, the multi-state model behaves better in all cases. Indeed, the training phase might have detected new frequent states and filtered out others less frequent.

Nevertheless, the disaggregation process of the FHMM in the *NILM-Eval* toolbox grows exponentially in time as the number of states per appliance increases. Figure 3.8 shows the experimental disaggregation time as a function of the total number of states that the FHMM model has to deal with. It was run in an Intel® i7-6700-based machine, with 32GB DDR4 memory. Therefore, increasing the number of states per appliances comes at a higher cost with respect to computation time, as well as the required memory. This might explain why a binary ON/OFF model generally suffices for load disaggregation purposes when coping with a large number of appliances and large disaggregation periods. However, applications dealing with less than five appliances might prefer multi-state modeling for better accuracy in the results.

3.4.3 Activity set identification

The algorithm presented in Section 3.3 and used for energy disaggregation allows identification of intrinsic power states of the loads independently from user behavior. As the building occupants use the various appliances, specific patterns can be identified such as when the appliances are used and what power modes are preferred in each of them. The temporal use of a specific appliance is defined in a probabilistic way over 24 hours. To get their characteristics and probability, the entire historical time-series in analyzed. First, all the consumption points are represented by their discrete modes (m_1 to m_k) instead of continuous values. This chronological sequence of power modes is then compressed into an linked list to reduce the memory need. Activity is then identified whenever a succession of non-zero modes is observed for more than a specific threshold of time, typically a few dozens of seconds. Daily activities a_i throughout the dataset can thus be extracted, defined by their starting time $t_{i,s}$, duration d_i, and succession of modes.

The set of raw user activities $\{a_i\}$ is then processed to determine a condensed set $\mathscr{A}$ of typical daily activities. Essentially, one tries to extract a set $\mathscr{A}$ as small as possible, gathering similar past activities together. To do so, scatter plots of raw activities have been created, as depicted in Figure 3.9 for three different loads of the ECO dataset. In the left column, the activity starting time is plotted against its duration, while the right column plots the activity ending time against its duration. The identified clusters and centroids (black crosses) are the result of iterative k-means algorithms, with a cluster selection procedure similar to the mode identification algorithm presented previously, involving the DB index. As the couples {starting time, duration} sometimes lead to a large number of clusters with a high variance in duration and starting time, the "ending time" feature has been preferred in these cases. The selection between the two features - starting or ending instant - is done by looking at the DB index and total variance of both solutions. Appendix A.4 includes more activity clustering examples.

Clustering the activities allows us to obtain a small set $\mathscr{A}$ of representative activities. For each of them, normal distributions of their duration and start/end time instant are computed from the corresponding data points they contain. As not all the activities happen every day, their probability to occur must also be computed, based on the data point density of their corresponding cluster. Furthermore, the linked lists of mode switching events lead to the

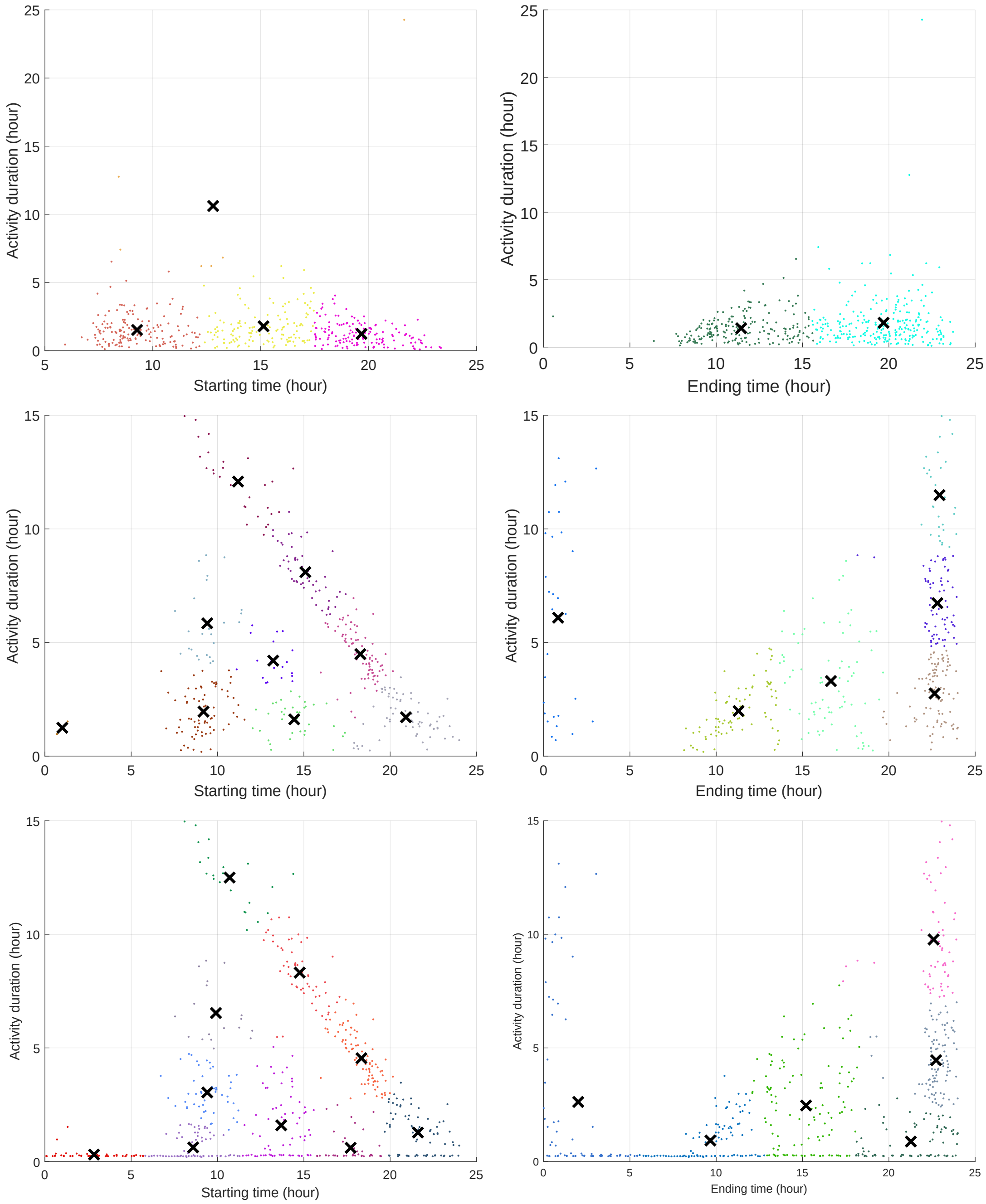


Figure 3.9 – Activity clustering by activity duration and (left) starting time or (right) ending time. From top to bottom rows: laptops, entertainment system, and stereo system.

creation of a set of sequence modes $\mathscr{S}$ that define the transitions of modes within each activity. The occurrence frequency of each mode fills the diagonal of the transition matrix in $\mathscr{S}$ and switching from mode i to j fills the corresponding cell in that matrix. Similarly, looking at the starting modes in each activity leads to the determination of starting mode probability.

3.5 Conclusion

In this chapter, we presented a new method to automatically extract power states and user activities of common building appliances, based on plug-level historical data. A *load profile identification* algorithm has been implemented for the automatic detection of the optimal number of states of the appliances under test and it then determines the model parameters in each of the states. The algorithm is based on the k-means clustering algorithm, for which the number of clusters is iteratively increased. A clustering quality index, penalizing the higher number of states, has been derived and used for state selection. Similarly, k-means clustering helped extract common user activity patterns in the consumption data. This results in a compact structure of representative user activities on their appliances.

The multi-state power modeling was then experimented on the ECO dataset by enhancing the FHMM available in the *NILM-Eval* toolbox, for 1 Hz disaggregation tests. Compared to the basic ON/OFF binary model, multi-state FHMM-based disaggregation reduces significantly the RMS error of the estimated appliance power. Modeling the appliances with more than a single active state allows the algorithm to better match the actual consumption. The proposed modeling, therefore, improves the accuracy of the total estimated energy consumption per appliance. Yet, as the set of possible states at each time instant grows bigger, this comes at the expense of larger computation time. This represents an interesting improvement for a more accurate user feedback towards better demand-side management, as the building occupants would get a better view of their appliances' energy consumption.

Part II

Advanced Integration of Smart-Buildings into the Smart-Grid

4 Smart-Building Energy Management

Buildings' electrical loads, energy storage systems, and local generation infrastructure form a complex heterogeneous fleet of flexible entities. The proper management of such energy flexibility requires the deployment, at the building premises, of an Energy Management System (EMS) that continuously ensures the appropriate use of the controllable part of the energy. Balancing both the building owner benefits and the the smart-grid objectives while maintaining user comfort, the building EMS should carefully be designed. Furthermore, it should seek compatibility with existing Building Data Management System (BDMS) for a large scale adoption.

The main **highlights** and **contributions** of this chapter are:

- A review of energy management algorithms in building, with a focus on Model Predictive Control (MPC) and existing tools to deploy them in buildings.
- We developed OpenEMS, an open-source building Energy Management System for smart-grid applications. The modular and generic architecture of the software allows a plug-and-play connection with existing BDMS. Programmable by the user, many advanced energy management algorithms can be implemented through its object-oriented structure.
- Simulations based on the Minergie test case highlight the modularity of OpenEMS for building energy management, by comparing Rule-Based Control (RBC) with MPC.
- The open-source tool, coded in Python, is available at [116] and common energy management strategies tailored to OpenEMS can be found in [117].

4.1 Energy Management in Buildings

The control of energy in buildings has received plenty of attention in past decades. Initially driven by the need to reduce the total amount of energy consumed at the building site, research has recently moved towards the energy management of flexible entities for dynamic grid support. The requirement of the grid utility is expressed through the notion of DSM [118]. DSM represents a set of programs aimed at changing the consumption of buildings, categorized into Energy Efficiency (EE) and DR. On the one hand, EE programs encompass passive techniques to reduce the overall building energy demand: enhanced building envelopes, efficient appliances, double-gazing windows, etc. On the other hand, DR programs represent active methods to shape the daily building power profile (residential or commercial) which can itself be subdivided into time-based and incentive-based programs. Time-based signals define the retail price of electricity, which can be as static as Time-of-Use (TOU) tariffs or as dynamic as Real-Time Pricing (RTP). Capacity market, interruptible, emergency, and Direct-Load Control (DLC) programs fall mainly into the event-based category.

Electricity prices and other DR signals generated at the grid side must be properly understood and translated into local actions at the building premises. To this end, the use of energy management algorithms in buildings has been intensively studied [6, 119, 120, 121, 122]. Traditional control methods such as rule-based decisions acting on a local part of the system have, therefore, progressively led the way to more advanced optimization-based strategies that take the whole building into account [123, 120, 122]. These strategies generally leverage a given model of the building to minimize the electricity cost paid by the user while still ensuring their comfort. In [6], authors review the common models that are used for advanced model-based control in building. They emphasize the importance of predicting future environmental conditions, as they directly impact the models. Vardakas et. al. [122] differentiates the optimization-based approach to tackle DR between load scheduling and energy management. While load scheduling models all the flexible loads as deferrable (shiftable) loads, energy management dives into the details of each of the underlying models and constraints.

The load scheduling (or appliance commitment) task consists of deciding on the most appropriate starting time instants of a set of deferrable loads, ahead of time [124, 125, 126]. Mohsenian-Rad et al. [127] and Law et al. [78] formulated the load scheduling task as a linear programming problem that can be solved by the Interior Point method or commercial solvers. In [128], the problem is approached via the knapsack method and eventually boils down to a linear programming formulation. Integration of ESS along with deferrable loads is presented in [31] and proves to reduce the daily Peak-to-Average Ratio (PAR). Authors in [129] relaxed the linear programming formulation for EWH temperature management and integrate a heuristic to schedule the load according to RTP. As a deterministic formulation might not be realistic, Chen et al. [130] consider uncertainties in household appliance operation time and fluctuating renewable generation in their load scheduling formulation. Compared to basic load models used in the aforementioned works, more realistic multi-state models have been used in [131], in addition to a method to mitigate the load uncertainty at the grid level.

Beyond load scheduling, more advanced models can be used to create a forecast of the building consumption ahead of time. Hubert et al. [30] included knowledge of the building's physics along with a deferrable load scheduling task to generate a 24-hour Mixed-Integer Linear Programming (MILP) problem. Comfort preferences and EV SoC could directly be included in the problem constraints set. In [132], authors formulated both a robust and stochastic optimization problem to schedule a set of deferrable appliances, an AC, an EV, and an EWH. To cope with the intraday changes in electricity prices, authors in [133] proposed to update the parameters and re-run the MILP problem, considering five types of loads. Particle Swarm Optimization (PSO) has been applied in [134] to solve the day-ahead building energy planning problem, in the presence of deferrable appliances, ESS, and local renewable energy.

Day-ahead optimization leads to optimal planning based on assumptions prone to forecast uncertainties. MPC mitigates this issue by enabling a real-time controller that solves a reduced optimization problem at every control step (typically one hour to five minutes). The control method has solely emerged recently for smart-building applications thanks to the ubiquitous ICT and cheaper computational power. Afram et al. [135] reviewed the use of MPC for HVAC control systems, from simulation to experimental fields. The gain in energy efficiency has been shown to be case-specific, as high as 30-70%. Compared to other traditional controllers, MPC can directly incorporate user comfort and physical preferences into the optimization problem formulation and the optimization function directly reflects user costs and/or grid benefits. Nevertheless, as the controller is mostly based on the forecast, a good prediction of the model parameters and influencing disturbances is required. The thermal model is generally the main element of the MPC formulation. Authors in [17] detailed a generic intermediate complexity model for room thermal comfort automation. However, simpler RC models are generally used for catching the main thermal dynamics of the building [136, 33, 137, 6], leading to a less complex problem to solve. To generate a building model tailored to predictive control, [138] developed the BRCM MATLAB toolbox. The toolbox outputs a discrete-time bilinear state-space model of the building, from EnergyPlus input files [38]. Furthermore, building occupants play an important role in the quality of the energy control algorithm because of their natural unpredictability. A review of the models used to predict user occupancy and behavior for MPC purpose has been conducted in [139]. Machine learning has also been applied to the field of MPC. In [140], authors used ANN to derive a model of the building to integrate it into an MPC at residential level. Approximate model-predictive building control was proposed in [141], where the model is simplified by the use of a time-delay neural network. The use of machine learning techniques helps reduce the large hardware and software requirements necessary for MPC.

Due to its structure, MPC is the ideal tool to enable a building-to-grid framework. In [142], an MPC-based strategy decides whether to involve the building or not into the Day-Ahead Demand Response program. During the next day, price of electricity is directly use to drive the building power consumption. Maasoumy et al. [143] proposed a contract-based framework between the building and the utility to leverage the flexibility offered by the HVAC fan. An optimal baseline is initially computed by the building, and then during the day the MPC leverages the thermal inertia of the system to increase building benefits through the tracking of a utility signal. Combining both price-based and event-based DR programs, Knudsen et al. [144] developed a two-stage MPC. The innovative structure allows the building to decide on optimal bids offered to an aggregator. Authors in [145] managed to decentralize the MPC problem within a building, leading to a consensus-based algorithm among to participants to agree on a maximum value of power peak.

Parallel to MPC, MAS has experienced a keen interest in decentralizing and distributing the energy management strategy. Though it is more often encountered in smart-grid applications, it offers interesting practical features for building energy management. Authors of [119] and [54] provide thorough reviews of MAS tailored to building control and energy optimization. The main idea consists of segmenting the complex building energy management system into sub-modules and letting independent agents interact to solve the whole problem. In [54], a MAS-based framework relying on four layers is proposed: a simulation layer, knowledge information layer, network data acquisition layer, and action layer. Hurtardo et al. [146] implemented a MAS control framework with the JADE framework able to take into account the grid key performance indicators to drive the energy management logic.

Beyond the design of innovative energy management algorithms, their practical implementation in the building EMS has also been investigated. Their simulations are generally carried out in MATLAB, unsuitable for real-time light embedded energy management. Zhou et al. [147] presents the concepts and configuration of residential EMS. They highlight the need for a (wireless) network of sensors and actuators connected to a main panel of EMS. The widespread smart-meter is referred to as the home gateway to the external utility for bidirectional communication purposes. As the set of loads, energy storage, and local renewable energy infrastructure run during the day, their power must be monitored, logged, controlled, and managed by the EMS. Authors in [148] detailed a multi-modal residential EMS for DR. The *device* object is at the center of their database design, linked to *measurement*, *controller*, and *room* objects. A *prediction* object collects external forecasts useful for *control*. A service layer offers a Graphical User Interface (GUI) for data extraction and user involvement in the EMS decision. An EMS based on ZigBee is introduced in [149], separating the design into sensing infrastructure, context-aware, and service management. Leveraging home area network systems, authors in [150] proposed a residential EMS to connect building loads to an external aggregator. The design of a Building Operating System (BOS) was described in [151] that enabled the test of multiple strategies, including MPC, on a university campus. Tools like VOLTTRON [152] and XBOS [60] coupled with MPCPy [153] can play the role of a combined robust BDMS-EMS platform for energy monitoring and optimization in buildings.

In this chapter, we present the design and implementation of a light plug-and-play EMS that can seamlessly connect to existing deployed BDMS. The tool is called Open Energy Management System (OpenEMS) and is entirely coded in Python. Unlike [149, 148, 150], we have decided to separate the sensor network and data storage from the EMS design. This allows better modularity and renders the tool more generic. Like MPCPy, it relies on external

BDMS infrastructure to get the necessary data for running the algorithms. Nevertheless, in addition to MPC, the OpenEMS can deploy other control algorithms as specified by the user. From a software design standpoint, the tool differs from MPCPy by being a standalone process that automatically plugs into the BDMS to enable energy management of its building.

The rest of this chapter is organized as follows. Section 4.1.1 concludes the literature review by describing the mechanisms of MPC in buildings and Section 4.2 further derives grid-oriented MPC formulations. The multi-threaded architecture of the OpenEMS is presented in Section 4.3. A case study involving a Minergie building demonstrates the usefulness of the tool in Section 4.4 by implementing and comparing two control strategies. Finally, Section 4.5 concludes the chapter.

4.1.1 Model-predictive control applied to building energy management

Definition 4.1.1. MPC - A Model Predictive Controller (also referred to as MPC controller) is a controller that uses the knowledge of the model of the system it intends to control to periodically compute setpoints aimed at the system. The setpoints sent to the various controllable elements constituting the system result from solving an optimization problem given the model of this system, constraints on its components, and data forecast over a limited future horizon.

Fig. 4.1 depicts an MPC controller applied to a generic building environment. In a smart-building context, the corresponding model in the MPC typically encompasses the thermal dynamics, batteries & EV, controllable loads, user behavior, and PV production. Constraints are added to the physical components, as well as to ensure temperature comfort. Input forecasts generally include environmental conditions, occupant behavior, and electricity prices. Solving the optimization problem generally implies the minimization of an economic cost function or the consumed energy over the next horizon of time H. By denoting the cost function $J[h]$ at each time instant h, the generic MPC formulation for building energy management has the following form:

$$\begin{aligned} \min_{u} \quad & \sum_{h=0}^{H-1} J[h] \qquad (4.1)\\ \text{s.t.} \quad & \text{building state at time } h=0 \\ & \underline{\forall h = 0 \ldots H-1}: \\ & \quad \text{building model at time } h \\ & \quad \text{constraints on building entities/user at time } h \end{aligned}$$

Out of the resulting control variable $\boldsymbol{u}^*$, the first value is practically applied as an input to the system.

The MPC controller receives feedback from the building in a closed-loop fashion. Sensors deployed at the building premises help construct part or the entire MPC model state to be

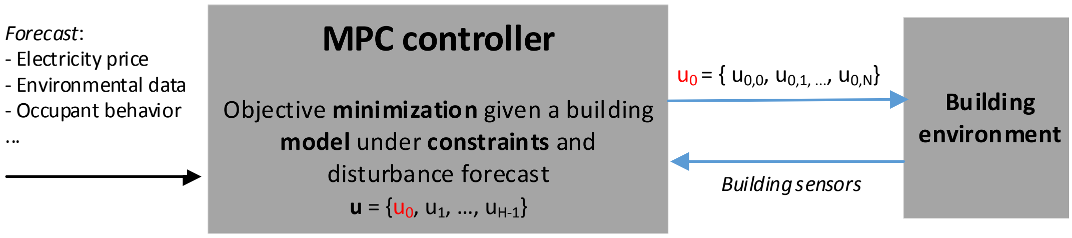


Figure 4.1 – MPC applied to buildings: system overview

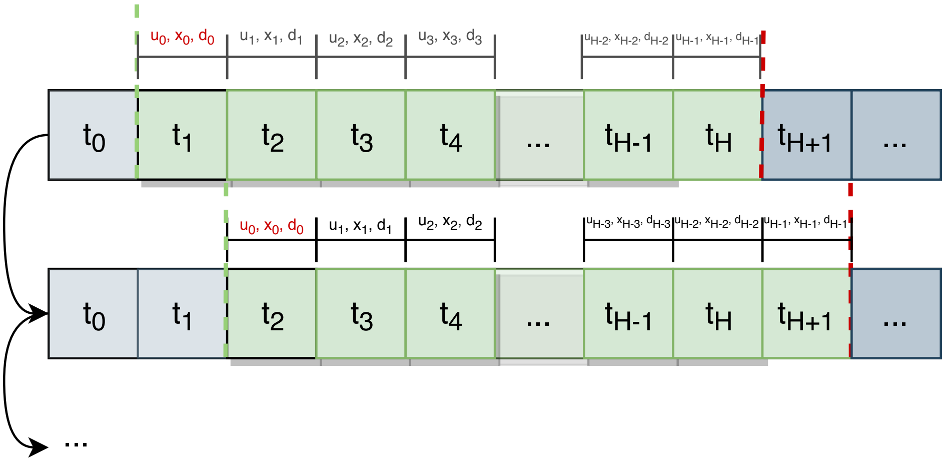


Figure 4.2 – Illustration of MPC horizon of optimization.

used for the subsequent optimization problem. Fig. 4.2 illustrates the MPC operation over time. At time t_1, the MPC model state x_0 is updated after reading the sensors data, and forecast data $[d_0,...,d_{H-1}]$ is retrieved. The cost function is then built over the horizon H, along with the building model and constraints over the horizon H given as constraints to the problem. Solving the problem (4.1) at time t_1 outputs an optimal trajectory for the building state $[\boldsymbol{x_0},...,\boldsymbol{x_{H-1}}]$ provided that the optimal input $[\boldsymbol{u_0},...,\boldsymbol{u_{H-1}}]$ was applied to the model over horizon h. Practically, the set of input commands $\boldsymbol{u_0}$ are applied to the building system at time t_1. Then, at the next control instant t_2, the updated state x_0 resulting from the application of $\boldsymbol{u_0}$ is retrieved along with new forecast data in order to repeat the same optimization problem at time t_2. A new input command set $\boldsymbol{u_0}$ will be computed, applied to the building at time t_2, and so on.

A note on control versus energy management

It is important to differentiate the setpoints resulting from MPC-based energy management and local controllers' actions. A local controller acts on a specific part of the building to regulate comfort or on a building component (e.g., temperature setpoint tracking, battery power regulator). These controllers are, therefore, listening to signals emitted by the EMS (temperature, power, on/off states, etc) and ensure the local system they control follows the received signal. Whereas the EMS relies on advanced model-based algorithms to decide on these setpoints, local controllers generally apply simpler rule-based or Proportional Integral (PI) control. There is also a significant difference in the order of magnitude of the operational timescale. Typical periods for EMS setpoints updated range from one hour to five minutes. Concerning the local controllers, they must work at a higher frequency to properly track a signal; this frequency depends on the dynamics of the system it controls.

In this thesis, the focus is set on the methods of energy management. Consequently, one assumes that local controllers are present in the building to ensure the tracking of EMS-generated setpoints.

4.2 MPC as a Means to Provide Grid Services

MPC has been introduced in the previous section as a powerful means to manage a building's energy consumption, being aware of its numerous stakeholders. In this section, we further discuss how an MPC-enabled smart-building can seamlessly implement common grid services. MPC formulations are developed to practically incorporate various DR programs' specification into their constraints and objective function. We consider that a building model is known with corresponding constraints and that the MPC outputs on the flexible signal u.

4.2.1 Price responsive smart-buildings

Price-based MPC (or economic MPC) for smart-buildings are commonly found when studied in a smart-grid context [154, 155]. The fundamental idea is to produce a set of optimal commands for the flexible entities that minimize the cost of energy in a given horizon [1]. The corresponding generic MPC formulation is expressed as follows:

$$
\begin{aligned}
\min_{u} \quad & \sum_{h=0}^{H-1} c[h]\; e[h] && (4.2)\\
\text{s.t.} \quad & \underline{\forall h = 0 \ldots H-1}: \\
& \text{building model at time } h \\
& \text{constraints on building entities/user at time } h
\end{aligned}
$$

where $c[h]$ is the price of energy ($\$/kWh$) at time period h. It could be as simple as day/night or a TOU tariff defined by the electricity retailer through a contract and fixed for a long time. Additionally, some utilities propose DR programs like Critical Peak Pricing (CPP) that increase the price of electricity for specific periods, generally announced a day in advance, while offering price compensation the rest of the time. More dynamic prices would consist of using

---

[1] Depending on the price signal, this does not necessarily translate into the minimization of the overall energy used.

market RTP or local aggregator/microgrid prices updated frequently as a function of local demand and generation. In either case, the structure of MPC allows for the easy integration of these prices into the objective function. Finally, TOU demand charges may apply to large commercial buildings, especially in the US. Balancing both the costs due to energy charges ($\$/kWh$) and the demand charge ($\$/kW$) is a non-trivial task and will be the focus of Chapter 5.

4.2.2 Event responsive smart-buildings

While energy prices represent a great vector for decentralized demand response, they might lead to a lack of guaranty on the change in energy/power consumption [156]. To this end, event-based DR, also referred to as incentive-based or physical DR, programs provide the grid-side agent with a better guaranty on demand behavior, as both parties are bound by a contract. The response to a DR event will be evaluated against a baseline by the grid-side agent (e.g., the utility) to verify whether the building responded appropriately.

Definition 4.2.1. Baseline - A baseline refers to a power consumption profile of a building used for reference purposes. When used in a DSM context, a baseline represents the building consumption that would be expected in the absence of a DR event.

Let us denote this baseline power by $\boldsymbol{s^r}$, i.e., a vector of power consumption over a defined horizon H. Furthermore, we consider that the building's EMS optimizes an objective function $f_c(e[h]$ (e.g., energy consumption or cumulative energy cost).

Load shedding/shifting/capping

Load shedding constitutes a widespread DR program to impose a time-limited decrease in demand during critical periods of the year. The reduction in power $\Delta P_{DR}[h]$ is computed with respect to the baseline $\boldsymbol{s^r}[h]$ that would have been expected without the event. An additional constraint must thus be incorporated into the MPC formulation:

$$\min_u \sum_{h=0}^{H-1} f_c(e[h]) \tag{4.3}$$

$$\text{s.t. } e[t_d] \leq s^r[t_d] - \Delta P_{DR}[t_d] \; \forall t_d \in \mathcal{H}_d \tag{4.4}$$

$$\underline{\forall h = 0 \dots H-1}:$$

building model at time h

constraints on building entities/user at time h

Constraint (4.4) prevents the power consumption from rising above$s^r - \Delta P_{DR}$ at strategic time instants $\mathcal{H}_d$ that activate the constraint. Out of these sets of time periods, the constraint is inactive and there is no restriction on power consumption.

As power demand is limited by load shedding programs for some periods of time, this might induce a *rebound effect* due to the need to compensate for the reduced energy used [157]. This effect especially occurs when dealing with loads linked to thermal inertial. The concept of *load shifting* solves this issue by specifying periods of increased demand ($\mathscr{H}_u$) and decreased demand ($\mathscr{H}_d$) which can be inserted in the MPC as follows:

$$
\begin{aligned}
\min_{u} \quad & \sum_{h=0}^{H-1} f_c(e[h]) && (4.5)\\
\text{s.t.} \quad & e[t_d] \leq s^r[t_d] - \Delta P_{dr}[t_d] \;\forall\, t_d \in \mathscr{H}_d && (4.6)\\
& e[t_u] \geq s^r[t_u] + \Delta P_{dr}[t_u] \;\forall\, t_u \in \mathscr{H}_u && (4.7)\\
& \underline{\forall h = 0 \dots H-1}: \\
& \quad \text{building model at time } h \\
& \quad \text{constraints on building entities/user at time } h
\end{aligned}
$$

While constraint (4.6) is similar to constraint (4.6) in formulation (4.3), constraint (4.7) enforces the consumption to be higher than the baseline by ΔP_{dr}. It is worth noting that these new sets of constraints might lead to an infeasible problem, in which case they should be expressed as soft constraints.

Instead of limiting the power consumption with respect to the building baseline, the grid-side agent might directly specify the absolute amount of power the $e_{max}[h]$ building should not exceed at time h:

$$
\begin{aligned}
\min_{u} \quad & \sum_{h=0}^{H-1} f_c(e[h]) && (4.8)\\
\text{s.t.} \quad & \underline{\forall h = 0 \dots H-1}: \\
& e[h] \leq e_{max}[h] && (4.9)\\
& \quad \text{building model at time } h \\
& \quad \text{constraints on building entities/user at time } h
\end{aligned}
$$

Load following

Unlike the aforementioned event-based programs that aim to punctually change power consumption compared to a given baseline, *load following* programs continuously shape the building power profile. To this end, the building is provided with a signal $\boldsymbol{s^r}$, that it must track as much as possible. This signal can either be an absolute power directly used for optimization or a normalized vector representing a relative expected consumption shape. In the latter case, the building itself is in charge of constructing the power signal that it has to track given the knowledge of the maximum power it will consume in the signal period. The corresponding MPC formulation can be derived as follows:

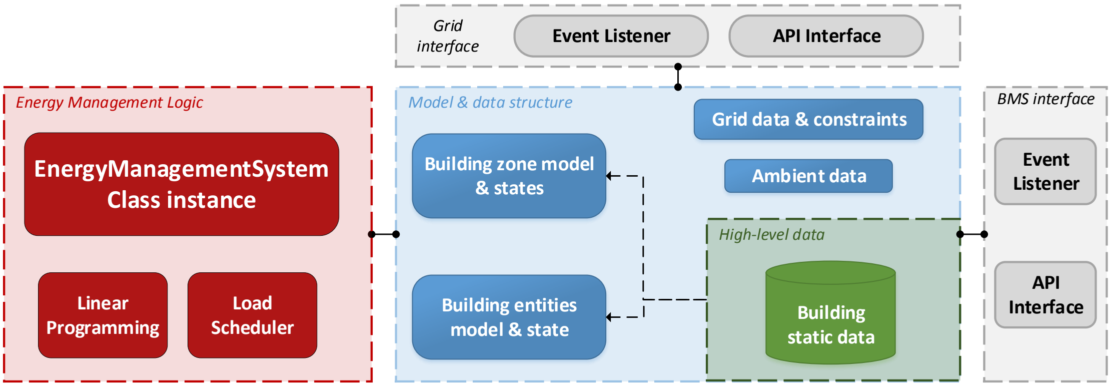


Figure 4.3 – OpenEMS distributed architecture

$$\begin{aligned} &\min_{u} \sum_{h=0}^{H-1} (e[h] - s^r[h])^2 \\ &\text{s.t. } \underline{\forall h = 0 \dots H-1}: \\ &\qquad \text{building model at time } h \\ &\qquad \text{constraints on building entities/user at time } h \end{aligned} \tag{4.10}$$

The quadratic objective function in Eq. (4.10) strives to reduce the least-square error between the forecast power consumption $\boldsymbol{e}$ and the desired one $\boldsymbol{s^r}$. If the special case where $H = 1$, only the next power consumption value is specified and must be followed. This can arise for providing ancillary services such as Frequency Control Regulation (FCR) where the building is asked to track a signal updated as often as four-second intervals to help regulating the system frequency [158]. Applications to energy arbitrage lead to a large value of H (typically 96 for a 24 h horizon with a 15 min precision) and intends to match as much as possible local generation with local demand; this scenario will be the focus of Chapter 6.

4.3 OpenEMS: a Generic Open-Source EMS

The architecture of the OpenEMS is depicted in Fig. 4.3. It has been designed as a multi-threaded process to decouple the input/output interfaces with external applications from the internal core functions to handle the building energy. Two interfaces, *BMS interface* and *Grid interface*, allow the core of the OpenEMS to leverage data coming from (sending data to) the BDMS and the grid utility, respectively. They both use the Application Programming Interface (API) offered by these applications, to retrieve metadata about the building, its entities, and the grid signals that will be sent in real-time. Mainly, the BDMS' API provides the OpenEMS with useful data to represent the building from an energy management point of view.

The *Model & Data Structure* module receives static data from the *BMS interface* already pre-processed by the latter. This data mainly contains building geometry, zones and their thermal

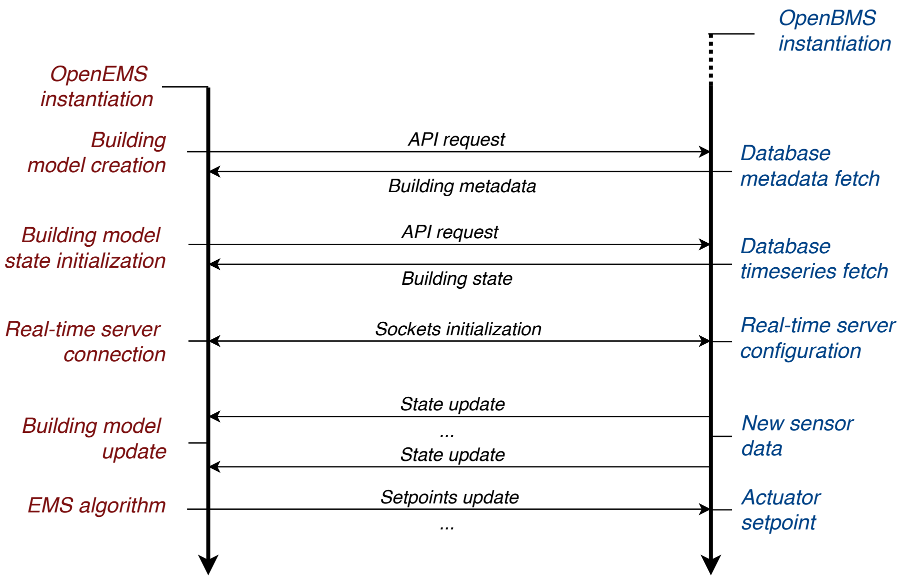


Figure 4.4 – OpenBMS-OpenEMS interactions timeline

interconnection, the energy-related entities information (loads, battery, generation), and other miscellaneous data (e.g., altitude, coordinates). It then structures them in a way that eases their manipulation for control purposes. Along with the building-specific data, structures co-exist to handle the exodata, such as the ambient conditions and information/constraints coming from the grid.

Finally, the *Energy Management logic* module may take advantage of the *Model & Data Structure* module to implement the core logic of energy management. The structure of this module has been created in a modular way that enables the users of the tool to program the management strategy themselves. In addition to building data, two advanced modules assist the energy management strategy designer in solving common problems: deferrable load scheduling carried out by the *Load Scheduler* module and model-based optimization formulation provided by the *Linear Programming* module.

Practically, *Model & Data Structure* and *Energy Management logic* are run in a unique thread, implemented by the Python class *EnergyManagementSystem*. This class inherits from the Python module *Thread* to infinitely run for a periodical management of the building energy. It collects real-time building data and other exodata from the *Model & data structure* module that connects itself to the aforementioned interfaces. The complete documented code of the OpenEMS is available at [116].

4.3.1 BMS interface

Fig. 4.4 shows the timeline of interactions between the OpenEMS (in red, left) and the building's BDMS (in blue, right), from the initialization to the real-time update of setpoints. Practically, this interaction is enabled by the *BMS interface* thread. It has the mission to decode the API responses and real-time data coming from the BDMS, as well as format API requests. Two main operations are performed by the *BMS interface*:

- API data decoding/encoding: different BDMS are likely to expose different APIs [7], forcing the *BMS interface* module to adapt itself to any of them. The first set of API calls aims to initialize the building static model by reconstructing the building geometry, its zones, and the interface separating the various zones. Then, it must obtain metadata about all energy-related entities present at the building site and link them with the zones.
- Sensors and actuators mapping: the OpenEMS core must be totally agnostic to the underlying technology and even to the notion of sensors and actuators. Both of these entities are fundamental for the BDMS and are generally linked to an energy-related entity or a comfort metrics. Hence, the *BMS interface* maps incoming data from sensors to the corresponding higher-level OpenEMS entities. For instance, new data from a power sensor of load x will update the current state in the object linked to that load, and a new temperature from an environment sensor will update the comfort state of the corresponding zone object. Similarly, setpoints updated on the OpenEMS objects are translated into commands to a specific actuator in the BDMS.

Due to the threaded architecture, inter-thread queues convey data in a bidirectional way, between the interface and the *Data Management* module of the OpenEMS.

4.3.2 Data management module

Pre-processed data coming from the *BMS interface* finds its way towards the *Data Management* module that encapsulates it into appropriate objects. Two root data structures, linked to static data in Fig. 4.3, separate the energy-related entities from the zones. Any data that refers to a state of power/energy is directed toward the corresponding energy-related object, while comfort data is stored at the level of its active zone. Detailed Unified Modeling Language (UML) diagrams of the classes making up the *Data Management* module can be found in Appendix A.3.2.

Fig. 4.5 provides a hierarchical overview of the OpenEMS energy-related entities.
The electrical loads are subdivided into *user-driven* (non-controllable) loads and *controllable* loads, as described in Chapter 1. Among the controllable loads, one finds the *deferrable* loads (shiftable but uninterruptible) and *interruptible* loads (shiftable and interruptible). Each of these subclasses is mapped to the corresponding class in the OpenEMS that defines

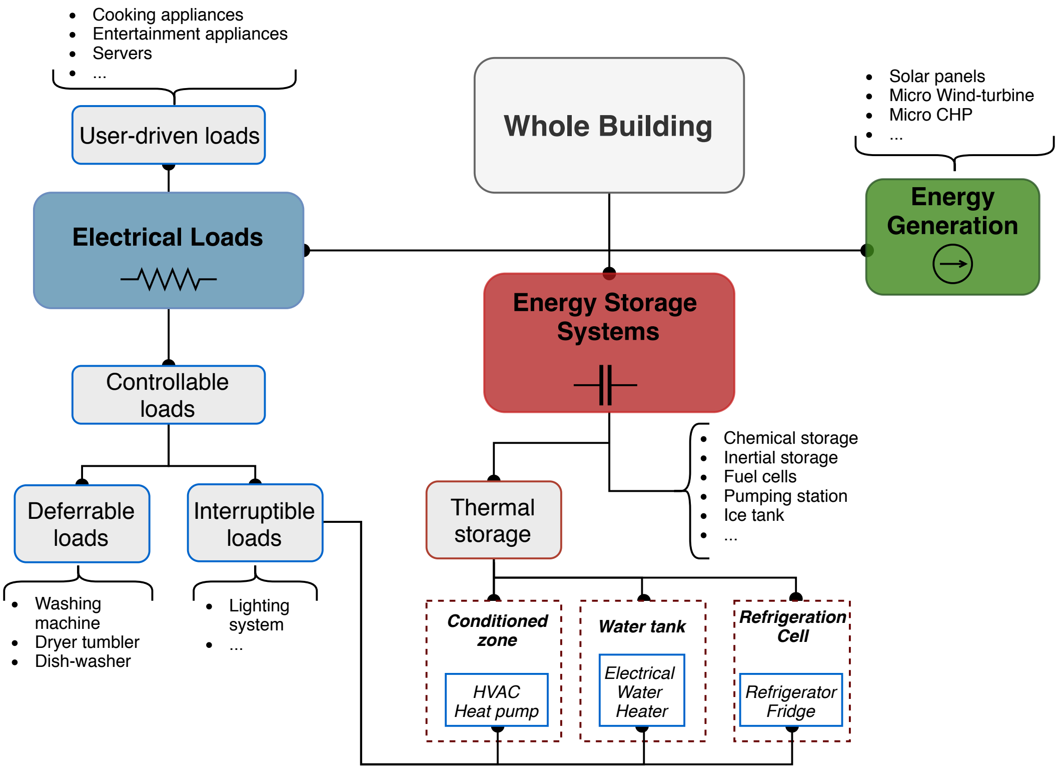


Figure 4.5 – Smart-building electrical entities categorization

parameters and useful methods to control them according to their capabilities, or prediction about their consumption.

Beyond reversible secondary energy storage (chemical, mechanical, etc.), the Energy Storage System category includes thermal storage linked to air zones, water tanks, or refrigerated cells that receive/lose heat from a thermal interruptible load. The storage capacity and leakage coefficient of zone thermal storage are closely linked to data stored in the second root data structure modeling the building geometry and storing the thermal comfort state and constraints. The third category contains data about local energy generation.

These energy-related entities must all implement the abstract method *getForecast()* that returns a power forecast for a specified time interval. Most of the uncontrollable loads and local generation infrastructure, as well as the deferrable loads, can provide this power estimation given environmental conditions or user-actions forecasting that will drive their behavior. In the case of a flexible model-based forecast, applying to thermal loads or batteries, this method will depend on the optimization results.

A zone object contains static information about its geometry and thermal model parameter of its interfaces with the adjacent zones, current comfort state, comfort constraints, and the loads acting on the zone. Like energy-related entities, any instance must implement the abstract method *getForecast()*. Whereas most of the zone (thermal) comfort forecasts are model-based and, therefore, depend on advanced optimization results, some zones, such as the outside environment, can provide a fixed forecast. To populate the latter, the *Ambient Data* module connects to an external forecast data provider that is periodically polled.

Finally, the *Grid data & constraints* module stores signals ready to be used for optimization, such as the price of electricity, coming from the *Grid interface*. This interface with the grid utility can either rely on a custom protocol or implement the widespread standard Open Automated Demand Response Standard (OpenADR) for message formatting [159].

4.3.3 Energy management core module

Data from the BDMS and grid utility is structured in handy objects in the *Data Management* module so core energy management strategies can leverage them in the *Energy Management logic* module. Practically, the Python class *EnergyMangementSystem* is run as a thread that instantiates the *Data Management* module and populates the structures by creating queues with the two interfaces. Periodically, the instance of that class calls the abstract method *update()* that must generate new setpoints targeted to the building entities.

The development and deployment of a new energy management algorithm, therefore, consists of creating a class that inherits from the *EnergyMangementSystem* and programming the *update()* method (as well as the *init()* to initialize the algorithm). To this end, the programmer can count on a panel of useful objects, methods, and advanced model-based libraries. Basic objects refer to the structures contained in the *Data Management* module, exposing the programmer to a building's list of entities and zones, their current state, constraints, and specific parameters. Useful methods include setpoints command creation and energy/comfort forecast.

Two libraries are available to the user for the advanced optimization problem formulation and solving. The *Load Scheduler* practically browses the set of available deferrable loads and generates an optimal triggering instant for each of them, given the scheduling cost function and time window (typically used day-ahead for the next 24 hours). In real-time, it automatically looks for the loads to trigger and generates the corresponding ON setpoint for the BDMS. In the case of advanced model-based energy management, the *Linear Programming* module digs into the details of the flexibility and forecast of available entities stored in the *Data Management* module to generate a set of optimization constraints, c, and a parametric cost function objective f. Both c and f can be inserted into a solver to create a sequences of optimal setpoints to apply to the building. The content and implementation of these libraries will be thoroughly described in the next section in the form of a case study.

The modular plug-and-play structure of the OpenEMS core allows its users to code a generic EMS strategy rather than a code tailored to a specific building as it is often encountered in the literature. Available samples of common EMS strategies tailored to the OpenEMS are available at [117].

4.4 OpenEMS Case Study: Energy Management of a Minergie Building

Introduced in Chapter 2, the simulated Minergie building is used in this section to emphasize the modularity of the OpenEMS. Two energy management strategies have practically been deployed in the tool, by inheriting the $EnergyManagementSystem$ class as explained in Section 4.3. The traditional on/off control encountered in most current buildings has been run against advanced MPC. While the traditional control solely ensures comfort to the user without considering grid services, the more complex MPC leverages the knowledge of the building model to provide grid services (price incentive). The next subsections describe the logic of each strategy and its implementation within the OpenEMS.

To test the management methods, the OpenEMS periodically sends new setpoints to the BDMS that conveys them to entities simulated by the Virtualization Engine (*vEngine*) introduced in Chapter 2. Both the *vEngine* and the OpenEMS get their static data from the BDMS, but use them differently. Namely, the *vEngine* reads data referring to virtual entities simulation that is hidden to the OpenEMS. Alternatively, the OpenEMS needs high-level information and forecasts not available to the simulation, to centrally generate the setpoints.

4.4.1 Baseline: traditional control

A traditional on/off energy manager was already used in Chapter 2, described by Algorithm 2. Practically, the strategy is implemented by instantiating the following $EMS\_TRADITIONAL$ class as the main OpenEMS core logic:

```
from ems_main import EnergyManagementSystem
from building_data_management.category_management.cat_config import *

class EMS_TRADITIONAL(EnergyManagementSystem):
  def __init__(self, **kwargs):
    ...

  def update(self):
    commands_set = []

    # Loop over the list of "zones"
    for r_id in self.building_data.zone_ids_list:

      temp_constr = self.building_data.zone(r_id).get_comfort_constraint(
                                              EMS_COMFORT_temperature)
      current_temp = self.building_data.zone(r_id).get_comfort_value(
                                              EMS_COMFORT_temperature)

      if current_temp > temp_constr.get_max(self.current_time):
        for hvac_id in self.map_zone_thermal_load[r_id]:
          commands_set.append(self.hvac_set_point(hvac_id, 0))
      elif current_temp < temp_constr.get_min(self.current_time):
        for hvac_id in self.map_zone_thermal_load[r_id]:
          commands_set.append(self.hvac_set_point(hvac_id, 100))

    # Loop over EVs
    phev_list = self.building_data.get_entity_list([
                                     EMS_CATEGORY_ENTITY_STORAGE_PHEV])
    for ev_id, ev_obj in phev_list:
      sp_max = ev_obj.max_power_charge()
      commands_set.append(self.batt_set_point(ev_id, sp_max))

    # Loop over deferrable loads
    def_list = self.building_data.get_entity_list([
                                     EMS_CATEGORY_ENTITY_DEF_LOAD])
    for l_id, l_obj in def_list:
      if l_obj.triggered_time is None and self.current_time >= l_obj.
                                     sched_time:
        commands_set.append(self.start_def_load(l_id))

    return commands_set
```

First, one has to import the main *EnergyManagementSystem* class that will provide the necessary objects to sense the building states. By inheriting from that class, the *EMS_TRADITIONAL* may access the BDMS object, "*self.building_data*" that contains all the necessary variables and methods to navigate through the data of the building. The module "*cat_config*" holds the keywords and useful mapping structures to ease the manipulation of such building meta-data and measurement values. Then, the on/off control logic practically run in the method *update*(), senses the constrained zone temperature to detect a possible comfort violation. Whenever such a case arises, the corresponding thermal load is switched on or off. In this implementation, it is worth noting that the hot water tank is modeled as a "zone" of water influenced by its EWH. Then, the EV object is retrieved and a charging instruction is emitted if it is connected and not fully charged. The deferrable loads are triggered as early as possible. Finally, the set of generated commands are collected in the list, *command_set* and returned by the function.

4.4.2 MPC formulation tailored to a Minergie building

Running MPC as the energy manager is performed by instantiating the *EMS_MPC* class that inherits from the *EnergyManagementSystem* class. Unlike the simple rule-based traditional control presented above, the MPC requires a model of the building to decide on the optimal sequence. To this end, the class uses both the *Linear Programming* and *Load Scheduler* modules. On the one hand, the *Linear Programming* module provides the *EMS_MPC* class with an interface to model the energy-related entities present in the building. On the other hand, the *Load Scheduler* module is in charge of appropriately scheduling the deferrable loads in a day-ahead manner every day and effectively triggering them at the corresponding time the next day.

Deferrable load scheduling

Scheduling the deferrable loads intends to minimize the cost of triggering them over the next 24 hours. The cost minimization problem is subject to constraints on the start/end time preferences specified by the user and models their predefined load profile. The problem can therefore be written as [147]:

$$\min_{\boldsymbol{t_d}} \sum_{h=0}^{D-1} \sum_{i=1}^{N_d} c[h]\, e_{d,i}[h] \tag{4.11}$$

$$\text{s.t.} \quad \underline{\forall i = 1 \dots N_d}:$$

$$e_{d,i}[h] = \begin{cases} \mathscr{P}_i[h - t_{d,i}] & \text{if } t_{d,i} \le h \le t_{d,i} + |\mathscr{P}_i| \\ 0 & \text{otherwise} \end{cases} \qquad \forall i = 1 \dots N_d \tag{1.22}$$

$$\underline{t}_{d,i} \le t_{d,i} \le \overline{t}_{d,i} - |\mathscr{P}_i| \qquad \forall i = 1 \dots N_d \tag{1.23}$$

where D is the number of time slots in the next 24 hours, $c[h]$ is the price of electricity ($\$/kWh$) at time instant h, $\mathscr{P}_i$ is the power profile of the i^{th} load, and N_d is the number of deferrable loads to be scheduled. The power consumption of load i at time h is expressed as $e_{d,i}[h]$ and is defined by Eq. (1.22), under the constraints Eq. (1.23) as described in Chapter 1. The optimization vector $\boldsymbol{t_d} = [t_{d,0} \dots t_{d,N_d-1}]$ contains optimal decisions on the starting instants of the deferrable loads.

Practically, the above load scheduling problem can hardly be solved due to the $if...else$ statement in Eq. (1.23) that involves the optimization variables $\boldsymbol{t_d}$. Using additional vectors circumvents this issue, leading to the following problem:

$$\min_{\boldsymbol{x_d}} \sum_{h=0}^{D-1} \sum_{i=1}^{N_d} f_i[h]\, x_{d,i}[h] \tag{4.12}$$

$$\text{s.t.} \quad \underline{\forall i = 1 \dots N_d}:$$

$$\sum_{h=0}^{D-1} x_{d,i}[h] = 1 \tag{4.13}$$

$$x_{d,i}[h] = 0 \qquad \forall h \in [0; \underline{t}_{d,a}] \cup [\overline{t}_{d,a} - |\mathscr{P}_a|; D] \tag{4.14}$$

where:

- The optimization vector $\boldsymbol{x_d}$, contains binary optimization vectors $\boldsymbol{x_{d,i}}$, whose boolean values $x_{d,i}[h]$ indicate whether the load i should be trigger at time instant h.

- $f_i[j] = \sum_{h=j}^{j+|\mathscr{P}_i|} c[h] e_{d,i}[h]$ is the cost to trigger load i at time j,

- Constraints defined by Eq . (4.13) enforce vectors $\boldsymbol{x_{d,i}}$ to only have one value different than zero. Consequently, only one scheduling instant may exist.
- Constraints defined by Eq . (4.13) ensure that the scheduling instant cannot fall outside user preferences.

Solving this optimization problem outputs N_d vectors $\boldsymbol{x_{d,i}}$ whose index corresponds to the unique non-null value that determines the scheduled time $t_{d,i}$ for load i. It's worth noticing that this formulation can be adapted for inter-dependences among the deferrable loads (e.g., the dryer must run after the washing mashing).

MPC formulation

The $update()$ method solves the following MPC problem at every update period dt_{mpc}, to generate setpoint $\boldsymbol{u}$:

$$\min_{\boldsymbol{u}} \sum_{h=0}^{H-1} c[h]\ e^{+}[h] + k_{\epsilon} \sum_{h=0}^{H-1} \boldsymbol{\epsilon}^{T} \boldsymbol{\epsilon} \tag{4.15}$$

$$\text{s.t.}\ x_b[0] = x_b^0,\ \ \boldsymbol{x_z^0}[0] = x_z^0 \tag{4.16}$$

$$x_b[h_a] = C_a,\ \ x_b[h_l] = C_e \tag{4.17}$$

$$\underline{\forall h = 0 \dots H-1}:$$

$$e[h] = u_{hp}[h] + u_{ewh}[h] + u_{b,c}[h] + u_{b,d}[h] + p_g[h] + p_{nc}[h] + \sum_{i=1}^{N_d} p_{d,i}[h] \tag{4.18}$$

$$e^{+}[h] + e^{-}[h] = e[h],\ \ e^{+}[h] \geq 0,\ \ e^{-}[h] \leq 0 \tag{4.19}$$

$$0 \leq e^{+}[h] \leq M\ s_e[h] \tag{4.20}$$

$$0 \geq e^{-}[h] \geq -M\ (1 - s_e[h]) \tag{4.21}$$

$$\boldsymbol{x_z}[h+1] = A\ \boldsymbol{x_z}[h] + B_u\ u_{hp}[h] + B_d\ \boldsymbol{d_z} \tag{4.22}$$

$$\boldsymbol{T_z}[h] = C\ \boldsymbol{x_z}[h] \tag{4.23}$$

$$0 \leq u_{hp}[h] \leq \overline{P}_{hp} \tag{4.24}$$

$$\underline{\boldsymbol{T}}_{\boldsymbol{z}}[h] - \boldsymbol{\epsilon_z}[h] \leq \boldsymbol{T_z}[h] \leq \overline{\boldsymbol{T}}_{\boldsymbol{z}}[h] + \boldsymbol{\epsilon_z}[h] \tag{4.25}$$

$$T_w[h+1] = \alpha_t T_w[h] + (1-\alpha_t)(\frac{AU}{R^*} T_o[h] + c_w \dot{m} R^* T_{in} + R^*\ u_{ewh}[h]) \tag{4.26}$$

$$\underline{T}_w - \epsilon_w[h] \leq T_w[h] \leq \overline{T}_w + \epsilon_w[h] \tag{4.27}$$

$$x_b[h+1] = \alpha_b x_b[h] dt + \eta_{b,c} u_{b,c}[h] dt + \frac{1}{\eta_{b,d}} u_{b,d}[h] dt \tag{4.28}$$

$$0 \leq u_{b,c}[h] \leq \overline{P}_{b,c}\ s_b[h] \tag{4.29}$$

$$0 \geq u_{b,c}[h] \geq -\overline{P}_{b,d}\ (1 - s_b[h]) \tag{4.30}$$

$$\underline{C} \leq x_b[h] \leq \overline{C} \tag{4.31}$$

The vector $\boldsymbol{u} = [u_{hp}^*, u_{ewh}^*, u_b^*]$ contains the power consumption setpoint of controllable entities: u^*_{hp} is the HP control power, u^*_{ewh} is the EWH control power, and $u^*_b = u_{b,c} + u_{b,d}$ is the battery control power.

Eq. (4.18) defines the power exchange with the grid e as the sum of the optimization variables $\boldsymbol{u}$, the locally generated power forecast p_g, the uncontrollable load power consumption p_{nc}, and load consumption due to deferrable loads $p_{d,i}$. The PV production is determined through Eq. (1.40) using the forecast of outside air temperature and sun irradiance provided by the *Ambient Data* module. Power consumption drawn by the uncontrollable and deferrable loads depend on their load profile $\mathscr{P}$. Whereas the deferrable loads have a deterministic schedule, the uncontrollable power forecast is set to the most likely profile in $\mathscr{P}$.
Furthermore, Eq. (4.19) to (4.21) separate the net positive import e^+ and the net positive export e^-, such that only the power consumption of the building is used in the objective function defined by Eq. (4.15). The variable s_e is used to implement the Big-M method [160] that ensures that e^+ and e^- are never simultaneously non-null.

Eq. (4.22) to (4.23) describe the thermal dynamics of the building zones and the hydronic system that drive the HP power consumption, thoroughly detailed in Section 2.3. Constraints (4.24) to (4.25) limit the HP power consumption and the zones temperature, respectively. Soft constraints are used in Eq. (4.25) to ensure problem feasibility through slack variables $\boldsymbol{\epsilon_z}$. These slack variables $\boldsymbol{\epsilon_z}$ contained in $\boldsymbol{\epsilon}$ are penalized in the objective function Eq. (4.15) by forcing their square values to zero through a high parameter k_ϵ ($\sim 10^6$).

Hot water tank temperature evolution is modeled in Eq. (4.22), where $\alpha_t = e^{-\frac{dt}{R^*C}}$, as defined in Section 1.1.1. In Eq. (4.23), the hot water temperature is forced to be bound between $\underline{T}_w$ and $\overline{T}_w$ with soft constraint slack variables ϵ_w. Similar to zone modeling, ϵ_w is inserted in $\boldsymbol{\epsilon}$ to be penalized in the objective function Eq. (4.15). The battery SoC evolution of the EV is represented by Eq. (4.28). Constraints on maximum charging and discharging power are applied in Eq. (4.29) and Eq. (4.30). The Big-M method is used again to prevent charging and discharging from being set at the same time. Eq. (4.30) bounds the battery SoC to acceptable maximum and minimum values.

Initial states of the battery, zone thermal state, and hot water temperature are described in Eq. (4.16). In addition, Eq. (4.17) models the initial SoC C_a at arrival instant and the final SoC C_e at leaving instant. Due to the latter constraints on the battery state at the leaving instant, a minimum value of the MPC horizon H must be set to ensure that the optimization problem has the necessary visibility on future states and constraints [2]. After solving the MPC problem, the $update()$ method practically extracts the first values of the optimal vector $\boldsymbol{u}$ and returns these values for a further transmission to the building management system. At the next control time step $t + dt_{mpc}$, the building state will be updated by OpenEMS and the MPC will use the new values to compute the next setpoints.

Both day-ahead load scheduling and the MPC problem are MILP optimization problems, well

---

[2] The MPC horizon H could be reduced by applying a linear change on the final SoC of the battery at the leaving instant, but this may lose the knowledge of future low prices of electricity.

studied and compatible with most solvers. Indeed, the scheduling task defined through Eq. (4.12) to Eq. (4.14) relies on a linear objective with linear constraints and the nature of the optimization variables must be boolean (0 or 1). Concerning the MPC, the models inserted as problem constraints are all linear to their optimization variables and binary variables are used in the Big-M formulation. The cost function holds a linear term to minimize the cost paid by the user and a quadratic term to penalize the comfort violations, resulting in a convex problem.

4.4.3 Simulation results

Daily simulations have been carried out, to observe the building behavior under both energy management strategies given the same environmental conditions. This section intends to highlight the use of OpenEMS to practically deploy these strategies and discuss their advantages/drawbacks, rather than quantifying precisely their effect on building consumption metrics. Indeed, the presented results are highly dependent on the chosen environment, user behavior, and the simulation models.
An Intel Core i7-4710HQ CPU (2.50GHz × 8) was used to practically run the *vEngine*/OpenEMS processes and the Gurobi solver [161] solves the MPC optimization problem every 15 minutes. The solver has been used for its ability to handle MILP problems and the handy interface with Python.

While the model parameters and environmental data are the same as the ones used in Section 2.3, the following parameters have been chosen for the simulations:

- Traditional energy management: both simulation timesteps in the *vEngine* and the update of setpoints in OpenEMS are fixed to 60 seconds to react swiftly to comfort violation.
- MPC energy management: the simulation timestep of the *vEngine* is fixed to 60 seconds while OpenEMS only updates the setpoints every 900 seconds (15 minutes). The price of electricity is derived from the Swiss Day-Ahead Auction Market prices [162]. A horizon of 18 hours is chosen for the MPC, leading to 72 time slots of forecasting. This value has been decided based on the constraints on the EV but also to appropriately catch the variation of the price of electricity and the large inertia of the hydronic system.

Fig. 4.6 depicts the power exchange with the grid for both methods, and the EV battery SoC over time. The price of energy ($/kWh) presents typical peaks in the morning around 7.00 am, towards the end of the day around 5.00-6.00 pm, and a minimum during the middle of the day. Traditional control (grey curve) is not aware of this and triggers a peak in power consumption at moments of high cost. The charge of the electrical car is also inappropriately launched from 4.00 am and crosses high a price period. Short peaks in power consumption are due to the EWH that strives to maintain a constant hot water temperature. During the

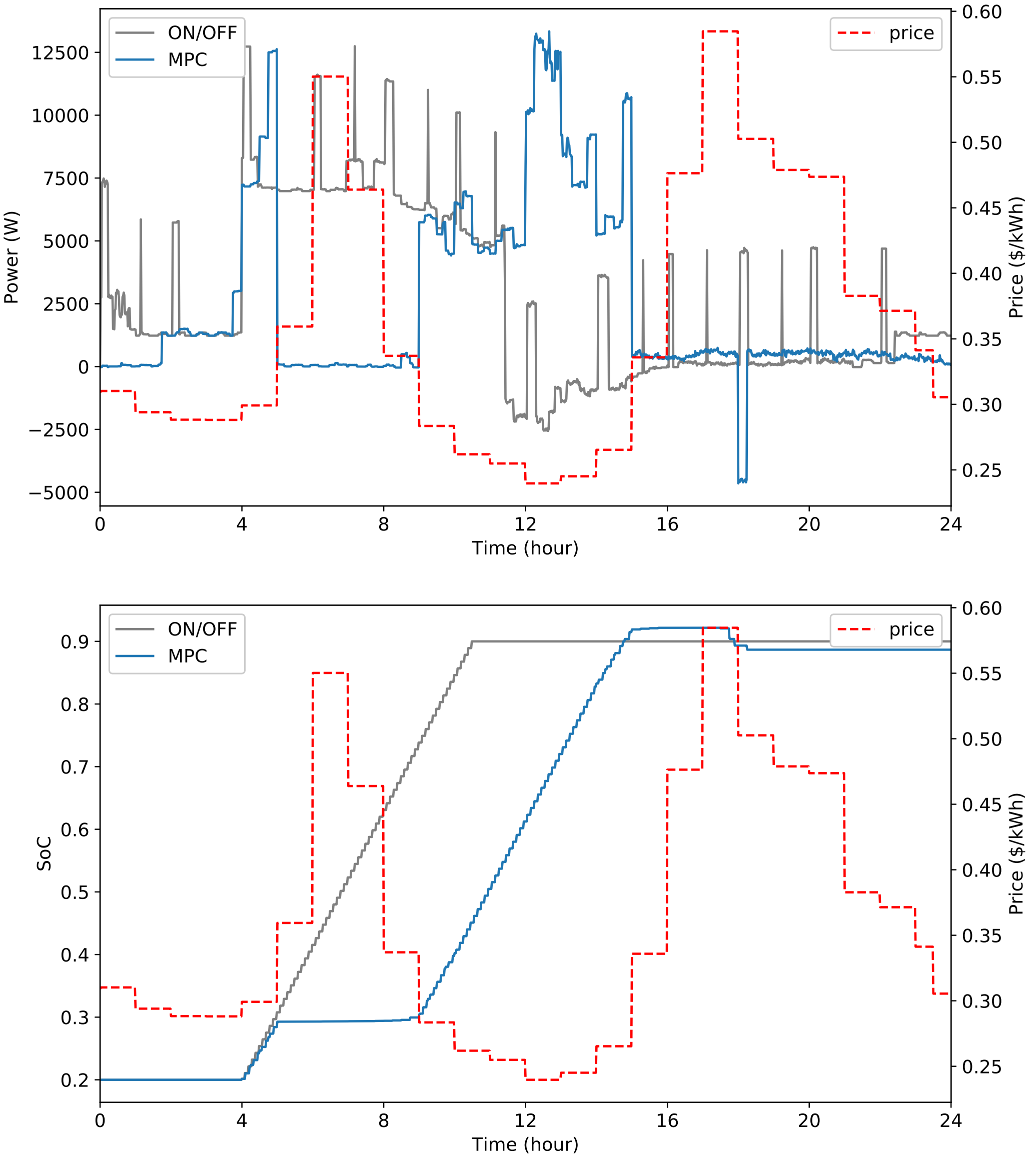


Figure 4.6 – OpenEMS applied on a Minergie building: comparison between traditional control and MPC (top) grid consumption (bottom) battery state of charge

day, local PV power is produced and cannot entirely be used by the loads under traditional control. Consequently, part of this energy has been injected into the grid. The MPC strategy (blue curve) could optimally handle the high peaks of energy by pre-heating the hydronic system and the hot water tank while holding onto the charge of the car in the morning peak. A high peak of grid power demand is witnessed during instants of low prices (around noon), due to the decision of the MPC to switch on to their maximal capacity every flexibility load, in addition to some of the deferrable ones already scheduled the day before.

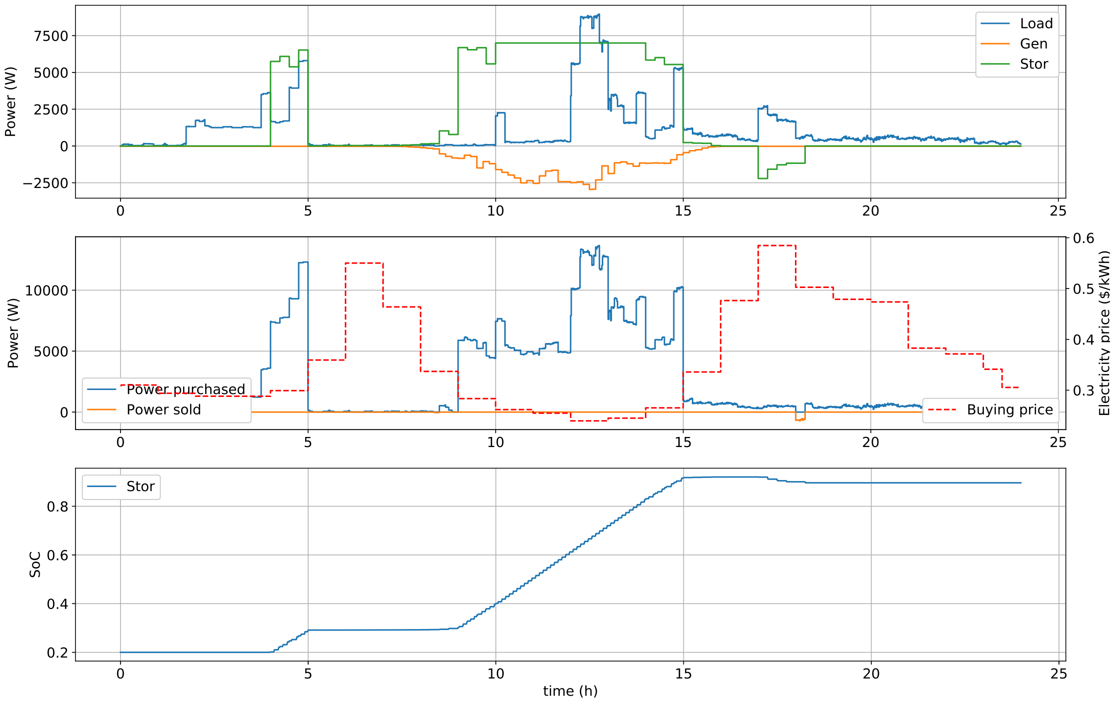


Figure 4.7 – OpenEMS applied on a Minergie building: focus on MPC

Interestingly, the MPC decided to charge the EV battery beyond its necessary SoC at leaving time (6.00 pm). This is to cope with a high price of energy during the second peak coupled with the absence of sunshine, hence using part of the energy stored in the EV to power loads. However, an odd behavior drives the battery in discharge mode to release energy back to the grid for 15 minutes. Fig. 4.7 presents the details of MPC method results, highlighting the behavior of each type of building entity. As expected, the flexibility load consumption is entirely covered by the battery of the EV around 5.00 pm, but the battery is wrongly programmed in discharge mode and therefore, releases energy to the grid, going against optimal behavior since the grid utility does not pay the user for the energy generated. This mistake is explained by an unexpected problem with the Gurobi solver, which failed at solving the MPC at 6.00 pm. Whereas the loads are switched off, the battery kept the last setpoint in memory and continued applying it.

As for the comfort of the user, Fig. 4.8 compares the temperatures resulting from both methods. Under traditional energy management, the hot water tank keeps a constant temperature, as the EWH is immediately switched on as soon as the temperature deviates from the mean level of acceptable comfort. The hydronic system and the zones present a slower dynamic and are influenced by external conditions such as solar irradiance and the outside air temperature. After an intense pre-heating in the morning to reach an acceptable mean zone temperature, the inherent large inertia of the water system keeps heating the zones, and the smaller zone even sees its top-constraint being slightly violated.

The MPC offers a more powerful management of the flexible entities, taking into account the daily local production and prices of energy through the linear model-based forecast. It realizes smarter temperature management for the Minergie building for both the hot water tank and the zones. Unlike the traditional control, it does not pre-heat the hydronic system nor the hot water tank and even guides their temperatures towards their lower bound for most of the morning. An exception occurs in the EWH that must counteract the frequent water draws triggered by the user. Nevertheless, it manages to do it at low energy prices (before 5qm and around 10am), like the charge of the EV that does not happen during the high morning peak. The knowledge of the building model, future environmental conditions, and prices of electricity allowed the system to consume as low as possible from the grid while keeping the lowest bound still acceptable. Then, as the sun shines and the price of electricity decreases, both the thermal loads and the EV are set to their maximum power. Relatively high prices in the evening and the corresponding absence of sun led the MPC to pre-heat the zones and hot water tank as much as possible. By doing so, good insulation of the building could avoid the need to provide heat for most of the time when the sun does not shine.

4.5 Conclusion

Energy management in buildings represents an active research topic, due to the forthcoming electrification of building loads, emerging RES, and intrinsic flexibility of most of the big thermal loads and batteries. Designing a proper energy management strategy is, therefore, of utmost importance for advanced Building-to-Grid (BtG) integration. In this chapter, we presented an open-source energy management system, dubbed OpenEMS. The main interest of the tool resides in its plug-and-play nature concerning existing BDMS. It leverages the metadata of the latter and periodically updates its local representation of the building. A set of objects mapped to the building environment (state, preferences, outside environment, etc.) is exposed to the user to easily program any kind of energy management logic, remaining agnostic to the underlying technology of sensors and actuators. Future work consists of rendering OpenEMS compatible with popular metadata schema for buildings like BRICK [163], towards a standardized BDMS-EMS communication.

A test case illustrated the use of this object-oriented library to manage the energy of a simulated Minergie building, initially introduced in Chapter 2. It aimed to highlight how the OpenEMS could be used to compute setpoints based on simple logic (traditional thermostats) to more advanced model-based management strategies (MPC). The compactness of code fosters the quick development of a proof-of-concept control strategy, and the advanced *Linear Programming* and *Load Scheduling* modules can be leveraged for smarter management strategies.

MPC has shown great results on the energy consumption of the Minergie-inspired house and is considered as the most promising building control technique. Nevertheless, its performance highly depends on the developed model, the quality of the forecast, and the identified parameters. In addition, it requires intense computational power and ill-conditioned problems can fail to be solved. The developed open-source tool, therefore, reduces the engineering time to deploy an EMS and allows researchers to focus on the enhancement of advanced control algorithms like MPC.

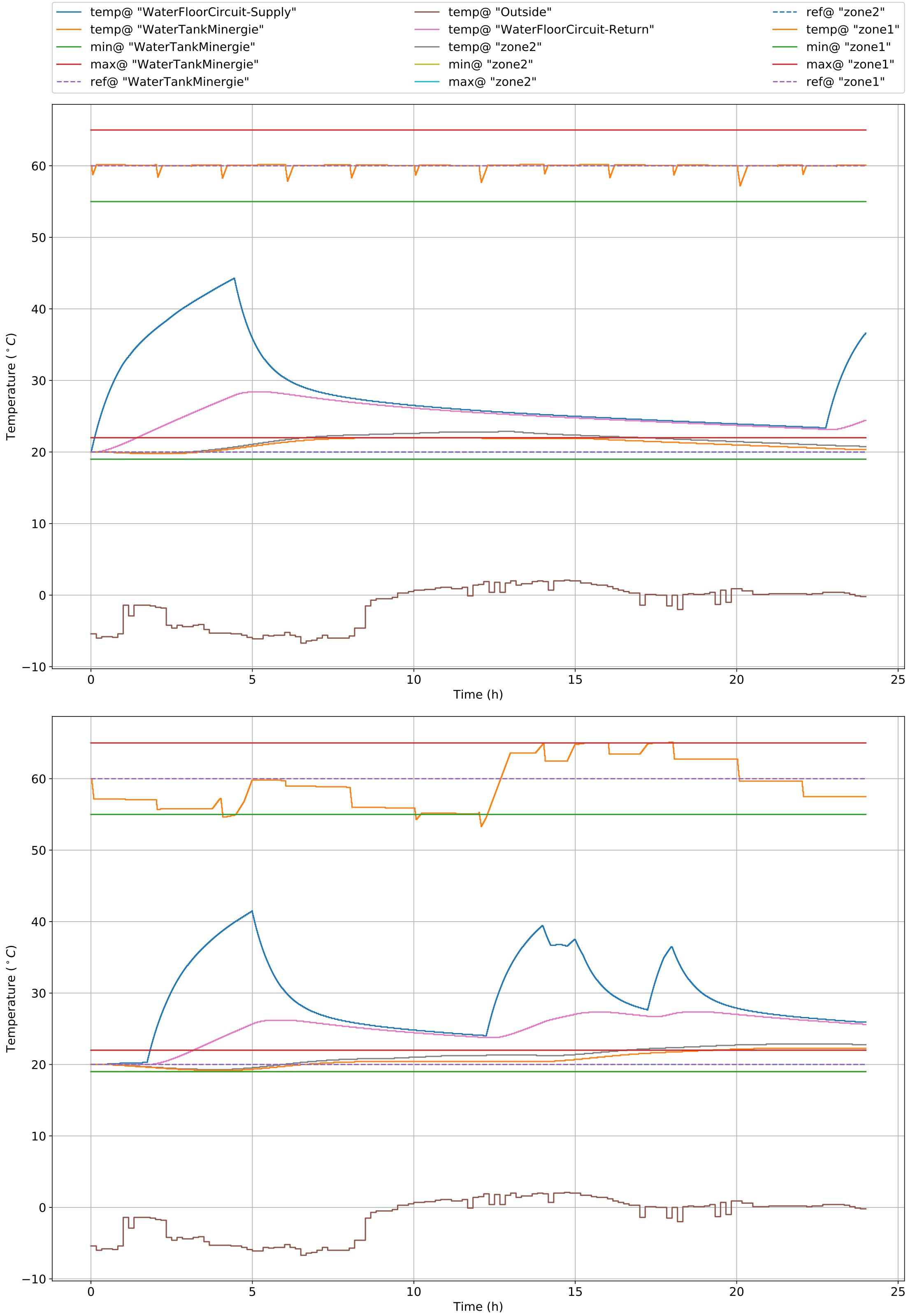


Figure 4.8 – OpenEMS applied on a Minergie building: temperature comfort (top) traditional control (bottom) MPC

5 MPC Applied to Commercial Buildings: Balancing Energy and Peak Demand

---

Model Predictive Control (MPC) has intensively been studied for managing the energy of commercial buildings. Their large controllable thermal loads and the increasing presence of automation systems make them the ideal candidate for providing grid flexibility. Due to their relatively high demand on the electrical grid, utilities charge them for their maximum peak consumption every month to foster their minimization. However, using the monthly bill definition, involving demand charges, in the relatively short MPC receding horizon does not optimize the owner's bill nor the grid services. Therefore, there is a need to thoroughly study the balance between energy and peak demand for MPC in commercial buildings.

The main **highlights** and **contributions** of this chapter are:

- Introduction of an innovative MPC method, called *incremental* MPC, to better tackle demand charges over the entire month and across multiple TOU periods.
- Simulation on realistic U.S. commercial buildings to compare multiple MPC formulations and their impact on owner's bills, grid peak demand, and load shifting flexibility. Substantial gains in grid flexibility can be obtained through our incremental approach, while maintaining, or even reducing, the owner's bill and grid peaks.

Related **publications**:

[164] O. Van Cutsem, D. Blum, M. Pritoni, and M. Kayal, "Comparison of MPC Formulations for Building Control under Commercial Time-of-Use Tariffs," *PowerTech 2019.*

5.1 Commercial Time-Of-Use Rates and MPC

The building sector in the U.S. accounts for 40% of the country's energy consumption [165], and commercial buildings are responsible for 36% of all U.S. electricity usage [166]. The latter is gaining increasing attention, due to its high potential for energy saving and load shifting. Constituting a large portion of the energy consumption of commercial building, HVAC systems represent a primary target of control method research. Most large commercial buildings and offices generally have a BDMS which collects sensor data and may act on HVAC/lights in the building. Therefore, they represent ideal grid-responsive candidates to support the grid through advanced control.

MPC recently emerged as a state-of-the-art method in commercial building energy management [135]. The ability of MPC to consider future conditions to drive the current system state makes it suitable for DR. DR refers to the set of grid mechanisms to shape a building's electric load when market prices are high or when the grid reliability is jeopardized [167], through either financial incentives or electricity pricing structures. TOU tariffs, which define distinct price levels for specific periods of the day as a form of DR, are widely used to incentivize consumers to shift demand outside of grid peak hours. In addition, CPP programs superimpose a large increment in energy price to the basic TOU rates for some strategic hours, generally decided a day in advance by the grid utility. This known structure can be directly leveraged by the MPC formulation to shift loads appropriately.

TOU tariffs are a widespread means to shape daily commercial building consumption. Unlike RTP, TOU rates are well defined in advance, over the period of the contract settled between the utility and the end-user [168]. Under TOU rates, prices of energy ($/kwh) are generally higher during peak load periods, and lower when there is an excess of generation with respect to the load. In addition to TOU energy costs, commercial building owners also face TOU *peak power demand* charges ($/kW) every month. These peak power demand charges (referred to as demand charges in the rest of this chapter) penalize the maximum power consumption values [1] for each of the TOU periods defined by the utility.

Definition 5.1.1. Demand cost - A demand cost refers to the part of the electric bill that charges the building's owner for the peak power consumption set during the month.

The monthly bill B_m ($) of a commercial building in the U.S. can, therefore, be derived as follows:

$$\mathrm{B}_m = C_{fix} + \sum_{h=1}^{N_m} c_e[h] \cdot E[h] + \sum_{\rho=1}^{N_{TOU}} c_d[\rho] \cdot \max_{j \in \mathcal{H}_\rho} \{P[j]\} \tag{5.1}$$

where:

- C_{fix} ($) is a fixed monthly cost

---

[1] The power consumption is generally computed over 15 minutes, or 5 min for large consumers.

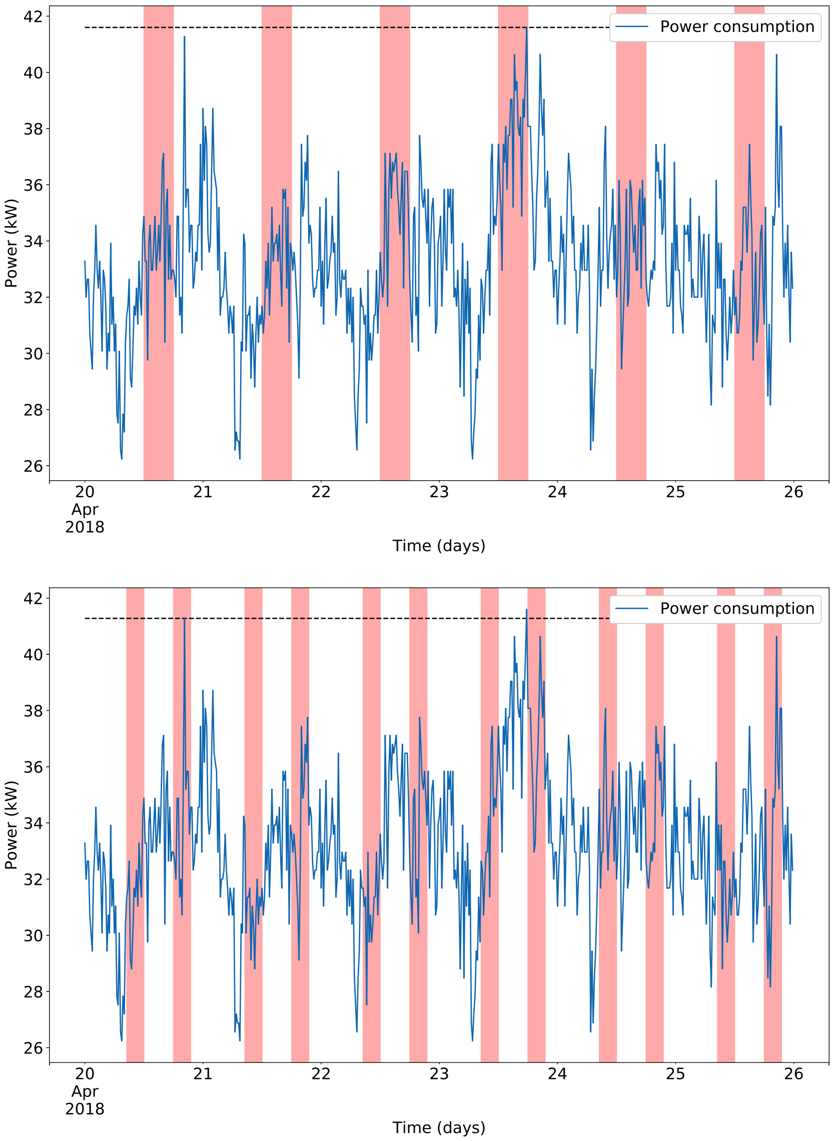


Figure 5.1 – Monthly Time-Of-Use Demand Charges: example of on- and mid- peaks on a 6 days power consumption signal

- $c_e[h]$ is the cost of energy ($/kWh) at time h
- $c_d[\rho]$ is the cost of demand ($/kW) for the TOU period ρ
- $E[h]$ (kWh) is the energy consumed by the building in period h
- $P[j]$ is the power demand (kW) of the building in period j
- N_m is the total number of time periods in the month, N_{TOU} is the number of TOU demand charge periods
- $\mathscr{H}_\rho$ represents the periods in the month when the TOU demand charge ρ is active

The first term C_{fix} incorporates infrastructure charges. The second term represents the integrative energy consumption charges. The third and last term is the demand cost. Whereas energy costs are incrementally summed up over time, demand charges account for a handful of power peaks set over a large period of time. Fig. 5.1 illustrates how utilities practically compute these demand charges, on a monthly power signal reduced to a 6-day example under a 2018 Pacific Gas & Electricity (PG&E) tariff structure. For each of the TOU periods - highlighted in red on the graphs - the maximum 15-min-averaged power consumption is found and multiplied by the corresponding demand cost for that period. In addition to the on-peak and mid-peak charges depicted in Fig. 5.1, a global demand charge over the entire month is added to the monthly bill.

Most of the MPC works encountered in the literature focus on optimizing the energy-related components of Eq. (5.1) [135]. When peak demand charges are tackled, the approach is generally a "best-effort" or "naive" manner, for it strives to reduce the peak as much as possible at every control step. In [169, 170, 171], MPC formulations include a maximum demand penalty alongside the incremental energy consumption. Kim et al. [169] presented an overall MPC approach to coordinate RTUs and the experimental field test showed both energy and demand costs could be reduced compared to traditional controllers. In [170, 171], week-long simulations on EnergyPlus models, coupled with a MATLAB controller, demonstrated that the simulated building could follow temperature setpoints, while precooling and reduce peaks. The authors in [172] presented a stochastic optimization method that keeps the grid power purchased below a defined demand charge threshold, considering multi-peak periods. For these MPC formulations, the coefficient of the demand component in the objective function is either set to a single monthly cost of demand or a weight for which the tuning is not discussed at all.

Nevertheless, these methods fail to look at the problem on a monthly basis. The MPC horizon only optimizes the power profile in the receding horizon, and does not guarantee the minimizing of a monthly bill. Even though it ensures peak reduction in a best-effort manner, this does not necessarily translate into bill optimization or better grid services. Therefore, in this chapter, we investigate how to change the MPC approach to ensure the overall bill optimization, and compare our approach with existing ones.

The rest of this chapter is organized as follows. Section 5.2 presents the generic commercial building model used throughout the chapter and describes the common MPC formulations encountered in the literature to tackle demand charge. It further introduces the innovative incremental approach used to better balance energy and demand costs on a monthly basis. In Section 5.3, two case studies are used to compare the different MPC formulations for handling demand charges and their impact on both the building owner and grid manager; simulations are performed to theoretically discuss the metrics. Section 5.4 concludes this chapter.

Table 5.1 – Nomenclature used in Chapter 5: parameters

Symbol	Description	Unit
C	The matrix linking building state $\boldsymbol{x}$ to building temperature t	/
$c_e[h]$	The price of energy at time instant h	\$/kWh
$c_d[\rho]$	The demand charge for TOU period ρ	\$/kW
$\boldsymbol{d}(t)$	The disturbance affecting the building model t	/
f	The building model function t	/
H	The number of steps in the MPC horizon	/
h	The time period index	/
$\mathscr{H}_\rho$	The time periods in the month when TOU period ρ occurs	/
N_{TOU}	The amount of TOU periods	/
k_d	The coefficient for demand charge in the "naive" objective function	/
$p_{nc}[t]$	The uncontrollable building power demand at time t	kW
ρ	The index of a TOU period t	/
$\underline{T}, \overline{T}$	The minimum/maximum comfort on temperature	$^\circ C$
u_{max}	The maximum HVAC power	kW
x_ρ^{thr}	The power demand threshold in TOU period ρ	kW

Table 5.2 – Nomenclature used in Chapter 5: variables and notations

Symbol	Description	Unit
$e[t]$	The building net power demand at time t	kW
$T[t]$	The building temperature at time t	$^\circ C$
$u[t]$	The building flexible HVAC demand at time t	kW
$\boldsymbol{x}[t]$	The building thermal state at time t	/
z	The slack variable used to model maximum "naive" power demand	kW
z_ρ	The slack variable used to model maximum power in TOU period ρ	kW
$\bullet$(bold)	Vector $[\bullet_0, ..., \bullet_n]$	/

5.2 Economic MPC to Tackle Demand Charges

Let us first introduce the generic continuous building model that is used in the MPC formulations of this chapter. We consider an instantaneous commercial building power consumption that can be expressed as follows:

$$e(t) = p_{nc}(t) + u(t) \tag{5.2}$$

where $p_{nc}(t)$ represents the uncontrollable building consumption and $u(t)$ is the flexible part of the building consumption that can be autonomously controlled by the EMS. The non-flexible part of the power consumption contains any kind of commercial loads, lighting, and appliances that cannot be actuated by the EMS. It is assumed that the commercial EMS can only act on the HVAC system, in which the electrical power influences the room temperature $T(t)$ according to:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), u(t), \boldsymbol{d}(t)) \tag{5.3}$$

$$T(t) = C\,\boldsymbol{x}(t) \tag{5.4}$$

where $\boldsymbol{x}(t)$ stands for the building thermal state, $u(t)$ the control set-points of the HVAC, $\boldsymbol{x}(t)$ the disturbances influencing the building state, and C is a vector mapping the building state to the building temperature $T(t)$.

The HVAC power consumption is physically limited:

$$0 \le u(t) \le u_{max} \tag{5.5}$$

Finally, the commercial building ambient temperature must be kept within the comfort boundary at any time instant, with a possible night setback:

$$\underline{T}(t) \le T(t) \le \overline{T}(t) \tag{5.6}$$

where $\underline{T}(t)/\overline{T}(t)$ represent the time-varying temperature comfort limits.

5.2.1 Common "naive" approach to Demand Charges

The common approach encountered in the literature [135, 172, 173, 169, 170, 171] to handle demand charges in MPC can be expressed as follows:

$$\min_{u} \sum_{h=0}^{H-1} c_e[h] \cdot e[h] \cdot dt + k_d \cdot \max_{h=0\ldots H-1}(e[h]) \tag{5.7}$$

$$\begin{aligned}
\text{s.t. } & \underline{\forall h = 0 \ldots H-1}: \\
& e[h] = p_{nc}[h] + u[h] && (5.2)\\
& x[h+1] = f_d(x[h], u[h], d[h]) && (5.3)\\
& T[h] = C_d\, x[h] && (5.4)\\
& 0 \le u[h] \le u_{max} && (5.5)\\
& \underline{T}[h] \le T[h] \le \overline{T}[h] && (5.6)
\end{aligned}$$

where $\max_{h=0\ldots H-1}(e[h])$ denotes the maximum power peak in the receding horizon. The *max* term in Eq. (5.7) can be replaced using the slack variable z:

$$\underset{u}{\text{minimize}} \sum_{h=0}^{H-1} c_e[h] \cdot e[h] \cdot dt + k_d \cdot z \tag{5.8}$$

$$\begin{aligned}
\text{s.t. } & \underline{\forall h = 0 \ldots H-1}: \\
& \text{Building model Eq. (5.2) to (5.4)} \\
& \text{Building constraints Eq. (5.5) to (5.6)} \\
& z \ge e[h] && (5.9)
\end{aligned}$$

The Eq. (5.9), along with the minimization of the z term, ensures that the variable z will take the highest value of the building power consumption in the receding horizon H.

This approach strives to minimize the power peak demand in each receding horizon H. However, this does not guarantee that the *monthly* bill is reduced at the end of the billing period, nor that the grid operator can expect a given building flexibility. This MPC problem does not consider that the utility demand bill is based on the highest peak throughout the entire month, during each TOU period. Given a reasonable optimization horizon on the order of 12–24 hours, rather than the whole month, it is, therefore, incorrect to weigh the monthly demand costs against only a day's worth of energy cost in the objective. This will bias the optimization to reduce demand as much as possible during each MPC control period, at the expense of a higher energy cost, especially if a high peak demand has already been set earlier in the month.

In light of the aforementioned features, the common approach encountered in the literature is referred to as a "naive" approach. Indeed, the corresponding economic MPC formulations applied to commercial buildings naively include the bill formulation Eq. (5.1) in their objective function.

5.2.2 Incremental approach to demand charges

To better account for the demand charges in the MPC horizon, they should be considered in an *incremental* way, like the energy charges:

$$\underset{x}{\text{minimize}} \sum_{h=0}^{H-1} c_e[h] \cdot e[h] \cdot dt + \sum_{\rho=1}^{N_{TOU}} c_d[\rho] \cdot z_\rho \tag{5.10}$$

$$\text{s.t. } \underline{\forall h = 0 \dots H-1}:$$

Building model Eq. (5.2) to (5.4)

Building constraints Eq. (5.5) to (5.6)

$$\underline{\forall \rho = 0, \dots, N_{TOU}}:$$

$$z_\rho \geq 0 \tag{5.11}$$

$$z_\rho \geq e[h] - x_\rho^{thr}, \quad \forall h \in \mathscr{H}_\rho \cap \{0, \dots, H-1\} \tag{5.12}$$

In this MPC formulation, z_ρ is a slack variable representing the maximum increment in demand for the corresponding TOU periods ρ of the horizon H, and x_ρ^{thr} is a peak demand threshold that only penalizes demand cost if the power $e[h]$ is larger than this threshold. It could be set to the maximum peak encountered since the beginning of the billing period, or to a prediction of what the maximum demand during the month will be. Therefore, $x_\rho^{thr} = max(x_\rho^{seen}, x_\rho^{exp})$ is the maximum of x_ρ^{seen}, the already seen maximum demand in $\mathscr{H}_\rho$, and x_ρ^{exp}, the expected maximum demand in the billing period in $\mathscr{H}_\rho$. We term this formulation as *incremental* because the objective function represents the incremental portion of the monthly bill for the given time horizon for both the energy and demand costs.

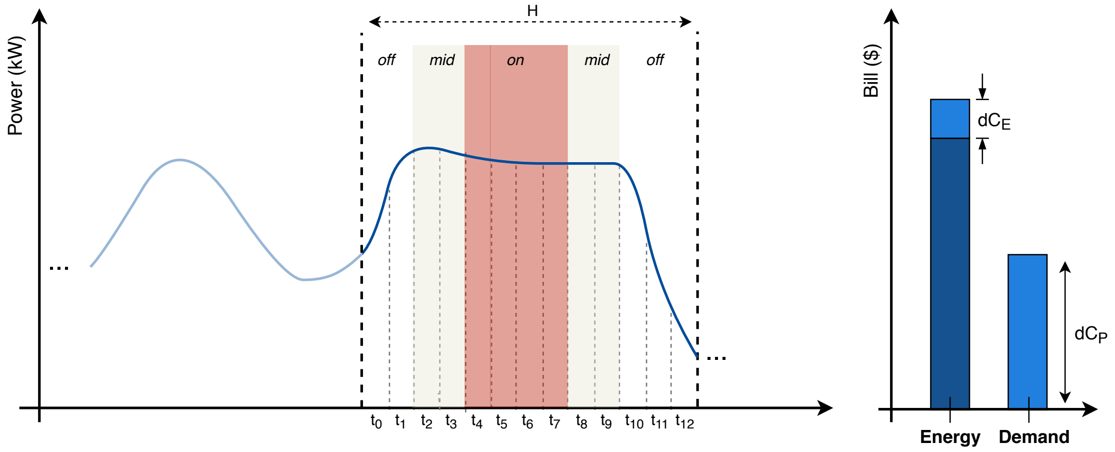

Power (kW)
H
off
mid
on
mid
off
...
...
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12}
Bill ($)
dC_E
dC_P
Energy
Demand

Figure 5.2 – Illustration of the "Naive" MPC approach to tackle demand charges

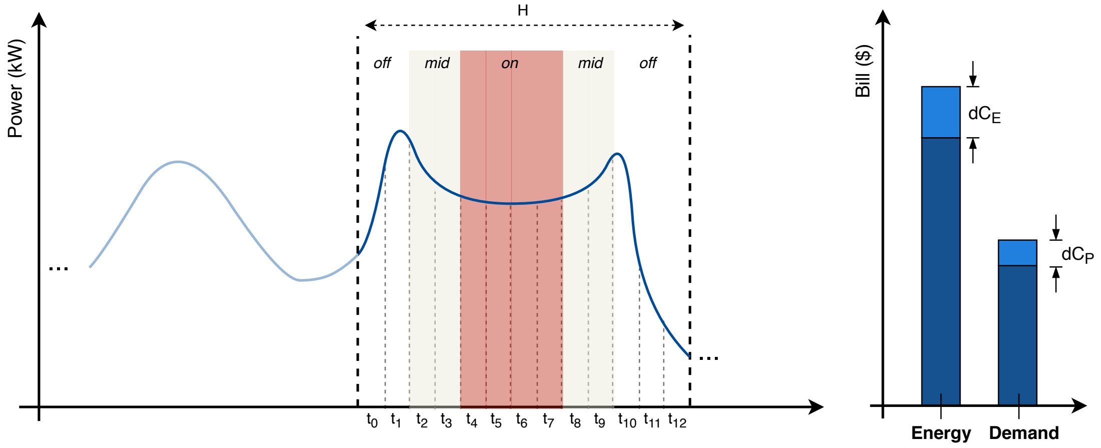

Power (kW)
H
off
mid
on
mid
off
...
...
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12}
Bill ($)
dC_E
dC_P
Energy
Demand

Figure 5.3 – Illustration of the "Incremental" MPC approach to tackle demand charges

As illustrated in Fig. 5.2, the "naive" approach formulated by Eq. (5.8) penalizes the absolute peak power consumption over the receding horizon, along with the incremental energy cost. This method does not leverage the knowledge of past peaks that have already been set earlier in the billing period - or any forecast of what it could be. Fig. 5.3 depicts how the *incremental* approach handles the additional demand charges. This method now penalizes the increment of energy cost and the increment of demand cost in the monthly bill, over each receding horizon.

The building owner can, therefore, optimize his/her monthly bill by applying Algorithm 3. At the beginning of each billing period, the function $init\_peak()$ initializes the threshold values x_{ρ}^{thr} in Eq. (5.11) & (5.12), for each TOU period ρ. The threshold could simply be set to zero - hence starting as the "naive" formulation - or with a forecast of the monthly peaks. Methods such as the one described in [174] could be used for estimating the peak for the entire month. The vector tou_period maps the time instants to their TOU periods.

Algorithm 3 Incremental MPC

```
procedure UPDATEMAXSEENPEAK(period $\rho$, power $p$)
    $x_\rho^{thr} \leftarrow max(x_\rho^{thr}, p)$
end procedure
procedure RUNMPC(time $t$)
    $p \leftarrow get\_current\_consumption(t)$
    UPDATEMAXSEENPEAK(tou_periods[t], p)
    $u \leftarrow$ solve Eq. (5.10)
    return u
end procedure
```

Periodically, at each supervised control interval dt, the $runMPC()$ function is called. It starts by comparing the current building consumption p with the past peak values, to update the new peak power consumption for the corresponding TOU period through $updateMaxSeenPeak()$. Then, with the updated threshold values x_ρ^{thr}, the constrained optimization problem Eq. (5.10) can be solved. The function returns the output of the latter, as a list of optimal setpoints for the flexible entities. Then, at the next control instant dt, the same steps are repeated.

5.3 MPC Formulations Comparison: Case Studies

This section presents simulations results to compare the aforementioned MPC formulations and their impact on monthly metrics. We start by presenting two simulation test cases. The first one aims to establish best case theoretical metric values on simplified models. The second test case uses a more detailed model to discuss qualitative effects of each method. Finally, we describe the actual implementation of such a MPC controller on the *Solar+* Blue-Lake Rancheria (BLR) test site, a microgrid research project involving advanced control.

5.3.1 Case Study 1 - Simplified simulated commercial buildings

Building model and data

The Case Study 1 highlights the impacts of various MPC approaches on building owner's bill, shifting load potential and peak demand. The study consists in multiple monthly simulations of commercial buildings energy consumption. It intends to provide a best-case quantitative analysis of the grid-level effects induced by each of the control strategies.

The simplified commercial building model is shown in Fig. 5.4: a unique zone represents the entire commercial building, whose homogeneous temperature evolves depending on the RC components and the various sources of heat. The flexible AC electrical power demand $u(t)$

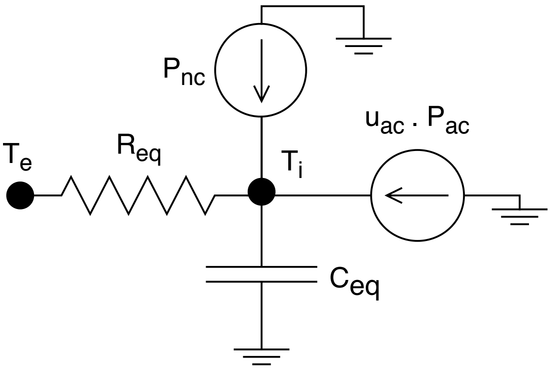


Figure 5.4 – Case Study 1: Building RC thermal model

influences the temperature according to:

$$u(t) = u_{AC}(t) \cdot \frac{P_{AC}^{cap}}{k_{cop}} \tag{5.13}$$

where P_{AC}^{cap} (kW) is the maximum thermal power capacity of the AC unit (negative value), and k_{cop} is the average COP of the AC unit, considered to be independent of ambient conditions. The cooling demand $u_{AC}(t)$ can be tuned in a continuous way:

$$0 \leq u_{AC}(t) \leq 1 \tag{5.14}$$

The AC unit cools down the whole building according to the following equation [6]:

$$C_{eq} \cdot \dot{T}_i(t) = u_{AC}(t) \cdot P_{AC}^{cap} + P_{nc}(t) + \frac{1}{R_{eq}} \cdot [T_i(t) - T_e(t)] \tag{5.15}$$

where $T_i(t)$ and $T_e(t)$ are the internal and external temperature at time t, respectively; C_{eq} and R_{eq} are the equivalent capacitance and resistance of the thermal zone model, respectively. In this model, the uncontrollable loads dissipate electricity entirely as heat into the zone, in addition to human heat gain. Moreover, the solar heat gain has not been modeled directly in the Eq. (5.15), but has been transferred as a shift in outside temperature values. Given the simple R1C1 model in use and the direct link between outside air temperature and solar irradiance, this method simplifies the simulation.

In order to model the thermal storage of the building internal mass, a zone capacitance multiplier is applied to the air capacitance [175]:

$$C_{eq} = k_{mass} \cdot C_{air} \tag{5.16}$$

The thermal comfort of the commercial building occupants must be ensured at any time t:

$$T_i^{min}(t) \leq T_i(t) \leq T_i^{max}(t) \tag{5.17}$$

The chosen model simplifies drastically the behavior of real commercial buildings, and would

Table 5.3 – Case Study 1: Simulated buildings equivalent parameters

Parameter & data	Retail Store	Secondary School
R_{eq} [K/W]	$4.311e^{-4}$	$4.774e^{-5}$
C_{eq} [J/K]	$1.4e^{7}$	$1.5e^{8}$
k_{mass}	4 / 8	3 / 6
P_{AC}^{cap} [kW]	-94.5 / -154	-595 / -735
k_{cop}	3.5	3.5
$P_{nc}(t)$ [kW]	See Fig. 5.5 (left)	See Fig. 5.5 (right)
Tariff	A-10	E-19
$T_i^{min}(t)$ [°C]	21 from 6am to 9pm, 16 else	
$T_i^{max}(t)$ [°C]	24 from 6am to 9pm, 30 else	

therefore never be used to simulate a specific building accurately. However, rather than modeling a specific commercial building very accurately, this study intends to assess the grid-level impact of MPC control methods. The model is therefore suitable enough for this purpose, as it represents a generic commercial building, containing both uncontrollable load and flexible demand.

The DOE Commercial Building Dataset [176] has been used to derive commercial buildings simulation parameters in the aforementioned equations. The dataset gathers generic building information (*e.g.* thermal envelope, zones size, uncontrollable load profile magnitude) and schedules (*e.g.* occupancy, heating/cooling, internal gain) for a wide range of commercial buildings. Two distinct buildings have been chosen for this study, whose parameters are listed in Table 5.3:

- Retail store: a medium-size commercial building, made of few conditioned zones on a single level. Its internal mass coefficient can either be 4 (lightweight) or 8 (heavyweight). The aggregated uncontrollable load signal is shown in Fig. 5.5 (left). The week-days and week-ends profiles are both very similar, with a rise around 6.30am and a slow decrease starting at 5pm. The AC electrical capacity varies according to the ambient climate, and can be as high as the larger value of the uncontrollable load.
- Secondary school[2]: a large commercial building, made of > 40 conditioned zones, spread on two levels. Its internal mass coefficient can either be 3 (lightweight) or 6 (heavyweight). The aggregated uncontrollable load signal is shown in Fig. 5.5 (right). The week-ends profile differs greatly from the week-days profiles, as no one occupies the building; this means that peak demand is more likely to be set during a week day. In hot climate, the AC electrical capacity exceeds the maximum uncontrollable load power.

---

[2] The secondary school is considered as a "commercial building" by the utility, for its power profile shape and peak demand

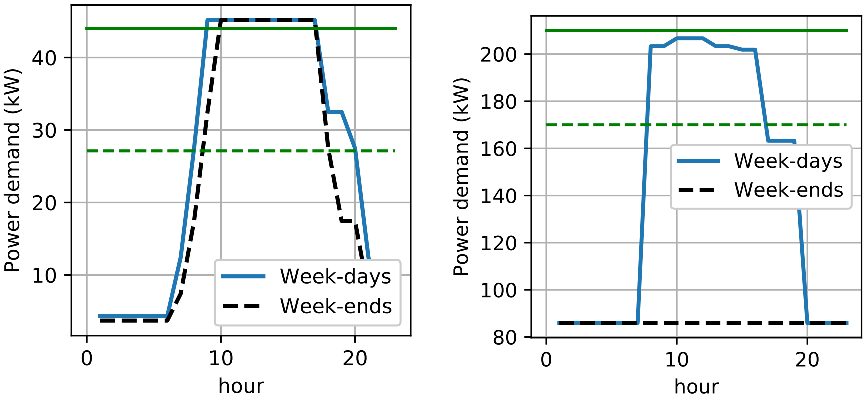


Figure 5.5 – Case Study 1: Uncontrollable load profile (left) Retail store (right) Secondary school. Maximum AC capacity in green: (filled) Warm and Hot environment, (dashed) Mild environment.

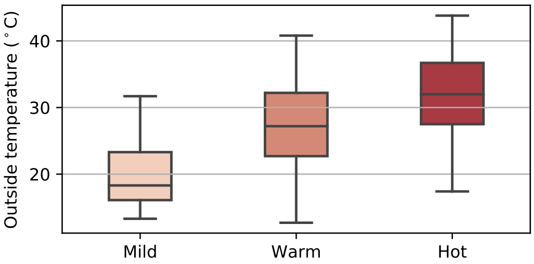


Figure 5.6 – Case Study 1: Outside temperatures in July, for 3 environments.

The outside temperature data depends on the climate zone (Mild, Warm, or Hot) and represents a typical month in the summer for three different cities in California [177]. Fig. 5.6 describes statistically the hourly signals used for the simulations. Depending on the simulated climate, the AC thermal capacity in each building takes a different value (see Table 5.3). A night setback applies to both buildings, relaxing the constraints on internal temperatures outside of occupancy hours.

MPC objective scenarios

Table 5.4 describes the MPC objectives and features implemented in this case study. The first one, "*A. Energy only*", is the most encountered in the literature and only optimizes on the energy part of the bill, hence setting $k_d = 0$ in Eq.(5.8). The second one, "*B. Peak best effort*" or "naive", strives to reduce the peak demand in the MPC horizon, setting k_d to the total demand cost in Eq.(5.8).
The method "*C1. Incremental TOU Multi-Peak*" implements the optimization as in Eq.(5.10), setting x_ρ^{thr} to the maximum peak already encountered earlier in the month and taking into

Table 5.4 – Case Study 1: description of the implemented MPC methods.

Scenarios Objective	A	B	C.1	C.2
Energy	X	X	X	X
Peak best effort		X		
Incremental TOU Multi-Peak			X	X
Max demand prediction				X

Table 5.5 – Case Study 1: PG&E summer tariff rates - off-peak = week-days 10pm - 8am, week-ends & holidays, mid-peak = week-days 8am-12pm & 6pm-10pm, on-peak = week-days 12pm-6pm.

	Energy ($/kWh)		Demand ($/kW)	
Time periods	A-10	E-19	A-10	E-19
off-peak	0.134	0.085	0	0
mid-peak	0.163	0.111	0	5.18
on-peak	0.218	0.152	0	18.64
All time	/	/	18.26	17.57
CPP increment	0.9	1.2	0	0

account the multiple TOU demand charges. At the start of the billing period, x_p^{thr} are set to zero. Compared to the former, the method "*C2. Incremental TOU Multi-Peak with prediction*" benefits from the knowledge of the maximum demand that will occur during the month, stored in x_p^{thr} at the beginning of the month. For practical purpose of these simulations, these forecast peaks are retrieved from the simulation results of method B.

This study considers two different TOU commercial tariffs of PG&E, the electrical utility in North California, described in Table 5.5. Both tariffs have TOU energy rates and a demand charge applied to all time periods of the month. In addition, tariff E-19 includes TOU demand charges, adding peak demand costs during mid-peak and on-peak hours. On top of these basic rates, PG&E can also trigger CPP events from 2pm to 6pm [3]. These events are generally called upon a very hot day, with a limit of 10 to 15 per year. In order to assess the shifting capability under realistic DR events, three CPP events increase the prices of energy for the three hottest days: the 8th, 17th, and 27th of July.

Due to the simplicity of the model used for the simulations, the results in this section represent the best-case scenario that could be encountered in real-life applications. A major hypothesis is to consider the same model for the MPC and the simulated building. Though unrealistic,

---

[3] The main 3 Investor-Owned Utilities (IOU)s in California changed their TOU structures in 2019 to better face RES production, but this study used the structure of 2018.

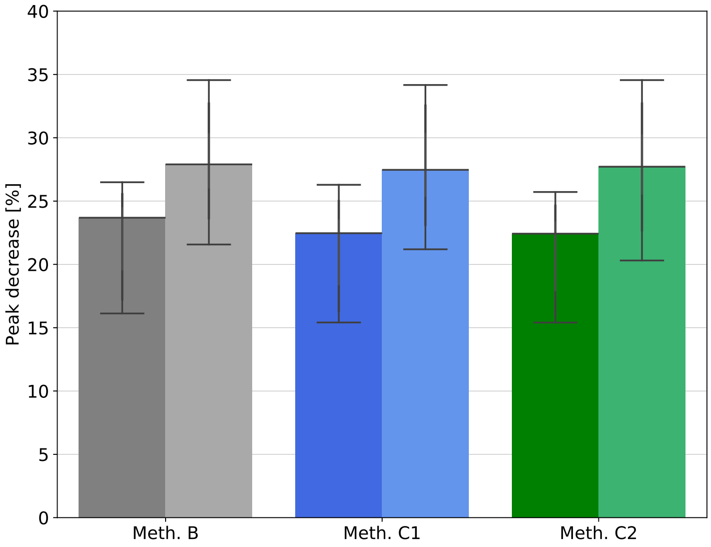


Figure 5.7 – Case Study 1: Relative *decrease* of the maximum power demand of the month, compared to control Method A. For each control method: (left bar) Retail store, (right bar) Secondary school.

this is useful in this case to mitigate the inherent inaccuracy of the control model.

Applying the various MPC methods to simulations of different buildings and environments allows for a sensitivity analysis of the results, instead of focusing on a specific configuration. In this study, the MPC methods of Table 5.4 are evaluated on all possible configuration triplets {blg type, internal mass, climate}. This corresponds to a total of 12 simulations per MPC method. Every simulation spans over an entire month, with an MPC update every 15 minutes. At each time step, the internal temperature is kept within bounds, while optimizing the objective over a time horizon. The MPC horizon is set to 12 hours, large enough to foresee the price, temperature, and uncontrollable power variation. From an implementation standpoint, the Python package *cvxpy* wraps the optimization formulations of Eq. (5.10) and calls the open-source ECOS solver [178]; the package *control* discretizes the continuous thermal model Eq. (5.15), for both the simulation and the MPC model.

Results and discussion

The two simulated buildings differ in their average power consumption by almost one order of magnitude. To place them together on the same graphs, a reference MPC scenario is needed. As the most encountered in the literature, scenario "*A. Energy only*" represents the ideal baseline. Therefore, instead of observing absolute metrics for each MPC scenario in Table 5.4, this section presents a relative comparison with respect to scenario "*A. Energy only*". The following metrics are used to compare scenarios B, C.1, and C.2 to the baseline A:

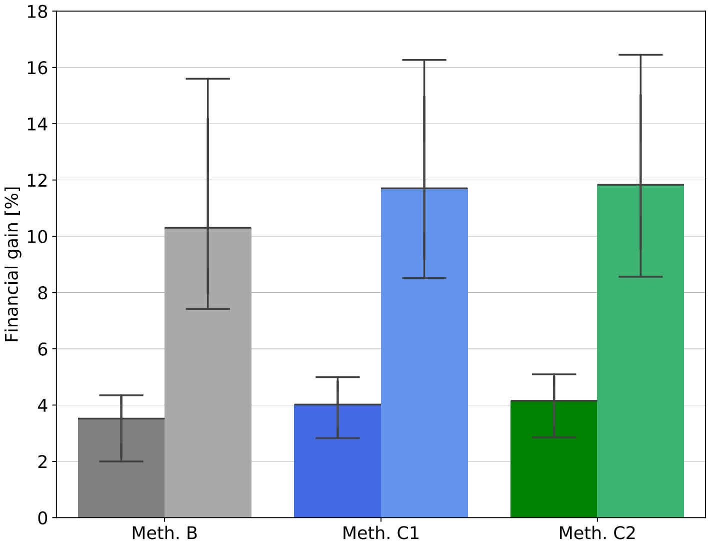


Figure 5.8 – Case Study 1: Relative *decrease* of the monthly bill, compared to control Method A. For each control method: (left bar) Retail store, (right bar) Secondary school.

- **Maximum peak demand**: the maximum power demand (kW) throughout the month, averaged over 15 minutes.
- **Monthly bill**: the bill ($) paid by the building owner at the end of the month, according to Eq.(5.1).
- **Load shifting capacity**: the energy (kWh) consumed during CPP DR events, initiated by the utility.

Fig. 5.7 shows the relative decrease of the **maximum peak demand** for the control methods B, C.1, & C.2 with respect to the control method A. One observes a tremendous decrease of the peak demand, ranging from 15% to 35%, due to the fact that the method A does not take the demand cost into account. The secondary school displays a larger gain than the retail store. The explanation is twofold. First, the proportion of controllable to uncontrollable load is higher for the secondary school, on average over time. Second, the tariff E-19, which the secondary school falls under, penalizes demand more than the A-10. The MPC formulations will therefore strive to reduce much more the maximum peak for each receding horizon H. As for the peak performance, the method B reduces peak demand slightly more than the methods C.1&C.2 for all of the cases. This is due to the method B using the full demand cost as the weight to penalize the demand over the MPC horizon. Nevertheless, the incremental methods C.1&C.2 only worsen the peak by 2%, in the worst case.

Fig. 5.8 shows the relative decrease of the **monthly bill** for the control methods B, C.1, & C.2 with respect to the control method A. The large gap between the retail store and the secondary school is mainly driven by the lower peak to average ratio of the latter, strongly penalized by

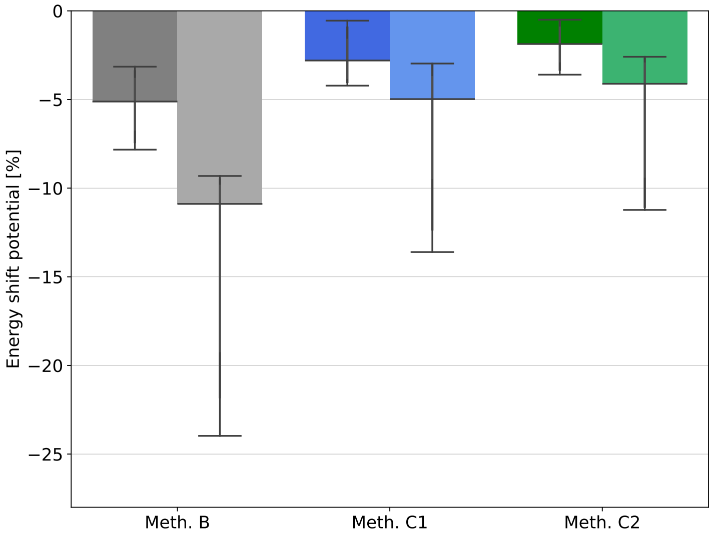


Figure 5.9 – Case Study 1: Relative *loss* of energy shifting potential during CPP events, compared to control Method A. For each control method: (left bar) Retail store, (right bar) Secondary school.

the tariff E-19 with higher demand charges. Methods C.1 & C.2 show an improvement of about 1-2% compared to the method B. The improvement can be explained by a better management of the TOU demand charges, whereas the method B does not differentiate the periods of the day. Moreover, the incremental demand charge feature of C.1 & C.2 enables them to dictate load shifting with energy rate fluctuations. The knowledge of the maximum peak in the month with method C.2 slightly improves the bill compared to method C.1. This feature leads to better pre-cooling decisions during the beginning of the month, where method C.1 does not risk setting a new peak. While the bill gain difference seems low, it is to be compared with an already-optimized system. Reducing the monthly bill through MPC software improvement does not induce any additional investment.

Fig. 5.9 shows the relative decrease (negative increase) of the **load shifting capacity** for the control methods B, C.1, & C.2 with respect to the control method A. This metric is computed by summing the energy consumption during the three CPP events of the month. Method A is therefore the best, as it only optimizes on energy cost and can shift the demand as much as possible, while respecting the system constraints. Compared to this baseline, method B clearly lacks the ability to shift load, especially under a tariff that strongly penalizes the peak demand (secondary school). Incremental methods C.1 & C.2 are capable of cutting the loss of load shifting potential in half compared to method B. This effect would even be more marked as the proportion of controllable load is increased. This is due to the fact that these methods leverage the knowledge of past behavior and prediction of future behavior. They are specifically being able to trade off cost savings from load shifting to cost increases from setting a new demand peak. The prediction of monthly maximum peak demand - method C.2 compared to method

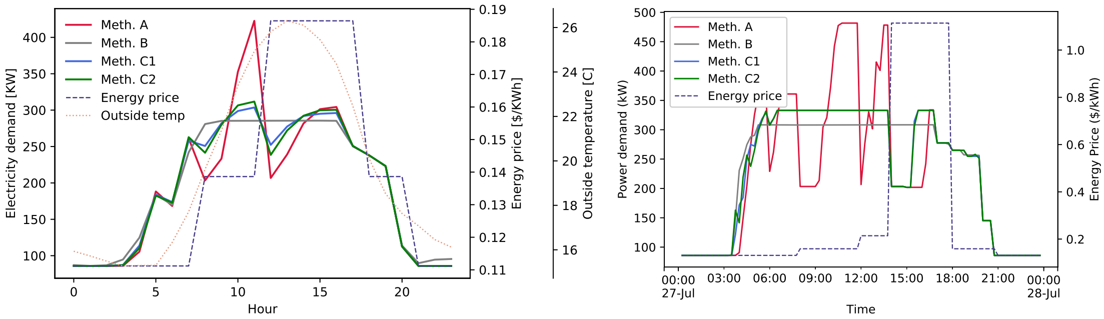


Figure 5.10 – Case Study 1: Timeseries analysis of "secondary school" control methods (left) hourly mean of power consumption, energy price and outside temperature throughout all simulations (right) 15-min power consumption on the 3rd CPP event.

C.1 - improves the shifting potential by about 1-2%.

Analyzing the details of timeseries power consumption results allows for further understanding of metric trade-offs and global trend of each method. Fig. 5.10 (left) plots the hourly power consumption induced by each control method, averaged over all of the simulations of the secondary school except the CPP days. The four methods exhibit the same behavior in the early morning (until 6am) and at the end of the day (from 5pm), due to the low risk of setting a new monthly peak. Just before the first TOU energy rate increment appears, they all pre-cool the zone. However, subsequent behavior diverges. Method A reduces the energy as much as possible, whereas incremental methods C.1&C.2 only slightly reduce it. Method B keeps a constant increase in power demand, disregarding the energy rate increase. At the hours before the on-peak period, the method A fully pre-cools the building to the lower bound of temperature to allow maximum free-float during the subsequent period of higher energy price. Methods C.1&C.2 enable some pre-cooling, though better account for the tradeoff between energy shifting and setting a new peak demand. The monthly peak demand prediction in method C.2 allows it to provide slightly more energy shift, as it knows the monthly peak will be set later anyway. Method B does not pre-cool at all since the demand cost prevails on the energy cost.

Fig. 5.10 (right) presents the last CPP day of the month, for a simulation of the secondary school (warm weather, large internal mass). On this day, the hot temperature prevents methods C.1 and C.2 from pre-cooling too much without setting a new costly demand peak, hence looking alike method B. Nevertheless, more pre-cooling during the morning than method B still allows for more load reduction during CPP hours, though less than method A.

5.3.2 Case study 2 - Advanced simulated commercial buildings

Although suitable for the purpose of the monthly simulations, Case Study 1 uses unrealistic R1C1 building model in Eq. (5.15) to drive the AC power consumption. The main limitation lies in the unique capacitance that accumulates a mix of energy coming from internal air

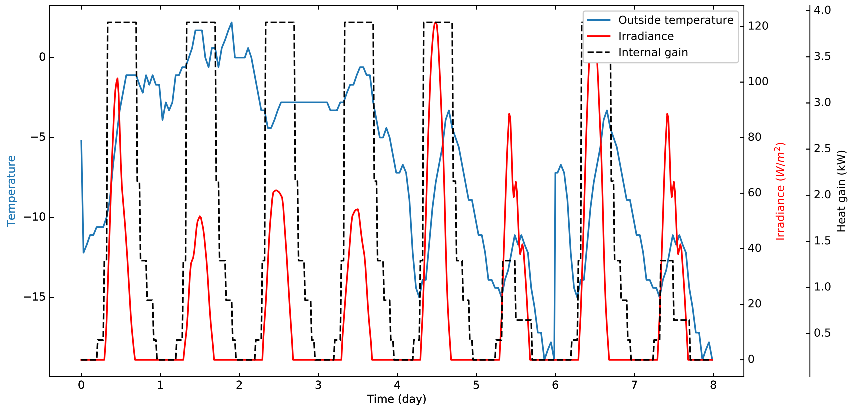


Figure 5.11 – Case Study 2: environmental conditions

and internal mass heat. Commercial buildings generally cools down the incoming air that circulates in it, to regulate the thermal comfort. The air heat exchange also impacts the building mass temperature, such as furniture and walls, but with a delay that the R1C1 does not capture. When used in a MPC controller on an actual commercial building, this would result in a too large AC power set point and therefore violation of comfort constraints.

In this Case Study 2, an advanced linear model has been used to analyse the different MPC formulations presented earlier in this chapter. The building consists in three interconnected zones, each containing their own HVAC system. Disturbances impact their temperature, namely outside air temperature, solar irradiation, and internal load heat gain. An EnergyPlus model has been linearized via OpenBuild toolbox [46], resulting in the following linear model:

$$x[t+1] = A\,x[t] + B_u\,u[t] + B_d\,d[t] \tag{5.18}$$

$$T[t] = C\,x[t] \tag{5.19}$$

where $x[t]$ is a vector holding the building state at time instant t, with no straightforward physical meaning, and $T[t]$ is the zone temperature at time instant t. Values of matrices A, B_u, B_d, C as well as initial building state are provided in Appendix A.1.2. The initial continuous model has been discretized with a sampling period of 20 minutes.

The commercial building must provide thermal comfort through the control of three heaters positioned in each of the rooms. Zone temperatures must ideally stay in range of 21-24 $°C$, and HVAC power are limited to 15 kW each. The simulated disturbance can be seen in Fig. 5.11. A cold winter month has been selected, when outside temperatures rarely exceed $0°C$ and a weak solar radiation impacts the building state. The internal load supplies uncontrollable heat throughout the three zones.

The price of electricity is artificially generated as follows, identical every day. The building pays the energy: 0.2 $/kWh from 00:00am to 04:00am, 1.2 $/kWh from 04:00am to 08:00am,

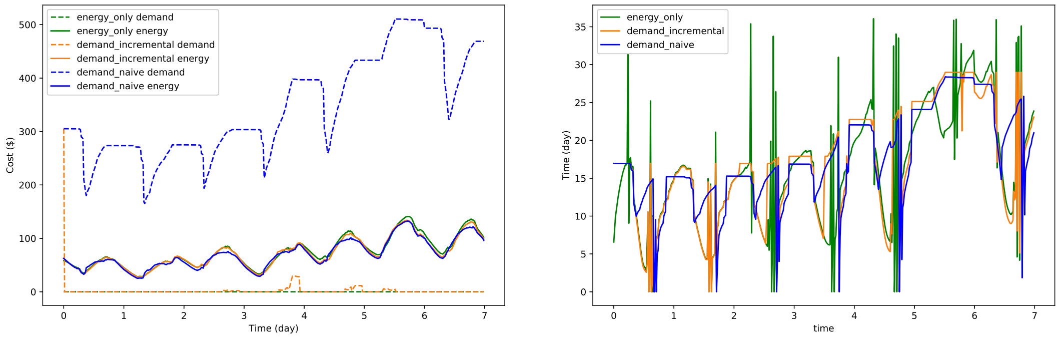


Figure 5.12 – Case Study 2: (left) MPC cost function (right) Power consumption

0.2 \$/kWh from 08:00am to 04:00pm, 1.2 \$/kWh from 04:00pm to 08:00pm, and 0.2 \$/kWh from 08:00pm to 00:00am. The building pays for the demand peaks: 18 \$/kW regardless the period in addition to 20 \$/kW for the peak occurring between 12pm to 6pm.

Results and discussion

Weekly simulations have been performed, driven by the same MPC formulations as in Table 5.4 with the exception of scenario C.2. Python package *cvxpy* wraps up the MPC constraints/objectives and Gurobi solver [161] finds the optimal trajectory at each control step fixed to 20 minutes. In this section, we present a qualitative analysis of the various MPC formulations, by analysis objective functions, power profiles, computational time, and temperature evolution.

The common approach to tackle demand charges found in the literature (Scenario B "naive") overestimates the importance of peak cost with respect to energy cost in each MPC iteration. Fig. 5.12 (left) shows the share of energy and demand in the objective function, at each iteration. Fig. 5.12 (right) depicts the total building consumption and helps understanding the link with the objective functions. One clearly observes a high share of demand cost when scenario B is applied (blue dashed line). Yet, the energy cost does not dramatically vary compared to the other two methods (A and C) at the beginning of the simulation, as one can see the three plain lines superimposed. However, as the ambient conditions become harsher at day 4, the wrong demand cost in the prediction pushes down the energy cost, compared to methods A and C. The incremental method (orange lines) behaves like the "naive" method at the very beginning of the month, due to a threshold x_p^{thr} fixed to zero. Right after and until the end of day 2, the mild ambient conditions drive the increment demand cost to zero, hence the method optimizes the objective like scenario A (Energy only). Arrived at the end of day 3, the harsher ambient temperature drives the incremental method energy cost towards the "naive" approach one. Day 4 shows two different behaviors for the incremental method: during low internal gain, the system is forced to ensure minimal thermal comfort while minimizing the peak, similar to the "naive" one. At the end of the day, as internal gains are accumulated in the structure of the building and demand charge drops, the incremental method behaves

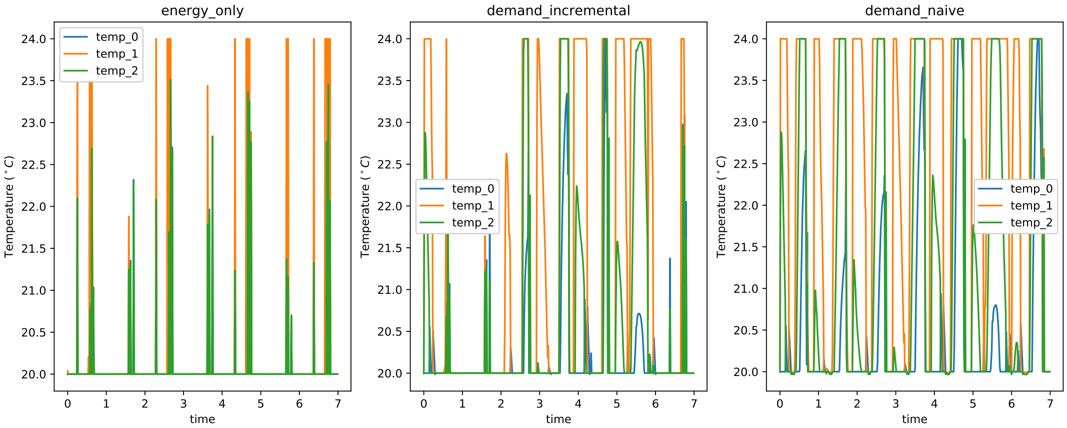


Figure 5.13 – Case Study 2: zones temperature comfort

like the scenario A, and dares to set a new demand peak charge. The harsher day 6 leaves no opportunity to the incremental method to relax its energy cost share, and must even set a new peak. The rest of the simulation follow the same duality for the incremental method: a behavior similar to the "naive" approach when ambient conditions are jeopardizing the peak demands while leveraging the knowledge of the past peaks, and a consumption/objective identical to "Energy only" when these conditions are relaxed.

Evolution of the three zones temperature are displayed in Fig. 5.13, for each of the control methods. The method A. Energy Only clearly aims to minimize energy cost and therefore keeps temperatures to their lower bound, except before each price transition. When energy prices increase, the MPC controller pre-heats the zones as much as possible, for one or few time steps. The incremental approach tends to pre-heat each zone using lower power inputs, but keeps the state at the highest temperature bound for a longer time. This case only happens with an advanced lumped model R3C3 model and would never happen with the R1C1 as a control model. Indeed, even though the first capacitance might be fully charged - and so the sensed temperature at the highest - the resulting heat transfer toward the second capacitance can greatly benefit the resulting thermal inertia and therefore lower the subsequent energy costs. This effect is even more pronounced in method "naive", where the two largest zones stay at maximum temperature for a few hours before each price transition. In that case, the temperatures decrease more slowly than in the previous cases.

We close this second case study by looking at the computational time for each MPC method, in Fig. 5.14. Faster than the other two, method A takes on average 170 ms to complete the entire MPC step. The common "naive" approach spends more time solving the problem, and shows a larger variance in the computational times. Close behind, the incremental method is the slowest with an average 240 ms to solve the MPC optimization. However, unlike the other two, some MPC control steps can take twice as much time to be solved in comparison to the average. This can be explained by the discontinuity induced by the incremental peak variable in Eq. (5.12), jumping from 0 to a non-zero value to penalize.

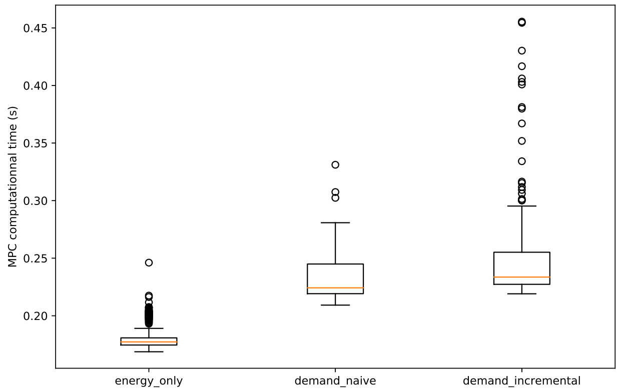


Figure 5.14 – Case Study 2: comparison of MPC computational time

5.4 Conclusion

This chapter reviewed existing economic MPC formulations applied to commercial buildings and presented a new *incremental* method that leverages past behavior of the building. Two simulated case studies compared the MPC formulations with the proposed approach. Results showed that the traditional MPC method of taking into account demand charges reduces both the peak demand and the electricity bill relative to the solution optimizing for energy costs only. However, it prevents the building from shifting load when needed, such as during CPP events. Compared to that "naive" approach, the incremental formulation improved the building responsiveness to price-based DR signals, while similarly keeping constant or even slightly reducing the owner's bill and maximum peak demand.

Simulation on a more realistic building model highlighted that building inherent thermal mass actually reduces the HVAC peak to spread the pre-cooling/heating on a larger time frame. It's also been observed that, depending on the ambient conditions, the proposed *incremental* MPC method constitutes an ideal trade-off between the methods optimizing only on the energy or on the peak. When facing harsh ambient conditions or high costs of demand, the proposed controller acts like the "naive" one, preventing peaks as much as possible. On periods when demand/energy costs decrease and ambient conditions do not push the physical loads to their limit, the controller behaves like the one optimizing solely on energy cost, hence providing the most energy flexibility to the grid.

The simulation results highlight that multiple MPC formulations can offer the same value for the building owner (in terms of utility monthly bill cost) but different grid service capabilities. Ongoing world decarbonisation efforts increasingly encourage the deployment of price-based DR programs to incentivize load shifting and peak load reduction. The tariff structures should, therefore, be carefully designed to optimally leverage building load flexibility offered by different MPC formulations. Furthermore, differences in building behavior will be accelerated by the electrification of building, that will offer a greater controllability. The grid operator will have to deal with such a large fleet of flexible buildings, controllable via prices of electricity, and therefore account for the variety of available building controllers.

6 Decentralized Demand Response in a Community of Smart-Buildings

The increasing penetration of distributed Renewable Energy System (RES) calls for a restructuring of the electrical grid and a change of paradigm in the way electricity is consumed. To this end, Demand Response (DR) is progressively moving from a centralized, unidirectional structure to a set of advanced, decentralized mechanisms that better balance distributed supply and demand. Yet, the large fleets of distributed flexible entities must be carefully handled, as it involves complex coordination of decentralized energy actors with different interests, heterogeneous technology, and data privacy requirements of participants.

The main **highlights** and **contributions** of this chapter are:

- We present a decentralized framework to autonomously manage the day-ahead energy planning of a community of smart-buildings, in the presence of local RES.
- The framework is enabled by blockchain, that allows the trustworthy decentralization of the algorithm. We presented a common economically-driven objective in this chapter, although the modularity of the framework allows any community to optimize a specific common objective, such as greenhouse gas emission reduction.
- Ethereum smart-contracts practically deploy and orchestrate the day-ahead decentralized algorithm, and the real-time monitoring and billing of participants.
- Simulations are performed on a community of realistic Swiss Minergie buildings, to analyse both the scalability and the energy-related metrics. Given the current Ethereum version, the framework is proven to handle up to 100 smart-buildings.

Related **publications**:

[179] O. Van Cutsem, D. Ho Dac, P. Boudou, and M. Kayal, "Cooperative energy management of a community of smart-buildings: a blockchain approach," *International Journal of Electrical Power & Energy Systems*, vol 117, pp 105643, 2020.

Table 6.1 – Nomenclature used in Chapter 6: parameters

Symbol	Description	Unit
α_b	Battery leakage coefficient	s^{-1}
η_b^+,η_b^-	Battery charging and discharging efficiency	/
a_l	Electricity cost of local production	\$/kWh
a_g^q	Quadratic cost of grid import	\$/kWh^2
a_g^l	Linear cost of grid import	\$/kWh
a_g^c	Constant cost of grid import	\$
$\overline{C}_b^k$	Max. k^{th} battery capacity	kWh
$\underline{C}_b^k$	Min. k^{th} battery capacity	kWh
$C_{b,a}^k$	Charge of the k^{th} EV upon arrival	kWh
$C_{b,l}^k$	Charge of the k^{th} EV when leaving	kWh
d_w^k	Water drawn from the k^{th} hot water tank	l/s
dt	Sampling period	s
H	Number of intervals in the planning forecast	/
l_*^+, l_*^-	Snapshot of import/export community forecast	kW
N_{sb}	Number of smart-buildings in the community	/
N_{res}	Number of RES in the community	/
n_{pv}^k	Number of cells on the k^{th} PV panel	/
$\overline{P}_{th}^k$	Max. power of the k^{th} thermal load	kW
$\overline{P}_b^k$	Max. charging power of the k^{th} battery	kW
$\underline{P}_b^k$	Max. discharging power of the k^{th} battery	kW
$\underline{P}_{pv}^n$	Nominal power of the PV system	kW
$\hat{\mathscr{P}}_{\boldsymbol{d}}^{\boldsymbol{k}}$	Power profile of the k^{th} deferrable load	kW
$t_d^{s,k}$	Min. starting time of the k^{th} deferrable load	h
$t_d^{e,k}$	Max. ending time of the k^{th} deferrable load	h
T_a	Outside air temperature	°C
$\mathscr{T}_{out}^k$	The set of periods in which the k^{th} EV is unplugged	/
$t_{b,l}^k$	The leaving time of the k^{th} EV	h
$t_{b,a}^k$	The arriving time of the k^{th} EV	h

6.1 Introduction and Motivation

The increasing world demand in electricity is envisioned to be entirely supplied by sustainable RES in order to counteract the global climate change. However, intrinsic volatility and uncontrollably of decentralized RES production pose severe challenges to the current electrical grid, that must ensure stability at any time. Progressively, the centralized grid sees a change in paradigms, transitioning from a system dispatching a production portfolio following the electrical demand to a smart-grid that handles a portfolio of controllable demand to match an

Table 6.2 – Nomenclature used in Chapter 6: variables

Symbol	Description	Unit
e	Building net demand	kW
p_d^k	k^{th} deferrable load power	kW
u_{th}^k	k^{th} thermal load power	kW
u_d^k	k^{th} deferrable load starting time	h
u_b^k	Power of the k^{th} battery	kW
$u_b^{+,k}$	Charging power of the k^{th} battery	kW
$u_b^{-,k}$	Discharging power of the k^{th} battery	kW
x_b^k	Charge state of the k^{th} battery	kWh
y^+, y^-	Community grid import/export	kW
y_l^-	Community power generation	kW

Table 6.3 – Nomenclature used in Chapter 6: conventions and notations

Notation	Description
$\hat{\bullet}$	Vector $[\bullet_0, ..., \bullet_n]$
$\bullet[h]$	Value at discrete time period h
$\bullet^+$	Positive power demand related value
$\bullet^-$	Negative power demand related value
$\underline{\bullet}, \overline{\bullet}$	Minimum/Maximum value
k	The index of an energy-related entity in a specific building
$\mathcal{N}$	Normal probability distribution
$\mathcal{U}$	Uniform probability distribution

uncontrollable supply [180].

To assist this transition, smart-buildings have recently emerged as a solution to leverage the flexibility offered by the various entities commonly found in buildings (appliances, thermal loads, lights, storage, and local generation). Equipped with the right hardware and ICT, they can provide active DR to the electrical grid [150, 156]. DR regroups a set of mechanisms divided into *incentive* and *price-based* programs, that specify various signals to be exchanged between the grid and the consumers in order to shape the power profile of the latter. Many works have tackled the problem at individual building level, demonstrating their capability to adapt their power consumption to grid signals while ensuring occupant comfort [181, 134, 182, 169].

Beyond individual building optimization, there is a need of handling the problem at the community level. By doing so, local resources such as PV production can optimally be harnessed and the aggregated overall peak power demand can be reduced [183]. Many optimization frameworks have emerged to collectively manage the energy of multiple users [184, 185, 186, 187, 173]. Nevertheless, these solutions require a central agent that collects user information to subsequently dispatch optimal set points to each of them. Even though generic simplified models can be used to represent buildings flexibility [185, 173], centralized solutions still face issues of privacy, single point of failure, scalability, and challenging market entry of small prosumers. Furthermore, the growing penetration of distributed RES leads to the need of decentralized and distributed DR solutions that become complex to solve centrally when considering a large community of flexible assets.

Game-Theory (GT) [188] and Peer-to-Peer (P2P) [189] energy trading have extensively been applied to energy scheduling in local microgrids/communities. On the one hand, GT defines a conceptual framework in which the individual actions of rational participants optimize a community objective [188, 190, 127, 191]. Notably, Mohsenian-Rad et al. [127] decentralized the central grid planning optimization problem, such that every participant in a microgrid solves locally a load scheduling problem taking into account the predictions sent by the others. The quadratic structure of the price of electricity incentivizes the whole community to reduce the aggregated PAR. On the other hand, P2P energy trading represents the virtual exchange of electricity among community participants, with the aim to locally match production and consumption[189]. P2P energy generally lays on GT principles to fix the price of energy transaction. In [192], a shared Energy Sharing Provider (ESP) is in charge of deciding the local prices, through an iterative process that involves all the local participants. Each consumer optimizes its objective function that is modelled as a combination of energy cost and inconvenience in load shifting. Furthermore, local markets such as the one presented in [193] help incentivizing local participants to shift their energy or optimizing the use a shared battery, by formalizing a framework that fairly shares revenues/costs among the users.

Renowned for the cryptocurrency applications [194], the blockchain has rapidly proven capabilities for energy trading and optimization in microgrids. Blockchains are distributed ledgers shared by participants that can securely store digital transactions, without the need of a central agent [195]. These transactions are aggregated into blocks, linked to one another through cryptography methods to form a chain of immutable information. When adding smart-contracts into the blockchain, like the Ethereum technology [196], decentralized algorithms can practically be deployed. A smart-contract is a piece of executable code shared by every node that defines immutable rules, running directly in the blockchain. This replaces the need of a centralized trusted entity to hold the algorithm logic, and can therefore foster the fast deployment of innovative community DR solutions. Many significant energy trading projects using the blockchain technology have been deployed worldwide, notably the Brooklyn Microgrid projects [197] and various research demonstrators [198, 199, 200]. In addition to the project description, Mengelkamp et al. formally present the seven market components that any efficient microgrid energy market framework should incorporate [197]. Authors of [201] demonstrated that the blockchain technology represents a reliable mechanism for energy trading, compared to traditional centralized transactive energy schemes.

The use of blockchain in microgrid goes beyond local energy trading, as thoroughly reviewed in [202, 195]. For instance, authors in [203] were the first to use smart-contracts in distributed optimization, relying on them to play the role of an ADMM coordinator. In [204], authors applied blockchain to deploy a GT algorithm that solves a DR problem and discuss how P2P trading could be incorporated in their work. Pop et al. [205] developed a decentralized solution to manage and monitor DR with the use of blockchain. Their smart-contracts penalize the gap between the expected baseline and the actual consumption, and manage in real-time the microgrid imbalance.

In the present chapter, we propose an innovative generic framework to manage the energy in a community of flexible smart-buildings with local RES production. Unlike P2P energy trading [197, 198, 199, 200] that optimizes the individual costs, the framework at hand allows participants to collectively optimize any generic objective, such as grid services or promoting local RES energy consumption. Practically, the framework works in two phases. During the planning phase, the participants iterative propose a forecast of their power profile - consumption and/or production - until a consensus is collectively found. Compare to the iterative algorithm in [127], the planning phase also includes RES production and a generic bottom-up model of the buildings. Then, during the online phase, the participants ensure that their power profiles matches as much as possible their planning forecast.

Although sharing some similarities, our work is different from the frameworks of Noor et al. [204] and Pop et al. [205]. In this study, we use a generic building model that better represents end-user flexibility such as thermal inertia, and we directly included local RES in the day-ahead optimization problem. Unlike [205] that used past data to construct the building baseline against which the participant is rewarded/penalized, the presented day-ahead decentralized algorithm defines the building baseline itself, allowing more flexibility in the decision. Moreover, we presented a reward/penalty decision based on the community behavior, instead of individual participant actions. The use of blockchain, and more particularly Ethereum smart-contracts, enables both the decentralization of the energy management algorithms among untrustworthy participants and the monitoring of the community in real-time. Beyond the interest of blockchain for price-based P2P energy trading leveraged in [204], we used it to empower smart-communities to collectively manage their energy in a flexible way according to their common interests. Finally, the simulation code is publicly available [61], and its modularity allows any developer to include additional models.

This chapter is organized as follow. A theoretical background on blockchain and smart-contracts is provided in Section 6.1.1. Then, Section 6.2 presents the generic smart-building model used in this study. Section 6.3 describes the community optimization framework with its generic community objectives, and the various smart-contracts to decentralize the logic. In Section 6.4, results of simulations and a general discussion are presented. The chapter is concluded in Section 6.5.

6.1.1 Blockchain and smart-contracts

Definition 6.1.1. Blockchain - A blockchain is a shared, distributed ledger that enables the

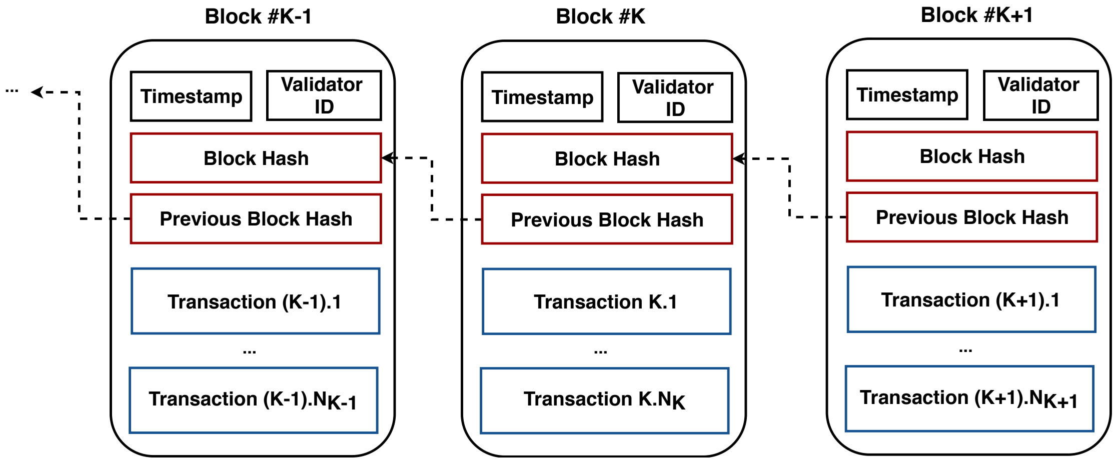


Figure 6.1 – Blockchain is a list of *blocks*, linked to one another by the unique previous block hash. Each block i contains N_i transactions.

process of recording transactions and tracking assets in a network [206].

The blockchain concept has been initially presented in [194] and its applications go now far beyond cryptocurrency. In its generic form, it represents a reliable network that enables transactions between anonymous users in a transparent and trustworthy way, useful for decentralized computation and data storage. Authors in [206] compared blockchain to an OS, for it allows various applications - such as Bitcoin [194] - to leverage its numerous services. However, unlike a central OS, blockchain relies on distributed nodes that work together, autonomously orchestrated by a distributed consensus mechanism and secured by a set of cryptographic functions. Hence, it removes the need of a central agent to perform the same tasks. Compared to centralized solutions, it is proven to be more:

- Cost-effective, as it removes the need of intermediaries.
- Efficient, because once a transaction is recorded, it becomes available to all the network participants.
- Safe and secure, for the shared ledger is tamperproof and immutable: a transaction cannot be changed, any new transaction is added in the right order, and the participants can access the agreed blockchain state even when facing cyber-attack or change in participants.

Blockchain gets its name for the resulting chain of *blocks* it generates over time, as network participants create transactions and these transactions are validated by the decentralized nodes. As illustrated in Figure 6.1, a block contains a bundle of transactions and necessary header data. Among this header data one finds the block number, the block hash code, the block timestamp, and the ID of the validator node. In addition, the block i stores the hash code of the block $i-1$, which ensures the right order of blocks for auditing the system history and preventing content alteration.

In a public blockchain, any newly emitted transaction is verified by a group of independent nodes called *miners*. Proof-of-work consensus algorithms, used in Bitcoin and the current version of Ethereum, challenge the miners with a difficult puzzle to solve, whose solution proves the legitimacy of the transaction. The challenge consists in finding a nonce, that when the new block data is hashed with, the resulting hash is smaller than a certain threshold. As this problem is non deterministic, it requires brute force that takes time, hence leading to a proof of the CPU work. With such a mechanism, any malicious entity must own at least 51 % of the total CPU power in the network to be able to take control of the generated blocks. In a private blockchain, these consensus algorithms are replaced by a verification process through a set of authorized individuals [198].

Definition 6.1.2. Smart-contract - A smart-contract is a piece of code running in the blockchain that automatically transfer funds upon specific events, transactions, or time instants.

While blockchain lays the foundations for decentralized transactions, *smart-contracts* actually define the logic of decentralized applications. Although they already existed in the Bitcoin technology, the Ethereum platform allows programmers to define them in a flexible and modular way, on top of a blockchain-based network [196]. This results in a set of distributed, immutable rules that autonomously enforce new transactions in the blockchain following specific conditions. In the microgrid context, smart-contracts represent a powerful means for peer-to-peer energy exchange and, more generally, advanced decentralized algorithms.

6.2 Energy Actors Modeling

Prior to the description of the proposed blockchain-based energy management framework, this section presents a generic "(smart-)building model", mostly relying on the state-of-the-art models introduced in Chapter 1. The prosumers in the framework are assumed to be equipped with adequate BDMS and sensors/actuators to enable the management of their energy. Any variable/parameter that is not explained in the text can be found in the nomenclature's Tables 6.1, 6.2, and 6.3.

Definition 6.2.1. Prosumer - A prosumer refers to a building environment that is both capable of consuming and producing energy over time.

A smart-building - or prosumer in this context - is an entity in the community that consumes or produces a total power $e[h]$ (kW) at a given time period h, which can be broken down as follow:

$$e[h] = p_{nc}[h] + p_g[h] + u_f[h] \tag{6.1}$$

where p_{nc} represents the non-controllable part of the building load consumption (kW), p_g

the behind-the-meter power generation (kW), and u_f the total flexible load (kW). The latter sums up the following components:

$$u_f[h] = \sum_k^{n_{th}} u_{th}^k[h] + \sum_k^{n_d} p_d^k[h] + \sum_k^{n_b} u_b^k[h]$$

where n_{th} is the number of thermal loads, n_d the number of deferrable loads, and n_b the number of batteries present in the building. The rest of this section presents the models impacting the variable $e[h]$.

Thermal loads

The variable $\hat{\boldsymbol{u}}_{\boldsymbol{th}}$ (kW) influences the temperature of air zones where home occupants live or the temperature of the water in hydronic pipes/tanks, used by home occupants. The corresponding loads are HVAC system, HPs, and EWHs.

The conditioned zones/hydronic system state and water tank state evolve as follow [6]:

$$\hat{\boldsymbol{x}}_{\boldsymbol{th}}[h+1] = A\, \hat{\boldsymbol{x}}_{\boldsymbol{th}}[h] + B_u\, \hat{\boldsymbol{u}}_{\boldsymbol{th}}[h] + B_d\, \hat{\boldsymbol{d}}_{\boldsymbol{th}}[h] \tag{6.2}$$

$$\hat{\boldsymbol{T}}[h] = C\, \hat{\boldsymbol{x}}_{\boldsymbol{th}}[h] \tag{6.3}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{th}}$ (°C) regroups both the constrained thermal air/water temperatures and intermediate model states, such as wall temperatures. Thermal model disturbance vector $\hat{\boldsymbol{d}}_{\boldsymbol{th}}$ is made of outside temperature, internal load heat gain, and solar heat gain. The matrices A, B_u, B_d contain the building model parameters, such as RC equivalent model parameters and thermal load efficiency.

The air zone and water temperatures should ideally be kept within min/max limits and the thermal load power is physically limited:

$$\hat{\boldsymbol{T}}^{\boldsymbol{m}}[h] \leq \hat{\boldsymbol{T}}[h] \leq \hat{\boldsymbol{T}}^{\boldsymbol{M}}[h] \tag{6.4}$$

$$0 \leq \hat{\boldsymbol{u}}_{\boldsymbol{th}}[h] \leq \hat{\bar{\boldsymbol{P}}}_{\boldsymbol{th}} \tag{6.5}$$

where $T^{k,m}$ and $T^{k,M}$ (°C) are the k^{th} air zone or water tank temperature limits, specified by the user.

Deferrable loads

A deferrable load refers to any appliance whose discrete starting time $\hat{\boldsymbol{u}}_{\boldsymbol{d}}$ (h) can be controlled and that cannot be interrupted once started. This category regroups mainly residential loads like washing machines, dryers, and washers. When switched on, the deferrable load power consumption is given by its predefined load profile:

$$p_d^k[h] = \begin{cases} \mathscr{P}_d^k[h - u_d^k] & \text{if } u_d^k \leq h \leq u_d^k + |\mathscr{P}_d^k| \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

where $|\mathscr{P}_d^k|$ refers to the duration of the deferrable load profile. The model is constrained by home occupants preferences that can specify a minimum starting time and a maximum ending time:

$$\hat{\boldsymbol{t}}_{\boldsymbol{d}}^{\boldsymbol{s}} \leq \hat{\boldsymbol{u}}_{\boldsymbol{d}} \leq \hat{\boldsymbol{t}}_{\boldsymbol{d}}^{\boldsymbol{e}} - |\hat{\mathscr{P}}_{\boldsymbol{d}}| \tag{6.7}$$

Energy storage system

The continuous variable $\hat{\boldsymbol{u}}_{\boldsymbol{b}}$ (kW) influences the SoC of chemical batteries used in EV. An integrative model describes the SoC evolution [173]:

$$\hat{\boldsymbol{x}}_{\boldsymbol{b}}[h+1] = (\alpha_b\, \hat{\boldsymbol{x}}_{\boldsymbol{b}}[h] + \eta_b^+\, \hat{\boldsymbol{u}}_{\boldsymbol{b}}^+[h] + \frac{1}{\eta_b^-}\, \hat{\boldsymbol{u}}_{\boldsymbol{b}}^-[h])\; dt \tag{6.8}$$

$$\hat{\boldsymbol{u}}_{\boldsymbol{b}}[h] = \hat{\boldsymbol{u}}_{\boldsymbol{b}}^+[h] + \hat{\boldsymbol{u}}_{\boldsymbol{b}}^-[h] \tag{6.9}$$

Battery charging/discharging power and capacity are physically limited. In order to ensure that $u_b^{k,+}[h]$ and $u_b^{k,-}[h]$ are not simultaneously non-null, binary variables s_b^k[h] are injected in the constraints:

$$\underline{\hat{\boldsymbol{C}}}_{\boldsymbol{b}} \leq \hat{\boldsymbol{x}}_{\boldsymbol{b}}[h] \leq \hat{\overline{\boldsymbol{C}}}_{\boldsymbol{b}} \tag{6.10}$$

$$0 \leq \hat{\boldsymbol{u}}_{\boldsymbol{b}}^+[h] \leq \hat{\boldsymbol{s}}_{\boldsymbol{b}}[h]\; \hat{\overline{\boldsymbol{P}}}_{\boldsymbol{b}} \tag{6.11}$$

$$0 \geq \hat{\boldsymbol{u}}_{\boldsymbol{b}}^-[h] \geq (1 - \hat{\boldsymbol{s}}_{\boldsymbol{b}}[h])\; \underline{\hat{\boldsymbol{P}}}_{\boldsymbol{b}} \tag{6.12}$$

When the battery is used in an EV, initial and final conditions apply on SoC when unplugged, and at time of arrival and departure:

$$u_b^k[h] = 0\ \forall h \in \mathscr{T}_{out}^k \tag{6.13}$$

$$x_b^k[t_{b,a}^k] = C_{b,a}^k,\ x_b^k[t_{b,l}^k] = C_{b,l}^k \tag{6.14}$$

Solar PV panel

Environmental conditions influence the power p_g locally generated by PV array installation. The PV output power is linearly modelled as follow [207]:

$$p^g[h] = f_{PV}(G[h], T_a[h])\; n_{pv}^{cells}\; P_{pv}^n \tag{6.15}$$

where $f_{PV}(.)$ is a function that modulates the PV array nominal power, depending on outside solar irradiance and cell temperature difference from standard condition $\Delta T[h]$:

$$f^{PV} = \frac{G[h]}{G_n}(1 + \alpha_i\, \Delta T[h])(1 + \alpha_u\, \Delta T[h])$$

where G_n is the nominal radiation $\frac{W}{m^2}$, α_i and α_u are the temperature sensitivity ($\%/°C$) of the PV output current and voltage, respectively.

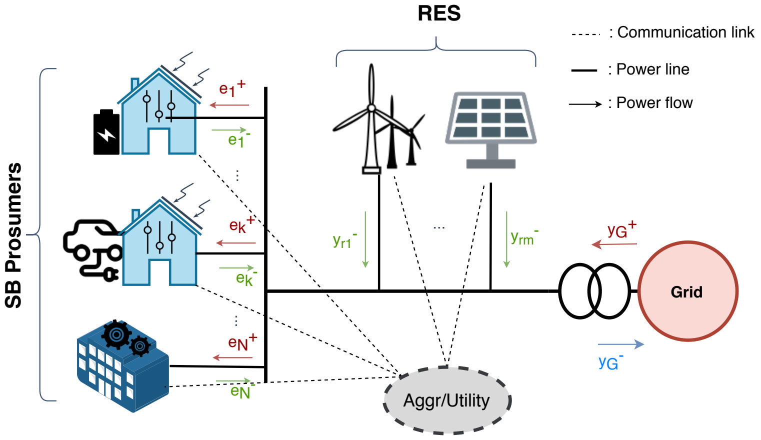


Figure 6.2 – Smart-Buildings community with local RES: structure, communication, and power flows: centralized aggregator approach

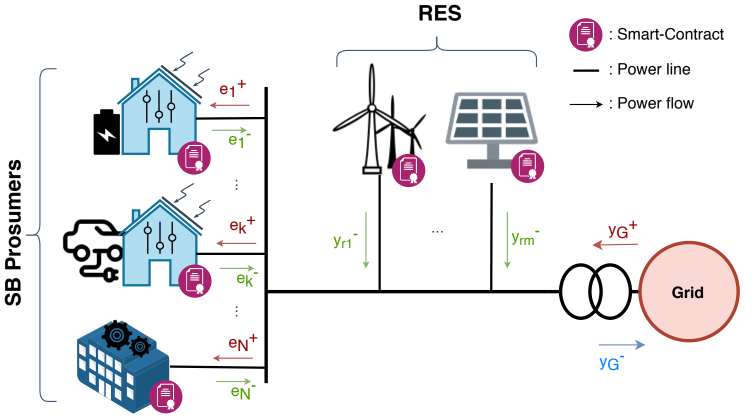


Figure 6.3 – Smart-Buildings community with local RES: structure, communication, and power flows: autonomous decentralized approach

Uncontrollable load

Building occupants and environmental conditions influence uncontrollable load behavior p_{nc}. Their power profile is supposed to be a given input parameter to the system.

6.3 Cooperative Decentralized DR Framework

The community encompasses the local energy actors, the physical power system through which electricity flows, and the energy management mechanisms. A common approach consists in using a local aggregator to centrally collect the parameters of flexible actors and RES data to subsequently optimize a given objective, both in planning and real-time operations [156]. Figure 6.2 depicts such a situation, where the centralized aggregator agent handles the communication with the grid operator.

The decentralized approach taken in this chapter is depicted in Figure 6.3. Without the aggregator agent, the goal of the community is to agree on a consensus that optimizes a given shared objective function. The aggregator is partly replaced by the blockchain environment and every energy actor only interacts with the blockchain. The latter orchestrates events that will trigger the appropriate smart-contracts functions to enable cooperative energy management.

The role of every participant in the community is to dispatch its flexible assets in order to meet an aggregated objective, a role that was granted to the aggregator in the centralized scheme. The proposed mechanism works in two distinct phases. In the *day-ahead* phase, the community decides on an optimal planning of the aggregated load profile, and transfers it up to grid operator. To ensure that the actual community consumption is as close as possible to its planning, the *online* phase strives to track the *day-ahead* aggregated profile during the day.

6.3.1 Decentralized planning algorithm

The planning profile is the result of the community cost function minimization problem, solved in a decentralized fashion. At time instant h, the community can buy local electricity $y_l^-[h]$ at a cost $f_l^c(y_l^-[h])$ and electricity coming from the grid $y^+[h]$ at a cost $f_{G^+}^c(y^+[h])$. If it doesn't import electricity from the grid, it might export $y^-[h]$ to the grid for a gain $f_{G^-}^c(y^-[h])$.

The responsibility to optimize the flexible assets falls into the smart-buildings themselves, instead of a central aggregator. Each participant in the community hence computes a local optimum for their planning, given an intermediate forecast of other nodes, to then shares their updated decision with the rest of the community. Iteratively, the smart-buildings will therefore adapt their power forecast, based on forecast actions taken by the others, in order to optimize an aggregated cost function. The resulting algorithm can be seen as a GT problem [188], in which the players are incentivized to change their power consumption, given grid utility electricity prices or a common community goal.

The local optimal planning of the i^{th} smart-building is given by the input sequence $\hat{\boldsymbol{u}}^{\boldsymbol{i}}$ solving:

$$\min_{\hat{\boldsymbol{u}}^{\boldsymbol{i}}} \sum_{h=0}^{H-1} f_{G^+}^c(y^+[h]) + f_l^c(y_l^-[h]) + f_{G^-}^c(y^-[h]) \tag{6.16}$$

$$\begin{aligned}
\text{s.t. } & \textit{Eq. (6.2) and (6.8) init. state at } h=0 \\
& \underline{\forall h = 0 \ldots H-1}: \\
& e^i[h] + l_{*-i}^+[h] + l_{*-i}^-[h] = y^+[h] + y^-[h] && (6.17)\\
& e^{i,-}[h] + l_{*-i}^-[h] = y_l^-[h] && (6.18)\\
& 0 \leq y^+[h] \leq M\, s_y[h] && (6.19)\\
& 0 \geq y^-[h] \geq -M\,(1 - s_y[h]) && (6.20)\\
& \textit{Eq. (6.1) to (6.15) at time } h
\end{aligned}$$

where l_{*-i}^+ and l_{*-i}^- represent the demand and production forecast of the community <u>minus</u>

Algorithm 4 Smart-Contract 1 - Day-Ahead planning logic

```
procedure STARTPLANNINGPHASE(participant i)
    emit runPlanningAndBroadcast(i)
end procedure
procedure UPDATEPLANNING(participant i)
    if new forecast ($l_j^+$,$l_j^-$) then
        emit updateCommunityForecast(i, $l_j^+$, $l_j^-$)
        if allowedToRunPlanning then
            emit runPlanningAndBroadcast()
        end if
    end if
end procedure
```

the i^{th} smart-building, respectively, and $e^{i,-}$ is the net power export of the i^{th} building. The vector $\hat{\boldsymbol{u}}^{\boldsymbol{i}} = [\hat{\boldsymbol{u}}^{\boldsymbol{i}}_{\boldsymbol{th}}, \hat{\boldsymbol{u}}^{\boldsymbol{i}}_{\boldsymbol{d}}, \hat{\boldsymbol{u}}^{\boldsymbol{i}}_{\boldsymbol{b}}]$ regroups the control input variables for all the flexible entities of the i^{th} building. Eq. (6.17) models the power balance at the grid entry point of the community, while Eq. (6.18) defines the total local production.

Binary variables $s_y[h]$ along with constant M ($\sim 10^6$) are used in Eq. (6.19) and (6.20) to ensure that variable representing grid importing and exporting don't take simultaneously a non-null value at time period h, via the big-M method [160]. It's worth noticing that slack variables are used on the air/water temperature constraints in order to ensure feasibility of the problem but removed from Eq. (6.16) for the sake of clarity.

Practically, the decentralized algorithm sequences and information broadcasting are enabled by the smart-contract described by Algorithm 4. The functions called through the keyword "emit" are located at the building premise, while the smart-contracts run in the blockchain. A periodic event (every 24h) is used to trigger the *day-ahead* planning phase, by calling the corresponding function *startPlanningPhase* in the contract. To orchestrate the iterations of the algorithm, the function *updatePlanning* of the contract is in charge of periodically reading whether a new planning-related transaction has been written in the blockchain. In that case, the smart-buildings update their community planning knowledge and then, if allowed to do so, run their own optimization, by solving Problem (6.16). Eventually, the decentralized planning phase will naturally be over when all the participants write on the blockchain the message "no planning change". The resulting community planning is then the aggregation of the last transactions containing individual planning forecast data:

$$\hat{\boldsymbol{y}}_{\boldsymbol{pp}} = [y^+[0] - y^-[0], \dots, y^+[H-1] - y^-[H-1]] \tag{6.21}$$

This decentralized planning algorithm represents an autonomous DR scheme in the sense that all the participants tap into their flexibility to shape their expected power profile. Traditional DR programs generally refer to a baseline against which building are compared to check

Algorithm 5 Planning update of building i

```
let (variable) $l^{+}_{*-i}$, $l^{-}_{*-i}$ = [0,...,0]
procedure updateCommunityForecast(power $l^{+}$, $l^{-}$)
    $l^{+}_{*-i} \leftarrow l^{+}$, $l^{-}_{*-i} \leftarrow l^{-}$
end procedure
procedure runPlanningAndBroadcast
    $e_i \leftarrow$ solve dec. planning Eq.(6.16) given $(l^{+}_{*-i}, l^{-}_{*-i})$
    if $e$ changes or significant objective change then
        broadcast new forecast $e_i$ on the blockchain
    else
        broadcast "no planning change" on the blockchain
    end if
end procedure
```

whether they appropriately responded. Unlike traditional methods that look at the last days power consumption to create this baseline, this iterative algorithm naturally leads to the declaration of such a baseline by each building, a day in advance. Then, the second part of this autonomous DR scheme consists in rewarding/penalizing participant with respect to that baseline, as explained in the next section.

6.3.2 Online phase: tracking and monitoring

The community is incentivized to ensure that the actual community grid power imports and exports follow the planning decided a day in advance.

Community billing

Smart-Contracts defined by Algorithms 6 and 7 are deployed to monitor the real-time buildings power profiles and to individually bill the participants, respectively.

In Algorithm 6 (Smart-Contract for monitoring), an event is periodically emitted to collect individual participant power consumption/production. The monitoring is carried out by $pMonitoring()$, which gathers the whole community state before calling *communityMonitoring()*. The latter compares the power profile of the entire community to the one that was announced through the cooperative planning. If the difference exceeds a given threshold, the participants must individually be penalized, for their aggregated behavior deviates too much from the planning they all agreed to follow. However, the actual billing is not yet carried out: instead, tracking errors are stored in the vector gap and the billing is deferred until the end of the day. The function $trackingErrorEvent()$ notifies the participants that the community failed to track their planning at time instant h.

Algorithm 6 Smart Contract 2 - Online monitoring (periodic, every dt)

```
let (constant) $\epsilon_{thres}[]$
let (previously computed) $\hat{y}_{pp}[]$
let (variable) power[:][:], gap[:][:] $\leftarrow 0$
procedure pMonitor(power $l^+$, $l^-$)
    power[i][h] $\leftarrow p$
    validate participant i
    if all participants validated then
        call communityMonitoring(h)
    end if
end procedure
procedure communityMonitoring(time h)
    $\epsilon_{track} \leftarrow (\sum_j \text{power[j][h]} - \hat{y}_{pp}[h])$
    if $\epsilon_{track}$ exceeds $\epsilon_{thres}[h]$ then
        emit trackingErrorEvent(h, $\epsilon_{track}$)
        for each participant i do
            gap[i][h] $\leftarrow power[i][h] - e_{pp,i}[h]$
        end for
    end if
end procedure
```

The Algorithm (7) (Smart-Contract for billing) ensures that each participant is properly billed individually at the end of the the day as follows:

$$
\begin{aligned}
B_D^i &= c_D^+ \cdot \sum_{h=0}^{H-1} e_i^+[h] - c_D^- \cdot \sum_{h=0}^{H-1} e_i^-[h] - \sum_{h=0}^{H-1} f_l^c(e_i^-[h]) \\
c_D^+ &= \frac{\sum_{h=0}^{H-1} f_{G^+}^c(y_G^+[h]) + f_l^c(y_l^-[h])}{\sum_{j=1}^{N} \sum_{h=0}^{H-1} e_k^+[h]} \\
c_D^- &= \frac{\sum_{h=0}^{H-1} f_{G^-}^c[h]}{\sum_{j=1}^{N} \sum_{h=0}^{H-1} e_k^-[h]})
\end{aligned}
\tag{6.22}
$$

where c_D^+ and c_D^- represent the average daily community prices ($/kWh) of buying or selling energy from/to the grid, respectively. The function $electricityBilling()$ in Algorithm (7) implements such a volumetric billing.

In addition to the daily volumetric bill, the participants in the community are also individually rewarded or penalized depending on the online planning tracking quality. The Smart-Contract for billing (Algorithm 7) rewards/penalizes the individual buildings as a result of the community behavior, through the functions $pool()$ and $incentiveBilling()$. The event $pool()$ is called by the community participants, with a certain amount of blockchain currency to enable the transaction, as specified by the keyword "payable". As all the community actors have filled up their balances, the actual tracking reward/penalty mechanisms can be executed

Algorithm 7 Smart Contract 3 - Accounting (periodic, every day)

```
let (constant) requiredAmount[]
let (previously computed) gap[][], power[][]
let (variable) balance[:] ← 0
procedure ELECTRICITYBILLING
    for each building i do
        bill ← compute $B_D^i$ as Eq.(6.22) given power
        emit billEvent(i, bill)
    end for
end procedure
procedure (PAYABLE) POOL(participant i, amount a)
    require a >= requiredAmount[i]
    balance[i] += a
    add i to whitelist
    if whitelist complete then
        authorize incentiveBilling()
    end if
end procedure
procedure INCENTIVEBILLING
    for each building i do
        $p^-$, $r^+$ ← PRComputation(i, gap[i]
        balance[i] += $(r^+ - p^-)$
        balance[grid_id] -= $(r^+ - p^-)$
    end for
end procedure
```

by $incentiveBilling()$. Variables p^- and r^+ represent the corresponding penalty and reward applied to the balance of the i^{th} participant. The index $grid\_id$ stands for the grid operator to which the community provides tracking services.

Forecast tracking

As for the implementation on the online tracking of the forecast, this could be done by solving a decentralized MPC to reduce the error between the community profile and the forecast planning, iteratively executed by each flexible asset:

$$\begin{aligned} \min_{\{\hat{\boldsymbol{u}}^i\}} \; & \sum_{h=0}^{H_o-1} (y^+[h] - y^+_{pp}[h])^2 + (y^-[h] - y^-_{pp}[h])^2 \\ \text{s.t.} \; & \textit{Community model and constraints as in Eq. (6.16)} \\ & \textit{Updated environmental data forecast} \end{aligned} \tag{6.23}$$

where H_o is the receding horizon. However, the large latency inherent to blockchain won't allow such a decentralized MPC to be deployed at a large scale. Instead, each energy actor can opt to track their individual day-ahead forecast $e_{pp,i}$ through a local MPC:

$$\begin{aligned} \min_{\hat{u}^i} \quad & \sum_{h=0}^{H_o-1} (e_i[h] - e_{pp,i}[h])^2 \qquad (6.24)\\ \text{s.t.} \quad & \textit{Eq. (6.1) to (6.15) at time } h = 0, \dots, H_o - 1 \\ & \textit{Updated environmental data forecast} \end{aligned}$$

The Smart-Contracts described by Algorithms 6 and 7 pave the way to incentivize the decentralized tracking of the *day-ahead* planning. In this 2-phase framework, the blockchain capabilities are fully leveraged: whereas it is used as a pure immutable shared database in the planning phase, the blockchain is then use to transfer funds in real-time through the second and third smart-contracts. Through its innovative structure, the blockchain allows a close to real-time monitoring and penalty/reward billing, enabled by the community and its decentralized rules themselves. This represents a huge advantage over traditional centralized third parties that would take more time to apply them, because they would be relying on banking systems and administrative services, and would also lack of transparency. Nevertheless, the specific implementation, such as the details of the function $PRComputation()$ and the *online phase* practical implementation, goes beyond the scope of this chapter. An entire framework, involving tailored penalties/reward functions, would be needed for an effective cooperative online tracking. Instead, this study focuses merely on the planning phase mechanisms and the blockchain deployment.

6.3.3 Community objective

The objective function shared by all participants in Eq. (6.16) influences the behavior of the entire community, with respect to grid services and local resources use. The framework at hand allows participants to join any program, according to his/her own personal interest. This chapter analyzed two different community programs, presented below. Both of them lead to a global convex formulation of the community optimization problem.

Price-based DR for grid services

In this program, community participants try to minimize their daily own bill according to Eq. (6.22). Pricing represents a traditional means for the grid operator to influence the whole community planning phase. The cost of importing electricity from the grid is given by:

$$f^c_{G^+}(x,h) = a^q_g[h]\ x^2 + a^l_g[h]\ x + a^c_g,\ (x \geq 0)$$

where the coefficients a^q_g, a^l_g vary over time. This generic form allows to take into account multiple influencing factors. Firstly, the constant term a^c_g (\$) encompasses infrastructure cost. Secondly, the linear term a^l_g (\$$/kWh$) can either be a constant energy price, a static TOU retail price, or could follow the wholesale market prices. Lastly, the quadratic coefficient a_g^q ($\$/(kWh)^2$) accounts for second order effects [203], such as quadratic dependency of some power plants with respect to their generated power, or the losses in the lines due to long distance energy transport. Such a quadratic dependency will have the effect to flatten the overall grid demand.

The cost of buying electricity from local RES is supposed to be time-invariant:

$$f_l^c(x) = -a_l\ x,\ (x \leq 0)$$

The gain (negative cost) of exporting electricity to the grid assumes that the locally generated energy is sold to the grid at a lower price than the price to buy it locally:

$$f_{G^-}^c(x) = \alpha_s\ a_l\ x,\ (x \leq 0, 0 \leq \alpha_s < 1)$$

Green community

This program solely aims to increase the use of local resources, regardless of their cost. The cost of buying electricity from local RES is therefore set to zero:

$$f_l^c(x) = 0$$

By joining this program, participants will foster the integration of RES into the grid, such as residential PV and windmills, and ensure that their production matches as much as possible the participants consumption.

6.4 Case Study and Discussion

This section presents a realistic test case and discusses the results of the decentralized planning algorithm described in Section 6.3. The code developed for this project is in open access at the repository [61] and has been run on a single Intel Core i7-4710HQ CPU (2.50GHz × 8) with 8GB of DDR4 RAM.

6.4.1 Simulation platform

The simulation setup architecture decouples the simulation coordination from the models described in Section 6.2, each of them running as an independent process. Figure 6.4 depicts the main components of the simulation environment. A *NodeJS* server is at the heart of the platform, and performs three distinct tasks:

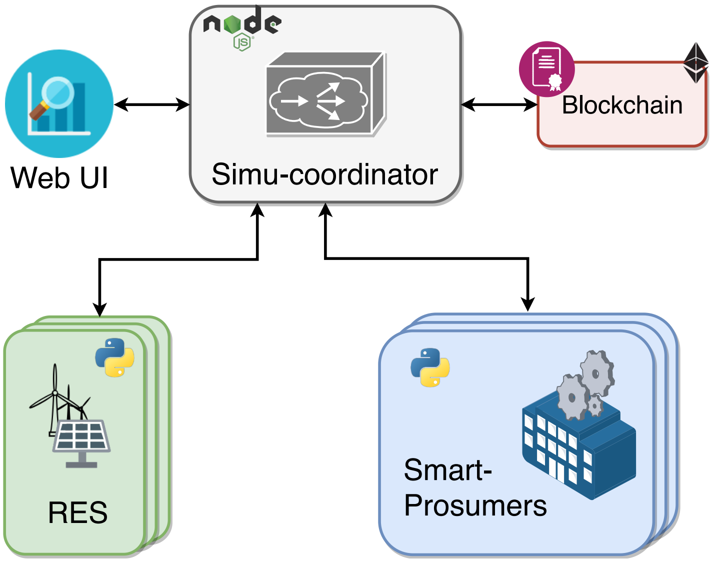


Figure 6.4 – Blockchain-based decentralized simulation setup

- Routing the simulation messages to coordinate the energy-related entities. The fast messaging protocol ZeroMQ [63] allows a bidirectional communication between the actors in the community, to enable the decentralized logic.

- Connecting the community to the Ethereum blockchain via Web3JS API [208]. Instead of connecting each process to the blockchain as it would be the case in reality, passing through the *NodeJS* server has been preferred for the maturity of the Web3JS API. Practically, the *ganache-cli* private blockchain, using EthereumJS [209], simulates a full client behavior for the purpose of these simulations.

- Binding a web interface with the simulation for display and control. Sockets created at the *NodeJS* server publishes any new information about the decentralized algorithm and actual power consumption, and the web interface updates in real-time the corresponding graphs.

As for the energy-related actors, both RES and smart-building simulated entities share the same *Python* class, in order to implement the same decentralized logic flow. However, while RES simply reads forecast files, a smart-building process is more advanced. The package CVXPY [210] models the equations presented in Section 6.2, and the linked Gurobi solver [161] is used to compute the solution of the local planning problem given by Eq. (6.16). Upon a day-ahead planning phase request, triggered by the the Smart-Contract described in Algorithm 4, every entity will periodically read and write in the blockchain by exchanging information through a PUB/SUB link with the *NodeJS* server. In real-time operation, they send their power consumption - a task performed by the smart-meter in reality - to the *NodeJS* router that both writes it in the blockchain and the web-client sockets.

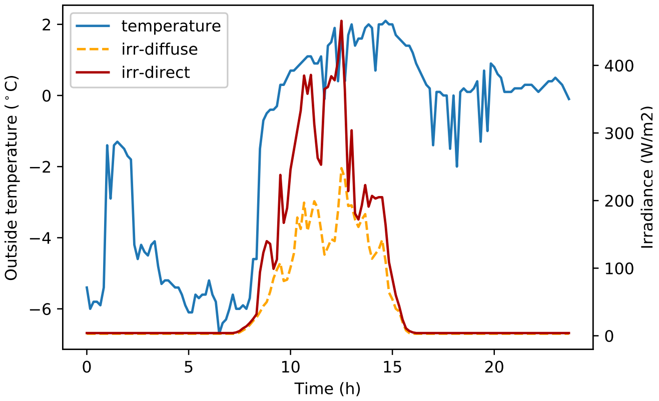


Figure 6.5 – Outside temperature and irradiance used for simulation

The Smart-Contracts presented in Section 6.3 are practically coded in Solidity [211]. The *ganache-cli* environment allows a tuning of the block size, mining time, hash rate, and other useful blockchain parameters. To each of the energy actors - prosumers and RES - is linked a unique blockchain account and a running node. An sufficient number of Ethers (the Ethereum currency) is granted to each of them, to ensure that they have the necessary amount to run the smart-contracts. The Solidity code describing the logic of the contracts is not directly used in the blockchain. It must first be compiled into more basic hash code that the Ethereum blockchain will be able to understand to automatically generate new transactions in it.

Simulation model parameters

The test case consists in a community of N_{sb} smart-buildings and N_{res} RES. The parameters of the buildings are derived from the Swiss standard Minergie [68], that specifies a set of constraints on the maximum electrical power use, thermal insulation requirements, air renewal, and other useful data facilitating the model parameters extraction task. More information about the Minergie standard and how the corresponding building parameters have been extracted can be found in Chapter 2. The EV parameters are derived from Tesla Model S, and the vehicles are supposed to be plugged-in during the day. The behind-the-meter PV system takes parameters from Solar's DIAMOND CS6X-310 manufacturer datasheet. As for the environment in which the community evolves, external temperature and sun irradiance are shown in Figure 6.5, retrieved from MeteoSwiss. The data corresponds to the weather in Pully, Vaud, Switzerland, during the month of January 2015.

To generate a fleet of similar yet different buildings, probability distributions on the parameters have been applied. More specifically, the following parameters vary from one building to another in the community:

- Zones and water tank volume - the nominal value is multiplied by a factor following a uniform distribution $\mathscr{U}(0.5, 2)$.
- Thermal mass - the thermal mass in each room follows a uniform distribution $\mathscr{U}(3, 6)$.
- Building envelope, i.e. the zones equivalent thermal resistance with the outside environment - the nominal value is multiplied by a factor following a uniform distribution $\mathscr{U}(0.6, 1.5)$.
- Temperature preferences - the lower bound nominal value is incremented by a value following the uniform distribution $\mathscr{U}(-2, 2)$, and the upper bound a uniform distribution $\mathscr{U}(-1, 3)$.
- Hot water use - the periodic usage is shifted for each building by a value following uniform distribution $\mathscr{U}(0, 6000)$, in seconds.
- Arrival/leaving time of the EV (if present) - the nominal value is shifted by a value following the normal distribution $\mathscr{N}(0, 1800)$, in seconds.
- Initial EV SoC (if present) - the nominal value is multiplied by a factor following a uniform distribution $\mathscr{U}(1, 3)$.
- PV size (if present) - the nominal value is multiplied by a factor following a uniform distribution $\mathscr{U}(1, 3)$.
- Deferrable load presence - each deferrable load is present with a probability of 2/3.

In practice, instances of the OpenEMS presented in Chapter 4 handle the prosumer simulation as a Python process. Upon instantiation as individual processes, they connect to the shared OpenBMS (see Appendix A.3.1) to retrieve the building model and its parameters. The parameters distribution is hardcoded for each building, and applied by the OpenEMS itself at instantiation through their BMS interface.

6.4.2 Scalability analysis

The decentralized planning algorithm converges once all the actors don't observe any change in their forecast planning or in their own objective function. This section explores the convergence rate for an increasing community size. Due to the decentralization logic, the order in which entities take the hand (referred to as sequence) can randomly vary depending on the computational time of each node and other factors. Therefore, for each community size N_{sb}, the planning algorithm has been launched multiple times with a random sequence, leading to statistical distribution of the convergence rate. Figure 6.6 (top) shows the number of iterations needed to complete the decentralized *day-ahead* algorithm Eq. (6.16) as a function of the community size N_{sb}, with the convex price-based objective. The trend clearly indicates a linear dependency between the number of iterations, and hence the number of transactions

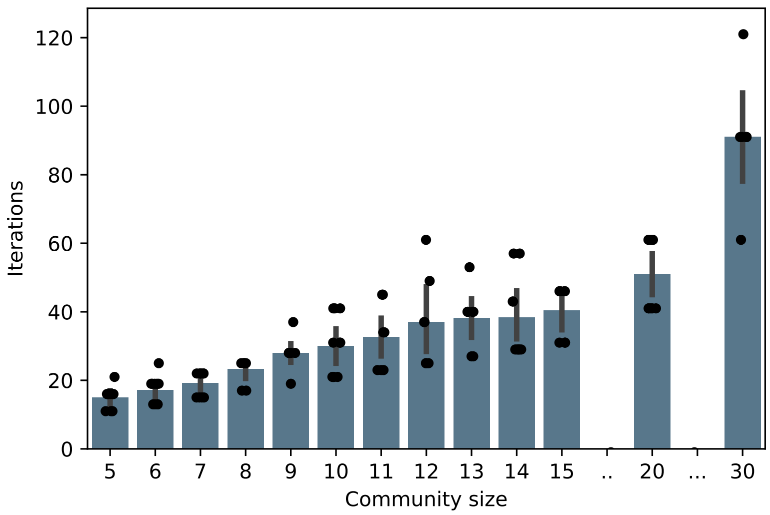


Figure 6.6 – Planning phase: algorithm iterations as the the community size grows

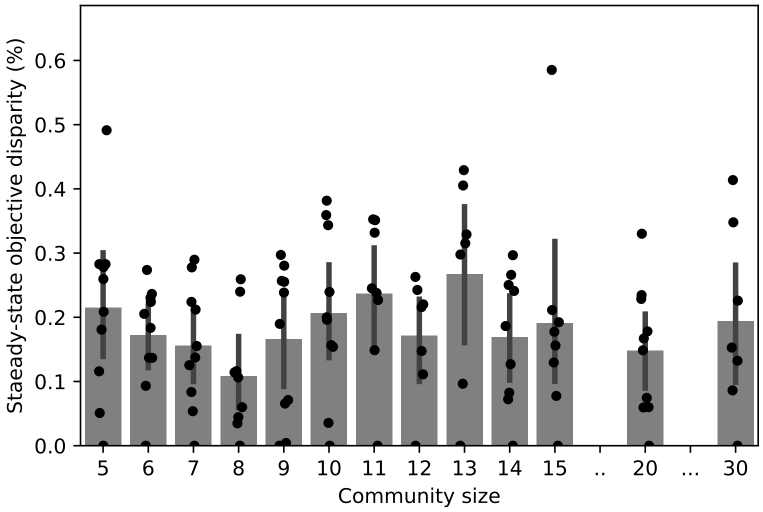


Figure 6.7 – Planning phase: steady-state cost function disparity as a function of the community size

in the blockchain, with respect to the community size. In the public Ethereum blockchain, a new block takes on average 15 seconds to be written and the system waits for 3 blocks to actually validate the transaction. On the tested machine, an iteration needs on average 15 seconds to complete the updated forecast. A community of 30 buildings could therefore take up to 2 hours to carry out the planning phase via the Ethereum blockchain. Practically, the planning phase should execute during the afternoon, in order to get reliable forecast, and finish before the end of the day for the grid to be able to use it. Considering that window of 6 hours, the proposed framework is therefore limited to a maximum of roughly 100 buildings. However, one must bear in mind that this statement is valid for the considered community cost function, the building models, and the chosen public blockchain.

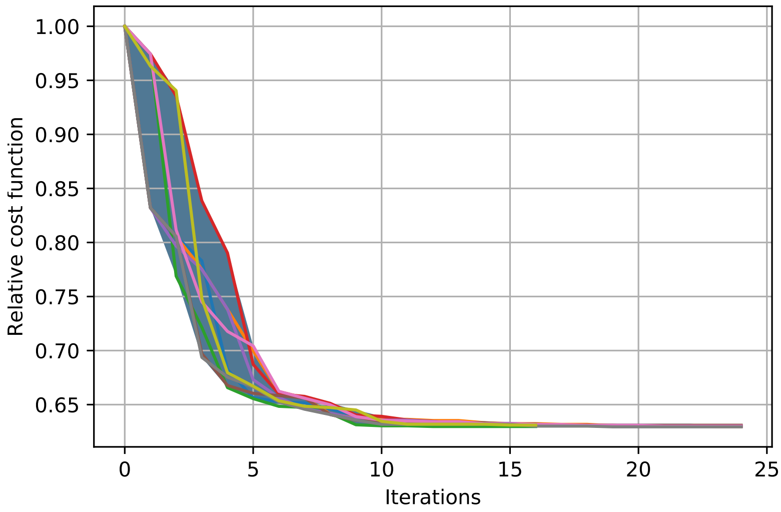


Figure 6.8 – Planning phase convergence for a community of 8 buildings. Individual curves refer to a given day-ahead algorithm sequence.

Since the sequence influences to some extend the convergence rate, it's interesting to analyze how does the order impact the steady-state result. One defines the steady-state cost function disparity $D_c(n)$ as the gap between the optimal solution of a sequence of N_{sb} buildings and the most optimal solution, after convergence:

$$D_c(n) = \lim_{k\to\infty} f_c^n(k) - \underset{x\in S_N}{min}(\lim_{k\to\infty} f_c^x(k))$$

where $f_c^n(k)$ is the community objective function at iteration k, S_N is the set of all possible permutations of $\{1...N_{sb}\}$ and n is a specific sequence in S_N. Figure 6.6 (bottom) plots this metric as a function of N_{sb}. One observes that the steady-state cost function disparity is less than 0.3% regardless the size of the community. The sequence therefore does not impact the planning cost function of the community, meaning that any participant can take the lead on the decentralized algorithm without impacting the aggregated cost.

6.4.3 Community simulation scenarios

The rest of this section considers a community of 8 Minergie smart-buildings and 1 local RES. The previous subsection concluded that the algorithm is not impacted by any particular sequence, and this is highlighted on Figure 6.8 for the small community at hand. In this specific case, the algorithm actually converges after 3 rounds (24 iterations).

This section highlights how the algorithm is impacted by the presence of local RES, the building assets, and the type of optimization objective. Two scenarios have been considered for the community:

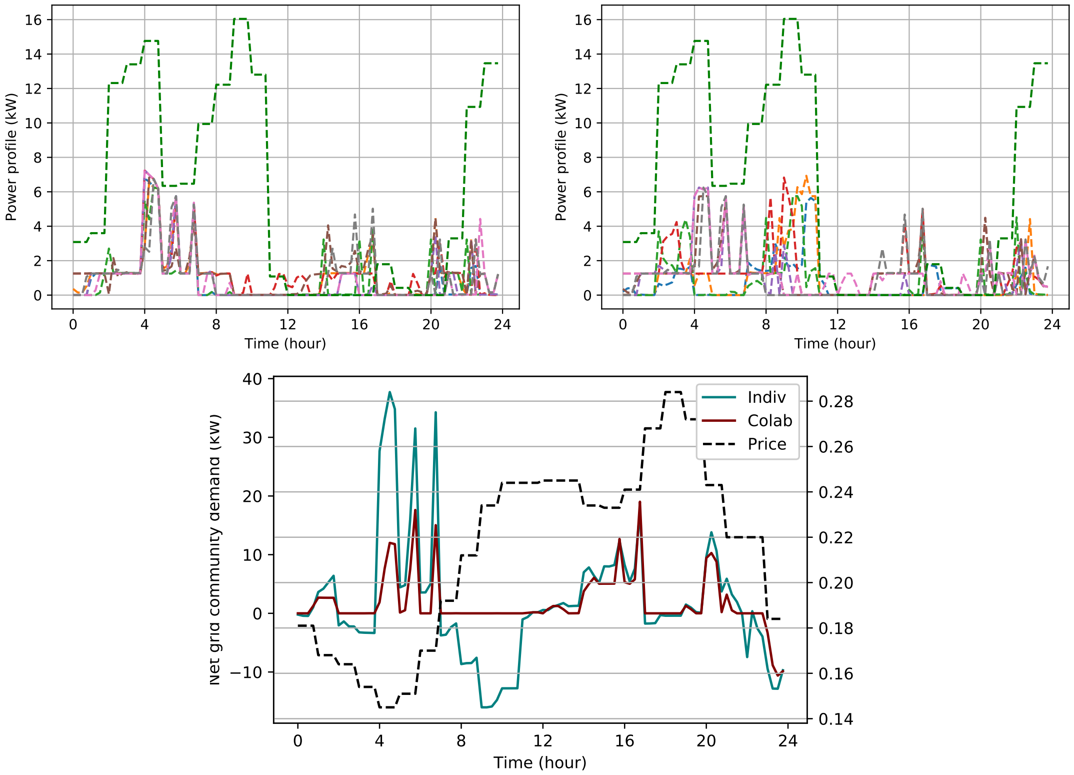


Figure 6.9 – Load profiles resulting from the Cooperative Planning Phase in the community of 8 Minergie buildings, in the presence of **Market prices**, and **with RES** (dashed green): (top left) non-cooperative load profiles (top right) cooperative load profiles (bottom) aggregated profiles. In the top row graphs, each colored dashed-line corresponds to a community participant power profile, either a RES or a smart-building profile, and their aggregation leads to the bottom graphs.

- *Green community with Market price.* The community consumes local resources in priority, and the cost of importing electricity from the grid is proportional to the Swiss Day-Ahead Auction Market prices [162] ($a_g^q = 0$).

- *Grid-services with Quadratic price.* The community is subject to both a linear price of energy proportional to the Day-Ahead Market prices and a quadratic price of energy. The quadratic coefficient a_g^q is set to 0.03125 (\$/kWh^2) throughout the day, such that the quadratic term prevails on the linear one when the community exceeds a certain demand threshold.

For each of the aforementioned scenarios, three community configurations are simulated: (1) the presence of a wind-powered RES, (2) 50% of the buildings are equipped with PV panels, and (3) 50% of the buildings are equipped with PV panels and EV (not necessarily the same buildings). Community-level metrics can then be derived from the simulation results:

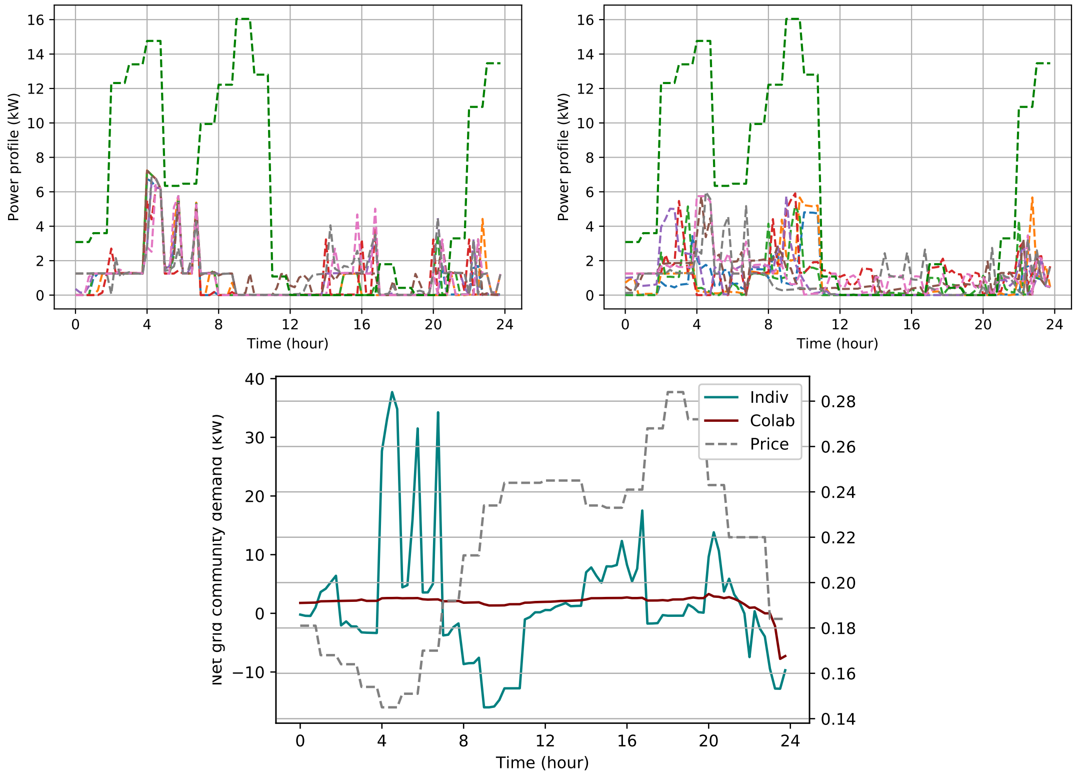


Figure 6.10 – Load profiles resulting from the Cooperative Planning Phase in the community of 8 Minergie buildings, in the presence of **Quadratic prices**, and **with RES** (dashed green): (top left) non-cooperative load profiles (top right) cooperative load profiles (bottom) aggregated profiles. In the top row graphs, each colored dashed-line corresponds to a community participant power profile, either a RES or a smart-building profile, and their aggregation leads to the bottom graphs.

- **RES consumption**: the ratio between the daily use of energy generated locally and the daily energy generated locally.
- **PAR**: the ratio between the maximum grid power demand and the mean grid power demand.
- **Community cost**: the community cost corresponding to the *day-ahead* planning consensus.

Table 6.4 and Table 6.5 regroup the simulation results considering the presented community configurations, for both scenarios *Green community (Market prices)* and *Grid-services (Quadratic price)*, respectively. The label "Indiv." stands for the non-coordinated simulation, *i.e.* the smart-buildings optimize their consumption regardless the behavior of the other; Inversely, the label "Coop." regroups the results of the cooperative planning algorithm. Figure 6.9, Figure 6.10, and Figure 6.11 help understanding the community load profiles for both

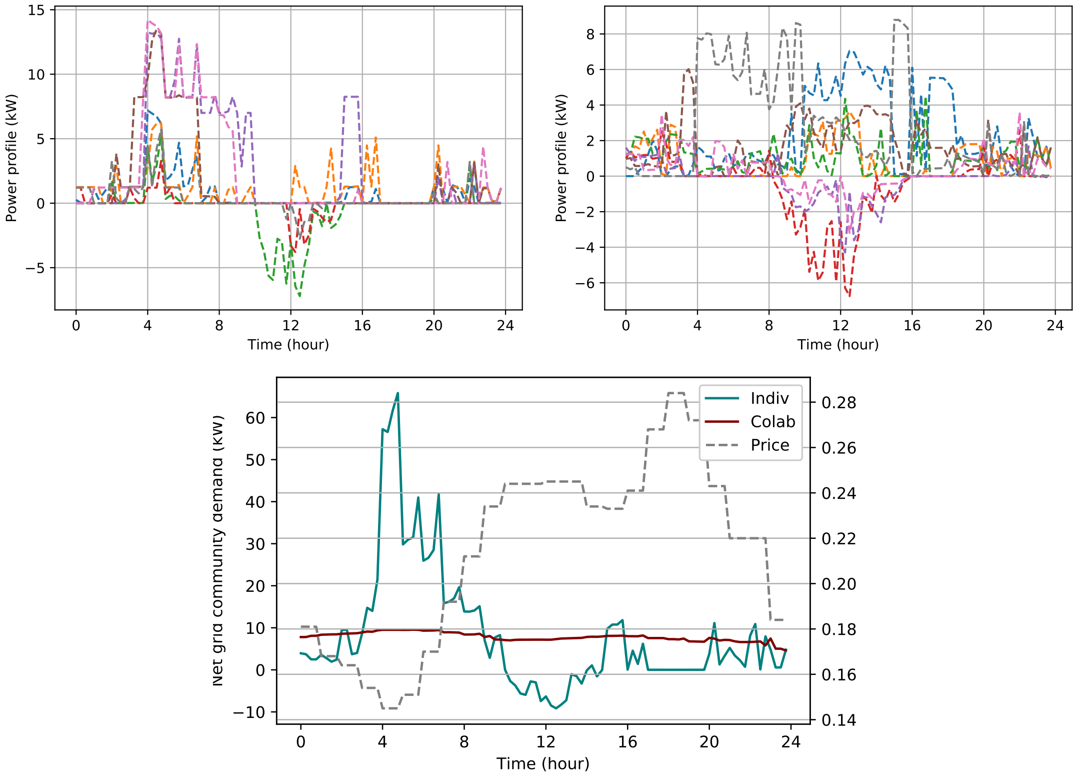


Figure 6.11 – Load profiles resulting from the Cooperative Planning Phase in the community of 8 Minergie buildings, in the presence of **Quadratic prices**, with **behind-the-meter PV**, and no RES: (top left) non-cooperative load profiles (top right) cooperative load profiles (bottom) aggregated profiles. In the top row graphs, each colored dashed-line corresponds to a community participant power profile, either a RES or a smart-building profile, and their aggregation leads to the bottom graphs.

individual and cooperative logic. It's worth noticing that the simulation doesn't account for uncontrollable loads, which would change the PAR as an uncontrollable baseline would appear. The simulations therefore represent a best case scenario, in which all the loads are perfectly controllable. The presented results thus set the highest limits of the community metrics, for a more realistic model/experiment would worsen the results.

The results of the *Green community* program (Table 6.4) indicate that the community could optimally consume the local resources, compare to a selfish algorithm. In the presence of 1 independent RES and no behind-the-meter PV (Scenario 1, Figure 6.9), almost half of the local resources were sold back to the grid. This can be seen in the bottom graph of Figure 6.9, as the global community export is negative for most of the morning. The decentralized cooperative algorithm allowed a local consumption of nearly 95%, only releasing energy back to the grid in the late evening. When half of the community has PV installed (Scenario 2, no figure), the surplus that could not be self-consumed was almost entirely sold to the grid (RES consumption < 1%). The cooperative algorithm manages to tap into the flexibility of the

Table 6.4 – Cooperative algorithm applied to *Minergie* community - Scenario *Greeen community with Market-price*. Scenarios (1) includes an independent RES, while scenarios (2) and (3) only involve buildings.

Scenarios Metrics	(1) No PV, no EV		(2) 50% PV, no EV		(3) 50% PV, 50% EV	
	Indiv.	Coop.	Indiv.	Coop.	Indiv.	Coop.
RES consumption (%)	56.29	94.25	0.008	100	14.65	100
Community cost ($)	29.07	24.16	20.35	19.74	39.05	36.1
PAR	8.61	8.79	9.15	9.6	7.16	8.1

Table 6.5 – Cooperative algorithm applied to *Minergie* community - Scenario *Grid-services with quadratic price*. Scenarios (1) includes an independent RES, while scenarios (2) and (3) only involve buildings.

Scenarios Metrics	(1) No PV, no EV		(2) 50% PV, no EV		(3) 50% PV, 50% EV	
	Indiv.	Coop.	Indiv.	Coop.	Indiv.	Coop.
RES consumption (%)	56.29	96.96	0.008	100	14.65	100
Community cost ($)	96.08	27.97	88.67	37.73	254.14	90.45
PAR	8.61	1.61	9.15	1.37	7.16	1.22

smart-buildings to entirely balance local generation with local consumption. When adding EV (Scenario 3, no figure), their profiles naturally match a bit more the local production, but most of the latter is still sold to the grid. Once again, the decentralized planning phase managed to harness all the local resources. Overall, since the price of RES (including PV) is lower than the market price, the cooperation results in a lower community cost. However, as the considered buildings are entirely made of flexible components, they all consume when the price of energy is lower, leading to a high PAR. The addition of a quadratic component solves this issue.

Adding a quadratic component to the linear pricing allows to really harness the full potential of the decentralized planning algorithm (*Grid-services* program. As each smart-building is concerned by the decision of the others, both for local resources use and grid consumption peak, the overall community profile is flattened (cf. Figure 6.10 and Figure 6.11). In Table 6.5, the PARs are therefore tremendously reduced, closer to unity. In the individual selfish scheme, the costs are now much higher due to lack of communication among the community actors, and such a tariff structure would therefore be unrealistic in a non-collaborative community.

6.4.4 Discussion

Price structure and fairness. The community framework can provide a service to the grid manager, that can locally tune the prices to shape the entire aggregated profile. To do so, the grid operator can decide on either the linear or the quadratic coefficient. The linear coefficient influences the flexible energy distribution over the day, while the quadratic coefficient dictates the PAR of the system. To a larger extend, the grid manager could handle multiple of these smart-communities and balance the demand and supply via the price of electricity across the set of communities. The billing scheme proposed in Eq.(6.22) fosters the buildings to cooperatively take part in the decentralized *day-ahead* consensus, because the community aggregated data are taken into account rather than individual actions. This means that it does not reward nor penalize who consumes/produces how much and at what time, but rather spread the actions of everyone on the resulting community cost.

Prediction uncertainty. The presented test case assumes perfect forecast knowledge and exact models for prediction, as well as ignored non-controllable loads, which is unrealistic especially at residential level subject to many uncertainties. The threshold discussed in Algorithm (6) could be a solution to this problem. By joining an uncertainty to its forecast planning, the smart-building could tell the community and hence the grid about the reliability of its prediction. The Algorithm (6) could then integrate directly this uncertainty as the threshold to reward/penalize the smart-building.

Decentralized algorithm robustness. Concerning the robustness, the presented smart-contracts would need to include more mechanisms to prevent non-responding nodes from jeopardizing the entire planning operation. Malicious participants could therefore deliberately block the algorithm. Such an issue could be addressed by a smarter way to pick the next building forecast, for example through bids, or simply by privatizing the blockchain.

Reproducibility and privacy. The framework enables the participation of heterogeneous decentralized actors, possibly gathering various building types and control techniques (e.g. MPC in commercial buildings, load scheduling in homes). This important feature circumvents the need of a central controller that must know the details of every entities it supervises. The privacy of data is therefore ensured, and it also reduces the engineering time to reproduce the framework in a different environment.

Blockchain. The type of blockchain (private, semi-private, or public) depends on the objective of the community. In the case of the sole optimization of local resources use, a public blockchain is suitable as it allows any participant to join. When providing services to the grid, the aggregator/utility is likely to deploy a semi-private blockchain to regulate the participants. Although the current consensus system for public blockchain (Proof-of-Work) is debatable when applied to energy efficiency [202], blockchain has a great potential for accelerating the decentralization of a large and complex system such as power system. The decentralized capability of blockchain allows to implement bottom-up solutions, without depending on grid operators or waiting for changes in policies. Equipped with the right technology and motivated to better consume energy, communities of people could fasten the pace of renewable integration and DR.

Deployment cost. In addition to their BDMS, smart-buildings require an energy management application to effectively shape their power profile. Optimization of individual building profile runs into this local application, generally by leveraging prices of energy emitted by the energy provider to provide local grid support [120]. In comparison to the individual optimization, the proposed cooperative framework replaces the price signal and the individual objective function by a community cost function and an exchange of power forecast information. It also replaces the utility monitoring/billing function by automated smart-contracts. The added cost therefore solely depends on the underlying network, in this case the blockchain itself. Practically, running a smart-contracts implies fees to the participants, that vary with the state of the blockchain system at the running time, as well as the type of blockchain itself as discussed in the previous paragraph. In the first incarnations of the proposed framework, such a cost might be high due to the intense energy need of the promising blockchain technology, but we envision this cost to significantly decrease as it gains popularity and maturity.

6.5 Conclusion

This chapter presented a decentralized framework to manage the electrical consumption in a community of smart-buildings and local RES. Blockchain and smart-contracts allowed the participants to collaboratively decide on a planning profile that minimizes the overall aggregated cost, through a succession of local optimization processes. This planning can greatly benefit the grid operator in its day-ahead dispatch, and the online tracking reduces the need of additional capacity reserve. Moreover, the simulation results showed that the algorithm fostered the local use of energy and, under special tariff structure, the peak grid demand could be reduced. Finally, the scalability analysis highlighted that it can be applied to a community of up to 100 smart-buildings, given the current state of Ethereum.

The proposed framework, and technology like blockchain in general, changes the paradigms in DR tailored to energy arbitrage. It provides a group of heterogeneous smart-buildings and RES with the capability to consume energy in a smarter way, without the need of a central entity that must know the details of every entity it supervises. As the future calls for an intense electrification of buildings and transportation, these decentralized assets must be efficiently incorporated in the global grid by taking into account their aggregated behavior. Beyond the grid services presented in this thesis, this blockchain-based solution allows the participants themselves to cooperatively work towards an overall decarbonized electrical grid.

7 Toward a Unified Smart-Building Model

The previous chapters have demonstrated the effectiveness of energy prices as a vector for shaping electricity consumption either at the building premises or at the level of an entire community. Yet, microgrid operators, market aggregators, and electrical utilities are also interested in directly leveraging the flexibility offered by smart-buildings instead of relying on prices that might not guarantee a required load change. Given the large heterogeneity in building types, controllable entities, and the technology used to enable smart-buildings, there is a need to use a standard representation of the flexibility of any building.

The main **highlights** and **contributions** of this chapter are:

- A review of existing models and data structures representing the building's energy flexibility. These data structures are meant to be exchanged with grid-side agents for integration into their optimization-based decision-making process. The studied models are: demand bidding curves, a flexibility contractual framework, and the Battery Equivalent Model (BEM).
- Simulations of these data structures on a large building model demonstrate their advantages and disadvantages as well as their strong dependency on environmental conditions. The advanced analysis illustrates the limitation of the BEM framework when used in MPC.
- A proposed generic Building-to-Grid (BtG) flexibility data structure mostly based on the BEM framework and a new Equivalent Kinetic Battery Model (Eq-KiBaM), as well as deferrable load models.

Table 7.1 – Nomenclature used in Chapter 7: Parameters

Symbol	Description	Unit
α	The BEM self-discharge	s^{-1}
$\underline{\boldsymbol{\beta}}/\overline{\boldsymbol{\beta}}$	The price of power ramp-down/ramp-up	\$/kW
$\boldsymbol{c}$	The price of electricity	\$/kWh
$\underline{C}, \overline{C}$	The BEM minimum and maximum capacity	kWh
ΔT	The system & control sampling period	h
$\boldsymbol{d}$	The disturbances on building's thermodynamics	/
$G, \boldsymbol{g}$	The matrix & vector containing the constraints on variables $\boldsymbol{\theta}$ and $\boldsymbol{u}$	/
H/H^m	The receding horizon length	/
H^c	The flexibility contract length	/
h_1/h_2	The "height" of the first/second well of the Eq-KiBaM	kWh
k	The "conductance" between the two wells of the Eq-KiBaM	s^{-1}
b	The capacity ratio between the two wells of the Eq-KiBaM	/
N_z	The number of zones in the building	/
η_{ch}, η_d	The BEM charging/discharging efficiency	/
$\overline{\sigma}, \underline{\sigma}$	BEM maximum charging/discharging power	kW
θ_k^r	The reference temperature of the k^{th} zone	$^\circ C$
u_k^r	The k^{th} HVAC input for reference tracking	/
P^r	The total building power consumption for reference tracking	kW
$\overline{P}, \underline{P}$	The maximum/minimum power consumption of building's HVAC	kW
x_0	The initial BEM State-of-Charge	kWh

Table 7.2 – Nomenclature used in Chapter 7: Variables

Symbol	Description	Unit
θ_k	The temperature of the k^{th} zone	$^\circ C$
u_k	The k^{th} HVAC input signal	/
P_b	The BEM power input	kW
P_{tot}	The total building power consumption	kW
$\underline{\boldsymbol{\phi}}/\overline{\boldsymbol{\phi}}$	The flexibility contractual framework's power envelopes	kW
x_b	The BEM State-of-Charge	kWh
x_1/x_2	The Eq-KiBaM State-of-Charge of the first/second well	kWh

Table 7.3 – Nomenclature used in Chapter 7 - Conventions and notations

Notation	Description
• (bold)	Vector $[\bullet_0, ..., \bullet_n]$
$\underline{\bullet}, \overline{\bullet}$	Lower and upper bounds of a variable

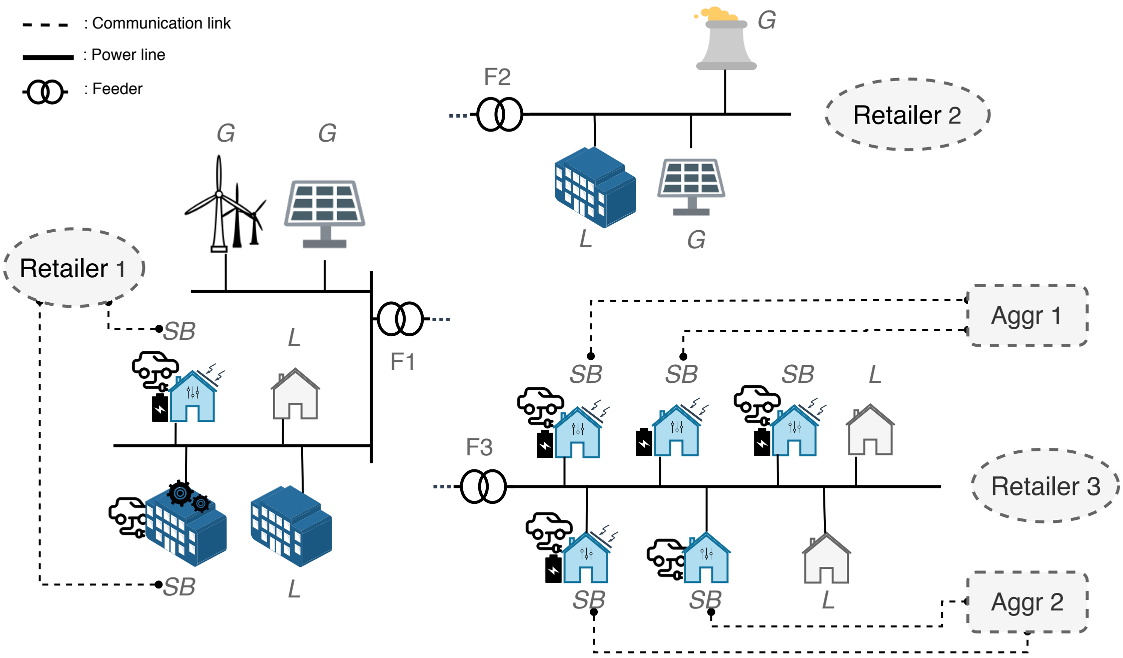


Figure 7.1 – Smart-grid actors representation.

7.1 Smart-Grid, Energy Markets, and Microgrids: the Role of Smart-Buildings

Buildings are the primary end-users of the electrical grid, continuously consuming energy injected along lines of the power system. Yet, the inclusion of behind-the-meter solar panels, secondary battery storage equipments, and their increasing ability to adapt their demand give them the potential to actively participate in the daily task of grid energy balancing and ancillary services. The future smart-building will have to interact with many different smart-grid entities to enable such a large task. Figure 7.1 schematizes the various actors present in the envisioned smart-grid. Vital elements of the system, *generators* (G), produce electricity on the power lines by using fossil fuel, nuclear resources, or renewable energy sources. The *end-loads* (buildings, electric vehicles, public infrastructure, etc.) consume this electricity, either in an inflexible way (L) or with some kind of flexibility of their load profile (SB). Distributions, transmission, and feeder lines are geographically deployed to connect these remote actors over areas that can span countries and continents.

In addition to the physical infrastructure and actors, an ecosystem of entities ensures a perpetual balance between supply and demand at various time scales (years, days, hours, minutes, and sub-seconds). Buildings (residential and commercial) do not buy their electricity directly from the generators. Instead, an energy *retailer* will sell electricity to a group of end-loads registered or assigned to it through a contract. Depending on the area, these retailers buy their electricity from a wholesale energy market or through bilateral contracts [212]. As smart-buildings offer flexibility in their consumption profile, *aggregators* are in charge of

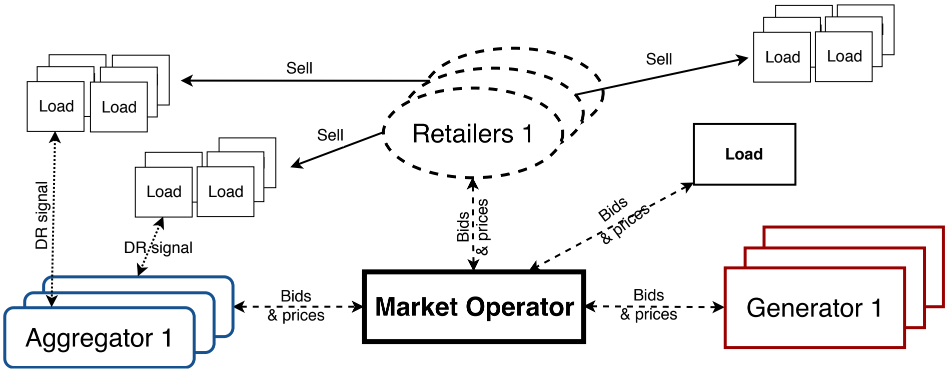


Figure 7.2 – Energy market actors

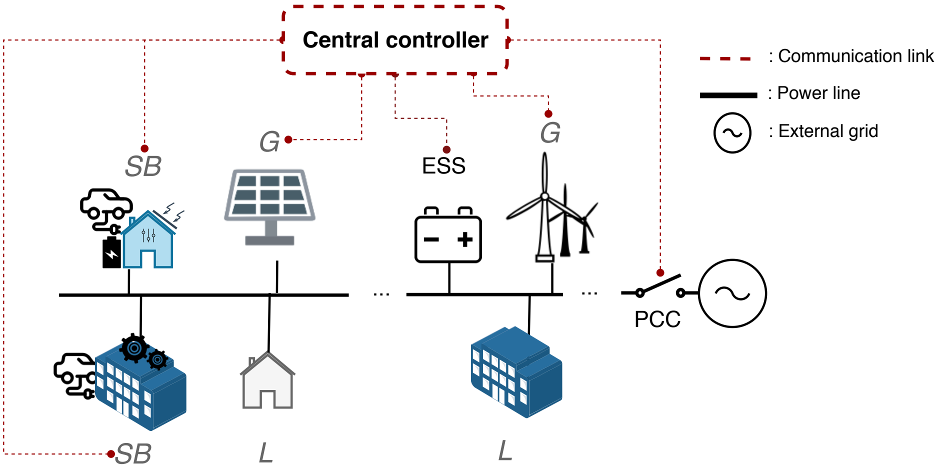


Figure 7.3 – Microgrid actors representation

pooling their resources to form a unique entity offering services to a higher-level grid-side agent [186]. Also referred to as a Virtual Power Plant (VPP), an aggregator commonly provides its services to grid utilities (or retailers) or directly bids into the wholesale energy market on behalf of a fleet of smart-buildings [213].

On the one hand, deregulated wholesale electricity markets have recently emerged as solution to match supply and demand. They work as a succession of bidding procedures that define clearing electricity prices both a day in advance in the Day-Ahead Market (DAM) and during the day in the Real-Time Market (RTM). As illustrated in Figure 7.2[1], the aggregators can send their bids to the market operator[2] by using the ability of smart-buildings to increase or decrease their consumption according to a reference. Whenever a bid is won by an aggregator, the corresponding price represents the economic value of its flexibility. For instance, the Demand Response Auction Mechanism (DRAM) was launched in 2015 in California to promote the

---

[1] This figure does not include all the actors in an energy market, for instance, *importation* from other areas are omitted.

[2] The market operator stands for the Independent System Operator (ISO) in the US

participation of DR agents into the California Independent System Operator (CAISO) markets [214]. Yet, the baseline energy of buildings is paid to local retailers who have bid in the electricity market.
On the other hand, smart-buildings can also be valuable assets in *microgrids* [215]. A microgrid refers to a set of interconnected electrical loads and Distributed Energy Resources (DER) within clearly defined electrical boundaries, seen as a single entity by the grid from which it can disconnect [216]. Microgrids are useful for efficiently optimize local resources and, therefore, represent valuable solutions to provide energy to loads in harsh environments (e.g., flood, wildfire, etc.). Either through an aggregator or directly interacting with the microgrid EMS, as depicted in Figure 7.3, the smart-building will use its flexibility to meet local objectives such as energy arbitrage or frequency regulation.

Being a hybrid entity between an end-load and a distributed flexible generator, the smart-building should declare to the aggregator its energy flexibility capabilities and related constraints as clearly and efficiently as possible. The protocol OpenADR, created in 2009, enforces the kind of information that must be exchanged between grid-side agents and load-side agents [159, 217]. However, it does not specify the model to be used by both parties and is mainly focused on traditional DR signals (cf. Chapter 4) which has limited value for advanced model-based aggregated algorithms. Furthermore, the integration of model-based optimized smart-buildings in the smart-grid changes paradigms in the way current incentive-based DR works, as the end-users become more proactive. Indeed, how can the aggregator (or a utility/ISO) measure and validate that a DR event actually really when facing an ever-changing power profile, optimally shaped based on an internal model, user preferences, and grid requirements. Gathering every single building model and their controllability details for the grid-side agent to compute a most-probable baseline would be unimaginable due to its large variety, the specificity of internal building control, and user data privacy.

The notion of Building-to-Grid (BtG) integration was introduced in [218, 219] as the set of technologies to achieve an integrated building energy efficiency and continuous demand response. They presented the technical opportunities and barriers at scale for residential and commercial buildings, highlighting the current lack of building response concerning its overall potential. Leveraging this definition, Taha et al. [220] developed a mathematical BtG integration framework in the smart-grid. By jointly optimizing the network grid optimal power flow and the energy flexibility of buildings, they showed that better allocation of resources could be achieved. Nevertheless, this method still requires a large amount of data to be shared between the grid-side agents and the buildings.

In light of the aforementioned considerations, there is a clear need to develop a unified and generic model of flexible buildings (or end-loads in general) to be leveraged by aggregators. This model should ideally federate heterogeneous groups of residential and commercial buildings based on different technologies. The idea of representing the building as a "power node" with capacity, power limits, and ramp-up/down limits was introduced in [221]. Subsequently, the concept of Virtual Battery (VB) to model a thermostatically-conditioned building was detailed in [158]. The authors showed that residential multiple Thermostatically Controlled Loads (TCLs) could be modeled as a first-order battery (referred to as Generalized Battery Model) and demonstrated that the grid operator could leverage such a model to provide ancillary services, namely frequency regulation. Huges et al. [222] extended the use of the model to commercial buildings driven by non-linear HVAC loads and referred to it as the Battery Equivalent Model (BEM). In [223], the same authors provided a method for identifying the BEM parameters of residential buildings, extending the model to a larger category of flexible loads. Considering a set of commercial and residential smart-buildings along with physical ESS, the BEM model was proven in [173] to effectively unify its representation on the aggregator side.

This chapter reviews and compares three distinct state-of-the-art BtG data structures that can benefit aggregators in handling large sets of heterogeneous flexible buildings. In addition to the BEM [173] mentioned above, the *demand bidding curves* [185] and the *flexibility contractual framework* [143] have been studied. Beyond a critical theoretical discussion, we highlight, through simulations, the advantages and limitations of these data structures; this is the focus of Section 7.2. In Section 7.3, we propose a unified advanced data structure that better captures building thermodynamics and includes other flexible entities (e.g., deferrable loads and EV). Finally, the summary and overall discussion can be found in Section 7.4.

7.2 Building-to-Grid Data Structures

Definition 7.2.1. Building-to-Grid data structure - A Building-to-Grid data structure (or BtG data structure) refers to the set of parameters, generally compact, that a building and its grid-side agent exchanges to characterize a shared model of the building energy flexibility.

Let us consider a building with N_z zones that must be thermally regulated through the use of an HVAC system:

$$\begin{cases} \boldsymbol{\theta}[t+1] & = f_z(\boldsymbol{\theta}[t], \boldsymbol{u}[t], \boldsymbol{d}[t]) \\ P[t] & = f_p(\boldsymbol{\theta}[t], \boldsymbol{u}[t], \boldsymbol{d}[t]) \\ G \begin{bmatrix} \boldsymbol{\theta}[t] \\ \boldsymbol{u}[t] \end{bmatrix} & \leq \boldsymbol{g}[t] \end{cases} \tag{7.1}$$

where f_z is a function (linear or not) dictating the next thermal state of the zones, given the current state $\boldsymbol{\theta}[t]$, the HVAC input power $\boldsymbol{u}[t]$, and the zone disturbances $\boldsymbol{d}[t]$. For simulation purposes, we used the 3-zone (commercial) building model presented in Chapter 5 and Appendix A.1.2. In this model, the HVAC system's power consumption can be continuously dimmed every 20 minutes and provides heat to the zones during a cold winter month. Furthermore, the zones are subject to heat disturbances throughout the day because of occupancy and solar radiation.

The rest of this section presents three interesting data structures that simplify and generalize

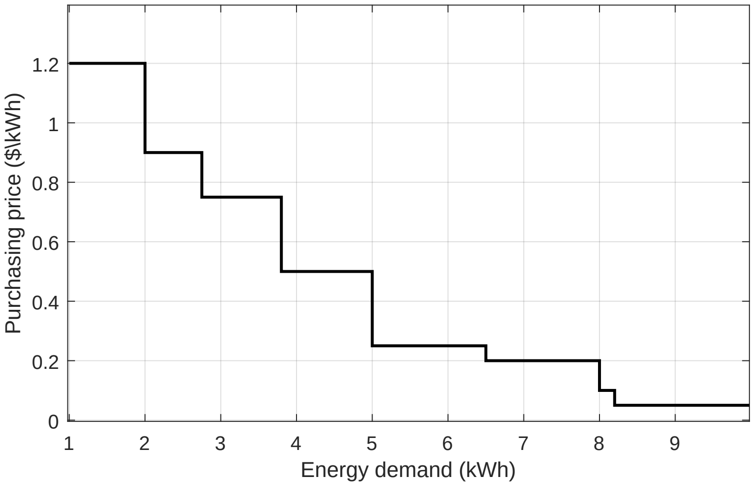


Figure 7.4 – Demand bid illustration

the model of building's flexibility. Although the frameworks to incorporate them in aggregator decision-making are also discussed, we do not quantify the cost of running them at building level. This cost, called *opportunity cost*, accounts for the impact, on the building environment, of providing ancillary services. Authors of [224] and [225] provided useful definitions and examples of these time-dependent costs on simulated commercial buildings.

7.2.1 Demand bidding curves

Generation and demand bids occur in liberalized energy markets[3], leading to the determination of the clearing price by the market operator (e.g., an ISO in the U.S.). Yet, most of the demand bids are price-takers while generation bids are the real price-makers. For instance, in the DAM of the CAISO in 2014, only 10% of demands bids (in terms of MWh) were economic bids, against 68% of economic generation bids [226]. An economic bid means that the bidder proposes various prices it is willing to pay (demand) or sell (generation) for a given amount of energy.

An economic demand bid represents an interesting data structure to model the demand flexibility of a smart-building at a given time instant. As illustrated in Figure 7.4, the smart-building is expected to produce a pair of demand-price $\{E, c_e\}$ that represents the amount of energy, E, it is willing to consume in a given time interval for the corresponding electricity price, c_e. The bidding curve takes the form of a decreasing step-wise function, as flexible buildings are likely to consume more energy when electricity prices get lower.

However, sensitivity to electricity prices is a relative metric and the building's EMS must,

---

[3] As for CAISO, economic demand bids occur only in DAM, as in RTM the demand is forecast by the market operator.

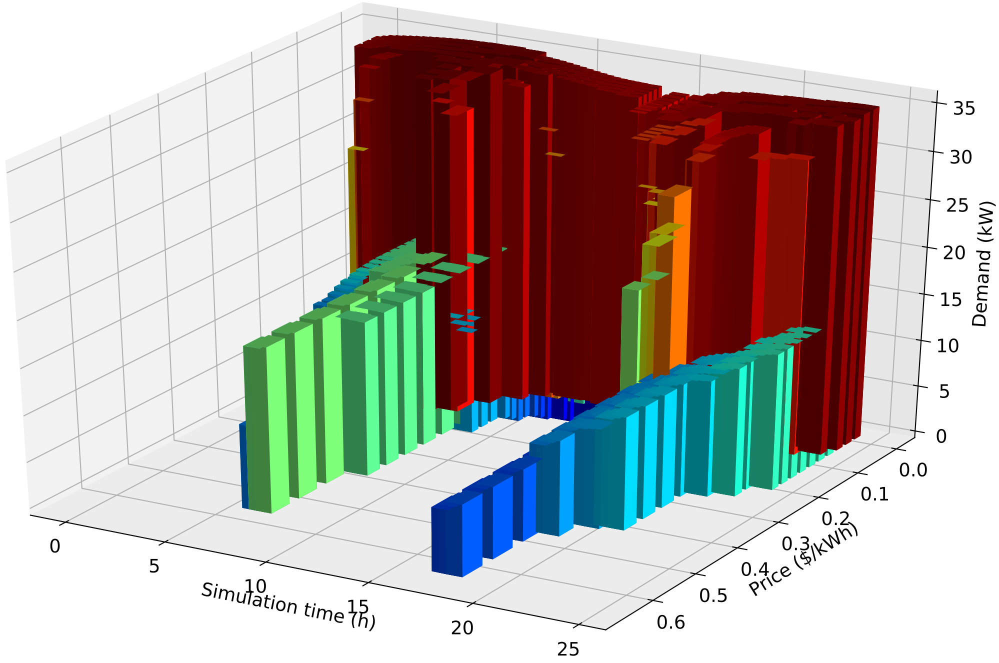


Figure 7.5 – The evolution of commercial building demand bids over one day

therefore, be aware of a reference price from which the curve can be built. Wei et al. [185] described an MPC-based method to construct bids for an aggregator. They considered that all the buildings were given a forecast price of electricity $\boldsymbol{c}$. Upon query of the aggregator at time t, the requested smart-building iteratively changes the price of electricity at the time interval starting at t while keeping the rest of the forecast signal unchanged: $\tilde{\boldsymbol{c}} = [c_k; \boldsymbol{c}[1 : end]]$, where c_k is the altered price of electricity at time t. This altered price vector $\tilde{\boldsymbol{c}}$, is subsequently used in the model-based optimization problem for computing an optimal consumption forecast. The building consumption at time interval t represents the demand it would agree to purchase at the corresponding altered price. By varying the price, c_k, in a large set of values, the smart-building progressively constructs a demand bid like the one shown in Fig. 7.4.

Figure 7.5 shows the time evolution of such demand bids based on the simulated commercial building (price and demand bid have been swapped for the sake of graph clarity). For this simulation, a TOU retail price was made up of a fixed energy price 0.2 \$/kWh ofr most of the day, except for two peak periods 4-8 am and 4-8 pm, during which the price was 1.2 \$/kWh. It's worth noting that the building HVAC systems are being controlled by a perfect MPC and, therefore, operates close to comfort limits. One observes a change in the bids during peak price periods according to energy price values as well as a correlation with the ambient environment. The afternoon exposes a highly inelastic demand as the building is already heated by the HVAC, sun radiation, and humans, and thus does not consume much power. Therefore, it would use HVAC power only at a very low energy price. In the morning (4-8 am), the bidding curves span a larger range of prices because short-term energy prices are high

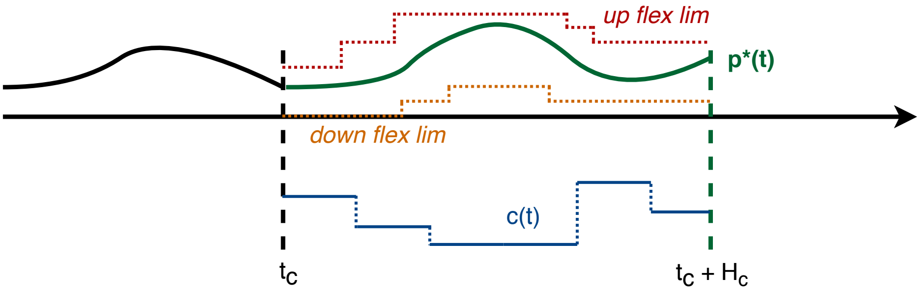


Figure 7.6 – Flexibility contract illustration

(1.2 $/kWh). An abrupt transition occurs in the demand-price bidding curves from 6-8 am as the model-based optimization foresees that the price will dramatically decrease after 8 am. Consequently, the smart-building decided to bid on power demand (around 10kW) for the price range [0.6-0.2] $/kWh (still better than the current price of 1.2$/kWh) and then suddenly raises the bidding power close to maximum capacity for prices below 0.2 $/kWh. A smoother bidding curve is observed at 4 pm when the energy price abruptly increases again. Compared to the morning curves, lower values of the power demand bids are observed at higher prices. Indeed, the building has been heated by solar radiation and internal load for the whole day and the minimum HVAC power to ensure thermal comfort is, therefore, lower; this effect fades away in the late evening, especially when the energy price decreases after 8 pm.

Leveraging the knowledge of local demand and generation bids, the aggregator logic presented in [185] consists of minimizing the customer utility and generator cost function, constrained by the physical limitations of the underlying power system (buses, transmission lines, etc.). Therefore, the optimization-based tasks of the smart-buildings and the aggregator are decoupled while being linked to each other through the demand bids. While these bids represent a way of encapsulating the building flexibility capabilities, they require knowledge of the electricity price forecast. This is realistic in the current grid state, in which retailers (i.e., IOUs) sell electricity to residential/commercial customers at a fixed price. However, to a larger extent, where buildings are directly subject to unknown real-time prices, advanced methods should be set up to forecast the value of the required price signal. Furthermore, demand bidding curves only capture a snapshot of the current building flexibility at a specific instant. The aggregator, therefore, loses the knowledge of the consequences of present decisions on the future, as it is kept at the buildings premises.

7.2.2 Flexibility contractual framework

Demand bidding data structures find their application mainly in large-period energy arbitrage, at aggregator premises. Any update requires running as many MPC simulations as there are energy prices to test, and the resulting curve applies only for the next period of time. Therefore, they are not scalable for frequent, short-term signals tailored to frequency regulation and load balancing that can occur as fast as four seconds [158].

To promote faster short-term regulation signals, Maasoumy et al. [143, 155] presented a *flexibility contractual framework* through which a smart-building specifies its power flexibility for a specified horizon, given a reward structure from its grid-side agent (e.g., utility or aggregator). Figure 7.6 illustrates the notion of a flexibility contract happening at time instant t_c. First, the grid-side agent must provide a forecast for the electricity price $c[t]$, in the contract horizon $[t_c; t_c + H_c]$, as well as ramp-up and ramp-down prices ($\$/kWh$) that correspond to the reward for increasing ($\overline{\beta}$) and decreasing ($\underline{\beta}$) the planned power consumption, respectively (not represented in Figure 7.6). Given these price signals, the building replies back with the following data structures over the entire contract horizon $[t_c; t_c + H_c]$:

- The baseline power, $P^*(t)$, it is planning to consume, corresponding to a baseline system input $u^*(t)$.
- Two signals *up-flex limits* $\overline{\Phi}(t)$ and *down-flex limits* $\underline{\Phi}(t)$ representing the maximum and minimum power consumption the building is willing to consume without violating internal comfort constraints, respectively.

Having agreed at time instant t_c upon a building baseline and flexibility, the grid-side agent will frequently send a power consumption signal, $s(t_s)$, in the contract horizon $\forall\, t_s \in [t_c; t_c + H_c]$. This consumption command will fall in the flexibility envelope and thus does not impact on comfort in the building. On its side, the building must ensure that its actual power consumption, $P(t)$, equals the agent signal, $s(t)$. Concerning the bill at the end of the day, the incentive lies in the fact that the building is charged based on the proposed baseline, $P^*(t)$, at price $c[t]$, instead of the actual consumption, $P(t)$. In addition, it receives rewards for each change of power consumption with respect to the baseline, i.e., $|P(t) - P^*(t)|$.

Mathematically speaking, the data structure transmitted by the building to its grid-side agent can be computed by solving the following optimization problem:

$$\begin{aligned}
\underset{\boldsymbol{u},\underline{\boldsymbol{\phi}},\overline{\boldsymbol{\phi}}}{\text{minimize}} \; & \sum_{h=0}^{H^m-1} C_b(\boldsymbol{u}[h], c[h]) - R(\underline{\boldsymbol{\phi}}[h], \overline{\boldsymbol{\phi}}[h], \underline{\beta}[h], \overline{\beta}[h] && (7.2)\\
\text{s.t. } \; & \boldsymbol{\theta}^u[h+1] = f_z(\boldsymbol{\theta}^u[h], \boldsymbol{u}[h] + \overline{\boldsymbol{\phi}}[h], \boldsymbol{d}[h]) && \forall h = 0 \cdots H^m - 1\\
& \boldsymbol{\theta}^d[h+1] = f_z(\boldsymbol{\theta}^d[h], \boldsymbol{u}[h] + \underline{\boldsymbol{\phi}}[h], \boldsymbol{d}[h]) && \forall h = 0 \cdots H^m - 1\\
& G \begin{bmatrix} \boldsymbol{\theta}^u[h] \\ \boldsymbol{u}[h] + \overline{\boldsymbol{\phi}}[h] \end{bmatrix} \leq \boldsymbol{g}[h] && \forall h = 0 \cdots H^m - 1\\
& G \begin{bmatrix} \boldsymbol{\theta}^d[h] \\ \boldsymbol{u}[h] + \underline{\boldsymbol{\phi}}[h] \end{bmatrix} \leq \boldsymbol{g}[h] && \forall h = 0 \cdots H^m - 1\\
& \underline{\boldsymbol{\phi}}[h] \leq 0,\; \overline{\boldsymbol{\phi}}[h] \geq 0 && \forall h = 0 \cdots H^m - 1
\end{aligned}$$

A vector notation has been chosen as multiple flexible entities may be controlled within the same building. The objective function in Eq. (7.2) is made up of two elements: a part $C_b(.)$, for charging the baseline P^*, and a reward function, R, that specifies the gain linked to a change of consumption, $\overline{\boldsymbol{\phi}}$ or $\underline{\boldsymbol{\phi}}$. The constraints involve a pair $\{\boldsymbol{\theta}^d, \boldsymbol{\theta}^u\}$ of two distinct possible building states corresponding to the system state when applying $\boldsymbol{u} + \underline{\boldsymbol{\phi}}$ and $\boldsymbol{u} + \overline{\boldsymbol{\phi}}$, respectively. When linked to temperatures, these two states represent the extreme thermal values the building is willing to set its zones for the corresponding rewards dictated by $\underline{\beta}$ and $\overline{\beta}$. An infinite value of these prices ($\underline{\beta} = \overline{\beta} = \text{inf}$) would set the states $\{\boldsymbol{\theta}^d, \boldsymbol{\theta}^u\}$ to their comfort boundaries, while null values ($\underline{\beta} = \overline{\beta} = 0$) would result in $\boldsymbol{\theta}^d$ equal to $\boldsymbol{\theta}^u$ and thus, an empty flexibility envelope.

The optimization horizon H^m in Eq. (7.2) is greater than the contract length, H^c. Therefore, only the first H^c values of the variables solving Eq. (7.2) will be transmitted to the grid-side agent. The latter incentivizes the smart-building to provide flexibility through the parameters $\underline{\beta}$ and $\overline{\beta}$ according to the need of the overall grid it connects to. At the next contract period $t_c + H^c + 1$, the building's EMS will update the building state and environment information to repeat the same transactions with its grid-side agent. Such a contractual framework has the advantage to clearly announce in advance at just one transaction instant, t_c, a baseline power profile and flexibility envelope around it. Then, the information flows solely from the grid-side agent towards the smart-building for setting commands.

Commercial buildings naturally lend themselves to this framework, as large HVAC fan power can be continuously regulated. This is why the authors of [143, 155] have considered a HVAC system of a commercial building that consumes energy from various sources:

$$P_{hvac} = P_f(u_t) + P_c(u_t, T_{out}) + P_h(u_t, T_{out}) \tag{7.3}$$

where P_f is the fan power, P_c is the cooling power, and P_h is the heating power (gas). To a larger extent, an aggregation of thermostatically controlled loads [158] could be integrated into this framework, in which the corresponding aggregator would play the role of the building site.

7.2.3 Battery equivalent model

Thermal storage in the building mass, water, and air of its zones represents a natural method of providing flexibility, as already thoroughly exploited throughout this thesis. Grey-box models such as equivalent RC circuits allow the capture of the main thermal features of the building's thermodynamics. This model already holds useful flexibility information that can be leveraged for control and energy management at the aggregator premises. Yet, a large variety of RC combinations exist, from the simple 1R1C to highly detailed zone models that represent heat transfers through doors, multiple material walls, windows, etc. In addition, the discrepancy in data usage by different buildings in connecting it to these models makes it difficult to use at an aggregator level.

The aforementioned drawbacks motivated the authors of [158, 222, 223, 173] to introduce the BEM formalism, for unifying the models of residential and commercial buildings, as well

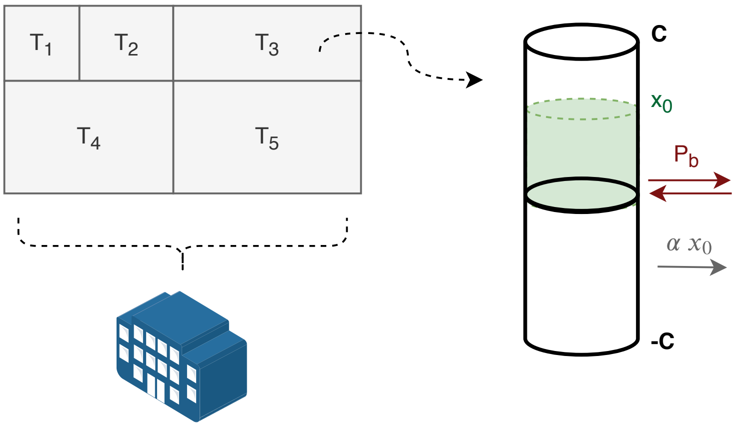


Figure 7.7 – Battery Equivalent Model illustration

as physical battery storage. They showed that the model could be applied to three families of entities (residential AC, commercial HVAC, and independent ESS). When gathered in the same distribution grid, the aggregator could leverage their unified model to provide both regulation services and energy arbitrage. The main idea of the BEM framework is to model the building's zones (physical spaces and HVAC systems) as a set of batteries, as depicted in Figure 7.7. Parameters of the well-known first-order model of the battery (cf. Chapter 1) are extracted by the building intelligence system itself (e.g., BDMS or EMS) based on user preferences of comfort and knowledge of a specific building model. This battery model describes the operational flexibility of the building with respect to a given predefined baseline. It allows the unification of flexible buildings with radically different models and technologies under the same generic model.

Mathematical framework

Let us first define $\boldsymbol{u^r}$ as the necessary HVAC input vector for tracking a given reference temperature $\boldsymbol{\theta^r}$ in a given horizon H. It can be obtained by solving the following optimization problem:

$$
\begin{aligned}
&\underset{\boldsymbol{u}}{\text{minimize}} \sum_{t=0}^{H-1} \sum_{k=0}^{N_z-1} (\theta_k[t] - \theta_k^r[t])^2 && (7.4)\\
&\quad \text{s.t. } \boldsymbol{\theta}[h+1] = f_z(\boldsymbol{\theta}[h], \boldsymbol{u}[h], \boldsymbol{d}[h]) && \forall h = 0 \cdots H-1\\
&\quad\quad G \begin{bmatrix} \boldsymbol{\theta}[h] \\ \boldsymbol{u}[h] \end{bmatrix} \leq \boldsymbol{g}[h] && \forall h = 0 \cdots H-1
\end{aligned}
$$

Solving Eq. (7.4) results in a vector, $\boldsymbol{u^r}[t]$, of HVAC input signal, such that the natural building thermal losses and disturbances are counterbalanced to keep the zones temperature as close as possible to $\boldsymbol{\theta^r}[t]$ in the horizon $t = 0 \cdots H-1$. The reference HVAC input signal corresponds to a total baseline power consumption, $P^r[t] = f_p(\boldsymbol{\theta^r}[t], \boldsymbol{u^r}[t], \boldsymbol{d}[t])$, in the horizon $t = 0 \cdots H-1$.

The linear battery model can then be introduced for each of the zones. A null SoC $x_z = 0$ means the corresponding zone, z, is at its reference state θ_z^r. The battery power at time t is defined as the gap between the actual building power consumption $P[t]$ and the reference power $P^r[t]$:

$$P_b[t] = P[t] - P^r[t] \tag{7.5}$$

Given the aforementioned definitions, the generic BEM model can be derived [173]:

$$\begin{cases} \underline{\sigma}[t] \le P_b[t] \le \overline{\sigma}[t] \\ \underline{C}[t] \le x_b[t] \le \overline{C}[t] \\ x_b[t+1] = \alpha\, x_b[t] + \begin{cases} \eta_{ch}\, P_b[t]\, \Delta t & \text{charge} \\ \frac{1}{\eta_d}\, P_b[t]\, \Delta t & \text{discharge} \end{cases} \end{cases} \tag{7.6}$$

It's worth noting that the model denoted by equations (7.6) can be applied to both the thermal zone (with respect to the pre-computed baseline) and the physical battery model:

- Physical battery: the minimum capacity $\underline{C}$ is positive or null, α is generally high (slow self-discharge) while η_{ch} and η_d are generally in the vicinity of 0.9 (losses during a round-trip cycle).

- Thermal zone: the minimum capacity $\underline{C}$ is opposite to the maximum capacity: $\underline{C} = -\overline{C} = C$. This means that the BEM' SoC can be negative. The parameter, α, is generally low (fast self-discharge) while $\eta_{ch} = \eta_d = 1$.

As an illustration, let us consider a single residential thermal zone model (See Chapter 1). The linear mapping between the thermal model linked with user comfort and the BEM parameters/variables can be expressed as follows [173]:

$$\begin{cases} \theta_z[t+1] & = f_z(\theta_z[t], u[t], \boldsymbol{d}[t]) \\ P[t] & = u[t] \\ \overline{\theta}[t] \le \theta_z[t] \le \overline{\theta}[t], & 0 \le u[t] \le \overline{P} \end{cases} \xrightarrow[where]{\text{Eq. (7.6)}} \begin{cases} x_b[t] & = C_z\, (\theta_r - \theta[t])/\mathrm{COP} \\ P_b[t] & = u[t] - u^r[t] \\ \eta_{ch} = \eta_d = 1, & \alpha = (R_z C_z)^{-1} \end{cases} \tag{7.7}$$

where C_z and R_z are the equivalent RC parameters of the zone and COP is the coefficient of performance if the HVAC. In this simple example, the mapping of a linear first-order model is direct and an analytic solution can, therefore, be found. The equivalent BEM capacity, omitted in Eq. (7.7), can easily be derived analytically. However, thermal equations driving the zones' state are generally more complex and the corresponding system may display a non-linear behavior. Therefore, such an analytical mapping cannot be identified in practice and one must rely on numerical methods.

Parameters identification

For a given zone, the corresponding equivalent battery parameters must be identified, namely, the maximum charging power $\overline{\sigma}$ and maximum discharging power $\underline{\sigma}$, the battery capacity C, and the battery self-discharge α.

The battery power is limited by the maximum and minimum electrical power capacity of the corresponding zone's HVAC:

$$\overline{\sigma}[t] = \overline{P} - P^r[t] \tag{7.8}$$

$$\underline{\sigma}[t] = \underline{P} - P^r[t] \tag{7.9}$$

However, these constraints do not consider the current state of the battery and could thus, cause constraints violation. Therefore, a more conservative approach has been preferred:

$$\overline{\sigma}[t] = min(\overline{P} - P^r[t], \frac{C - x_0}{2\,\Delta T}) \tag{7.10}$$

$$\overline{\sigma}[t] = max(\underline{P} - P^r[t], \frac{-C - x_0}{2\,\Delta T}) \tag{7.11}$$

where factor 2 has been chosen in practice to mitigate the non-linear effects of BEM mapping and hence, prevents constraint violation as much as possible.

The BEM capacity C and self-discharge α are tightly coupled and must therefore be determined together. The idea, introduced in [223], consists of using a set of constant inputs $\{u_i\}$ to compute the corresponding set of *BEM violation times* $\{t^b_{v,i}\}$, such that $x_b(t = t^b_{v,i}) = C$ when u_i is added to the baseline reference. Since the BEM is a first-order model, any BEM violation time $t^b_{v,i}$ can analytically be derived given the (positive) input u_i:

$$\begin{cases} \dot{x}(t) & = -\alpha x(t) + u_i \\ x(t=0) & = x_0 \\ x(t = t_{v,i}) & = C \end{cases} \quad \rightarrow t^b_{v,i} = -\frac{1}{\alpha} \ln \frac{C - \frac{u_i}{\alpha}}{x_0 - \frac{u_i}{\alpha}} \tag{7.12}$$

Details can be found in Appendix A.5, especially for negative inputs u_i. Alternatively, the building zones model given by Eq. (7.1) might not be linear or complex to compute analytically. Therefore, one needs to simulate the system to compute the set of *system violation times* $\{\hat{t}_{v,i}\}$, defined as the duration needed to violate comfort constraints when applying u_i to the baseline power. Figure 7.8 shows the simulated temperature evolution until violation constraints, given various input signals, for the simulated commercial building presented at the beginning of this section.

Having the simulated system violation times, $\{\hat{t}_{v,i}\}$, and the analytical form of BEM violation times Eq. (7.12), one wants to identify the parameters C and α that ensure a best fit of both violation time sets. This can be done through least-squares optimization:

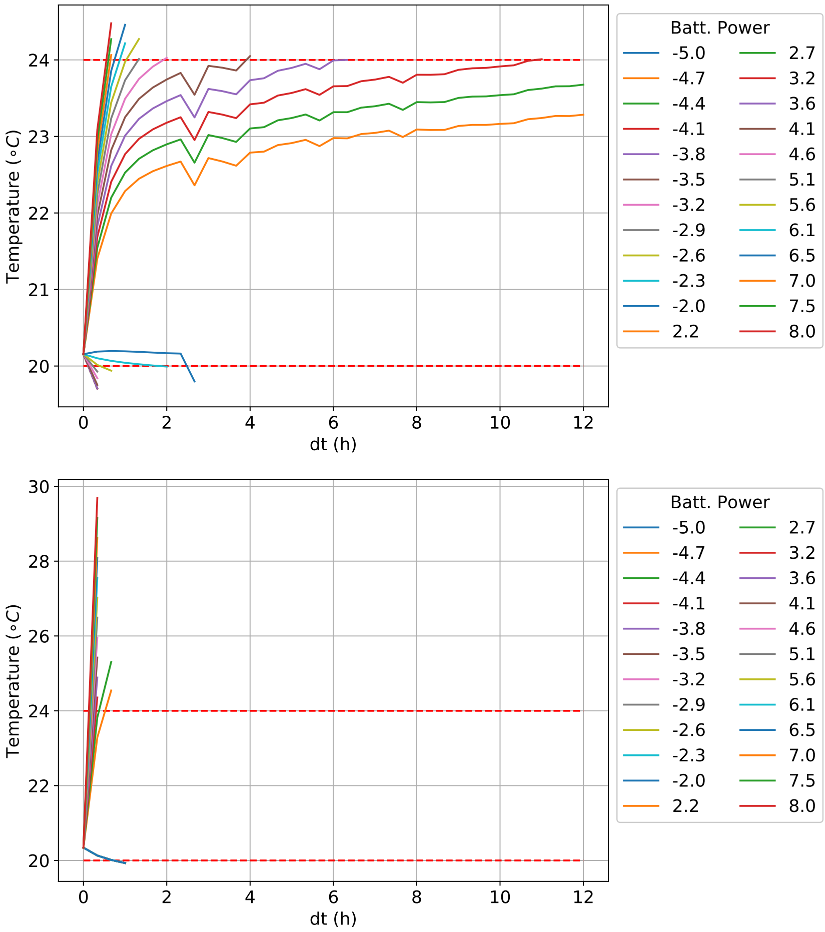


Figure 7.8 – BEM excitation signals at t=12h: effect on the room temperature (top) Room 1 (bottom) Room 2

$$\underset{x_0,\alpha,C}{\text{minimize}} \sum_{i=0}^{n^+} (\hat{t}_{v,i} + \frac{1}{\alpha} \ln \frac{C - \frac{u_i}{\alpha}}{x_0 - \frac{u_i}{\alpha}})^2 + \sum_{i=0}^{n^-} (\hat{t}_{v,i} + \frac{1}{\alpha} \ln \frac{-C - \frac{u_i}{\alpha}}{x_0 - \frac{u_i}{\alpha}})^2 \tag{7.13}$$

$$\text{s.t. } t^b_{v,i} \leq \hat{t}_{v,i} \tag{7.14}$$

$$C > 0,\ 0 < \alpha \leq 1,\ -C \leq x_0 \leq C \tag{7.15}$$

where n^+ and n^- are the number of samples in which u_i is positive and negative, respectively. The objective function denoted by Eq. (7.13) strives to ensure a parameter fitting that optimizes a mean-square error between the system violation times and the BEM ones. By adding constraint (7.14), a conservative approach is taken, leading to parameter values that will never violate the temperature constraints of the actual system.

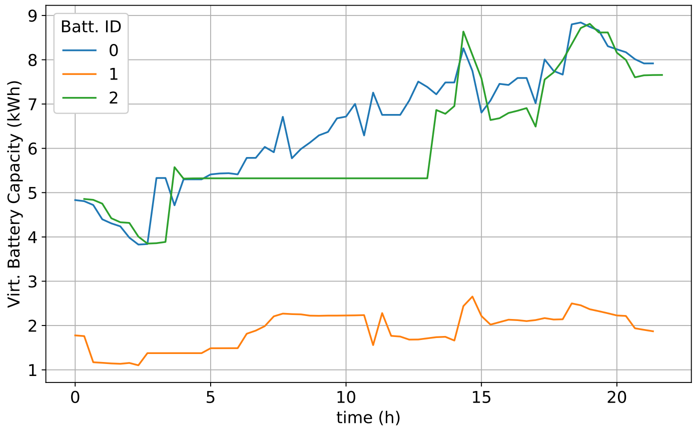


Figure 7.9 – BEM capacity

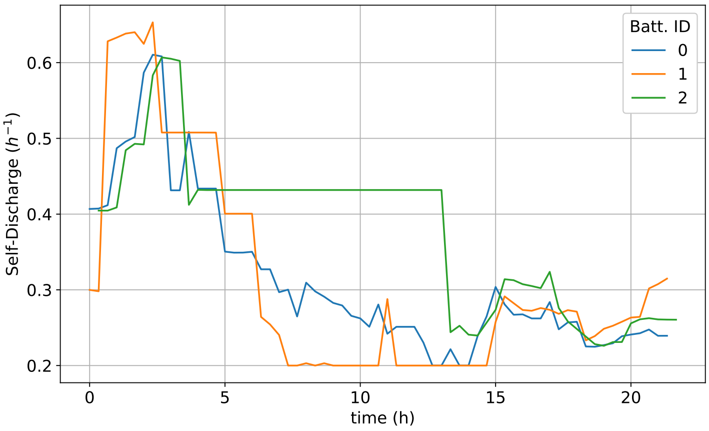


Figure 7.10 – BEM self-discharge

BEM-based MPC simulations

The BEM model finds its application when centrally gathered by an aggregator (e.g., utility or private aggregator), as demonstrated in [173] for a large set of residential and commercial buildings, along with physical batteries. Periodically, the aggregator requires an update of the buildings baseline $P^r[t]$, for a given horizon, $t = t_s \cdots t_e$, and the BEM parameters, $\{x_0, \alpha, C, \overline{\sigma}, \underline{\sigma}\}$, corresponding to the baseline in the same horizon. The building's EMS will, therefore, be in charge of computing these quantities periodically, following the method described in the above sections, and will transmit them to the aggregator. Subsequently, the aggregator will send the power signal command $\Delta P_j[t]$, back to the building j at time t, after optimizing a shared aggregated objective such as the one described in Chapter 6. In this section, we intend to analyze the effect of using a BEM-based MPC algorithm to optimize the power consumption of a single building. As the aggregator would base its decision on BEM

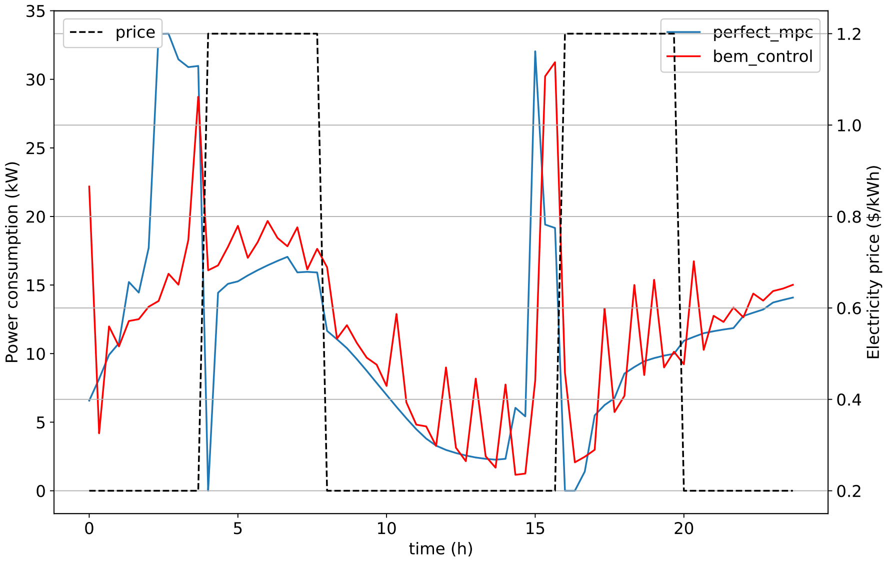


Figure 7.11 – Building total power consumption under BEM-based MPC (red) and perfect MPC (blue)

parameters, it is important to assess how the use of this simplified model impacts the building power profile and user comfort.

In this simulation, the building baseline, P^r, is the result of a PI control applied to the simulated system with the knowledge of external disturbances. This baseline represents the necessary HVAC power consumption to track a mean zone temperature of $22°C$, given the boundaries $[20,24]°C$. Then, the determination of the BEM parameters linked to each of the three zones is conducted by applying a constant increment to the reference P^r, for each of them [4]. Figure 7.8 shows the effects of these excitation signals on the zones' predicted temperature until their comfort boundaries are reached, for Zone 1 & 2 (Zone 3 shows similar behavior as room 1) at noon. The asymmetry between positive and negative input power is worth noting, as the zone losses are always working against the positive power increment. Additionally, non-linear behaviors can be observed, due to the disturbances impacting on the model over time. The temperature evolution, thus, does not follow exactly a first-order model and, therefore, the BEM simplification will approach the real dynamic with a best-effort first-order model.

Using the set of simulated violation times with BEM constant charging/discharging power in Eq. (7.13), one can derive the capacity C, self-discharge α, and initial state x_0 at each control interval t. Practically, the Python $scipy.optimize$ library was used to identify the parameters. To help the solver converging, the initial state $x_0[t]$ was assessed at each control period as

---

[4] It's worth noticing that the BEM model does not work with strongly coupled zones. In the case of the present simulated model, the zones almost don't exchange heat with one another.

follows:

$$\tilde{x}_0[t] = (1-\alpha\,\Delta t)x_0[t-1] + \Delta t\, P_b[t-1] \tag{7.16}$$

Along with the use of $\tilde{x}_0[t]$ as the initial state, the capacity and self-discharge coefficient of the BEM at time $t-1$ were used to warm-start the solver. Figures 7.9 and 7.10 show the evolution of virtual BEM capacity and self-discharge over time, as the system is being controlled. The parameters are not constant over time. On the one hand, the capacity seems to increase throughout the day, starting from 4–5 kWh to end between 7 and 9 kWh for Zone 1 and 3; Zone 2 experiences the same trend but with a capacity three times lower than the other two zones. On the other hand, the self-discharge coefficient decreases over time, to stay between 0.2 and 0.3 h^{-1} at the end of the day. Such a time-dependent trend does not occur in a low-mass system that can be approximated by a single node RC system, but does occur in the advanced model at hand that stores heat in both the air and the mass of the zones.

The BEM model can now be used in an MPC formulation in order to minimize the building owner's bill. For each of the zones, leveraging Eq. (7.6) naturally leads to the following constrained optimization problem:

$$\begin{aligned}\underset{\{P_b\}}{\text{minimize}} \; & \sum_{h=0}^{H-1} c[h]\; P_b[h] \\ \text{s.t.}\; & x[0] = x_0 \\ & x_b[h+1] = (1-\alpha\,\Delta t)\; x_b[h] + \Delta t\, P_b[h] \\ & -C \le x_b[h] \le C \\ & \underline{\sigma} \le P_b[h] \le \overline{\sigma}\end{aligned} \tag{7.17}$$

where $c[h]$ denotes the price of electricity at time h. Solving Eq. (7.17) outputs the optimal battery setpoint, $P_b[0]$, to be added to the HVAC power reference, $P^r[t]$, of the considered zone, at time instant t. At the next time instant $t+\Delta t$, new BEM parameters will be acquired and used again in Problem (7.17), and so on. However, as the BEM model is less accurate than the actual one and the solver might result in incorrect values of the parameters, a traditional ON/OFF control is applied any time the constraints are violated. A daily simulation produces the data shown in Figure 7.11 and Figure 7.12, where the red curves refer to the BEM-based MPC and the blue curves refer to an MPC energy management using the perfect model (i.e., the control model is the simulated model).

Figure 7.11 displays the total building power consumption. One observes that the BEM-based MPC tends to behave like the perfect MPC but often departs from it for a few reasons. First, the numerous spikes compared to the perfect MPC are caused by constraint violations, in which case $P_b = 0$ and hence, the reference power consumption is applied. Indeed, the BEM-based controller strives to steer the SoC towards its lower bound $-C$, but the imprecision of the model triggers a comfort violation. Second, the BEM-based solution does not pre-heat the building as optimally as does the perfect MPC before an increase in electricity price. This is because the BEM model represents a first-order system and gathers all the heat storage

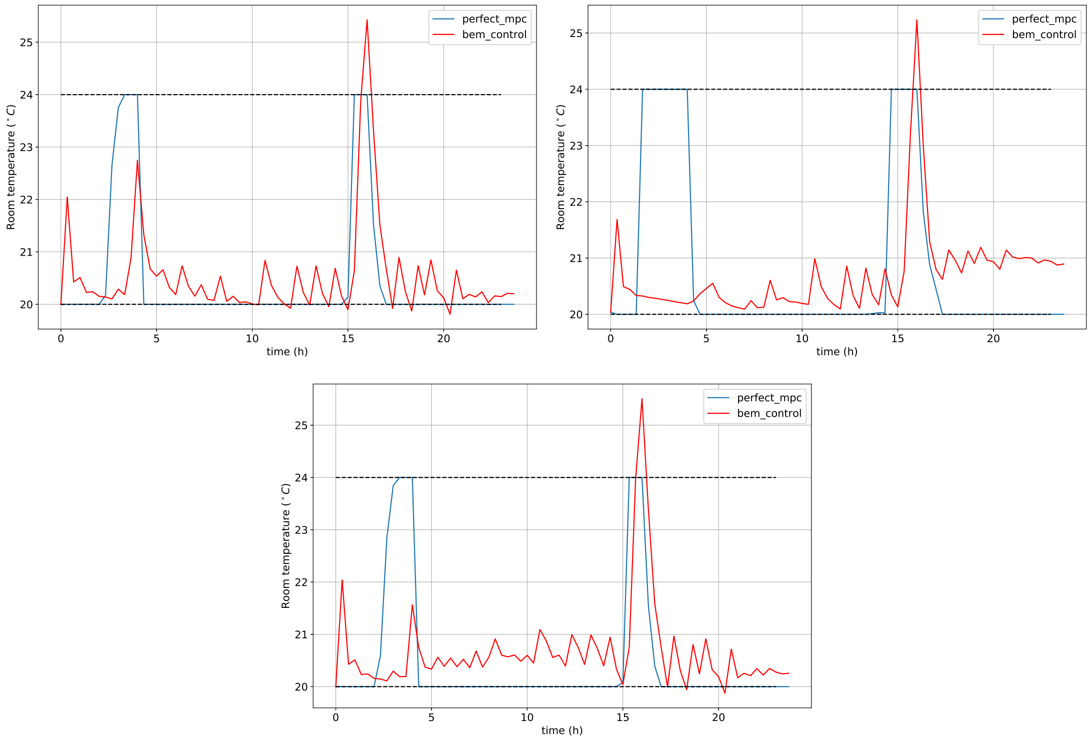


Figure 7.12 – Zones temperature evolution under BEM-based MPC (red) and perfect MPC (blue); (top-left) Zone 1 (top-right) Zone 2 (bottom) Zone 3

capacity in one single node. However, the simulated building model considers, for each zone, an air node and a mass node, whose temperatures are closely interdependent. Therefore, the perfect MPC foresees that it's worth pre-heating the mass by activating the HVAC for a longer time. The single-node battery model is unable to predict this behavior.

As for the temperature of the zones (see Figure 7.12), the comfort violation mainly occurs in Zone 1. Interestingly, the BEM-based controller did not judge economically in fully pre-heating the building before the first morning peak in energy price. At the second peak in the late afternoon, an opposite behavior occurs: pre-heating is too intense, causing an upper-bound comfort violation. This observation comes from the poor parameter identification process that either underestimate or overestimate the capacity and self-discharge of the equivalent battery and from the intrinsic approximation.

7.3 A Generic Building-to-Grid Model Proposition

In this section, we proposed a generic BtG model that could be used by an aggregator for either balancing a microgrid, bidding in an energy market, or applying common DR for the utility services. It is essentially based on the same idea as the BEM framework presented in the previous section and further integrates more flexible entities. In addition to HVAC and EWH

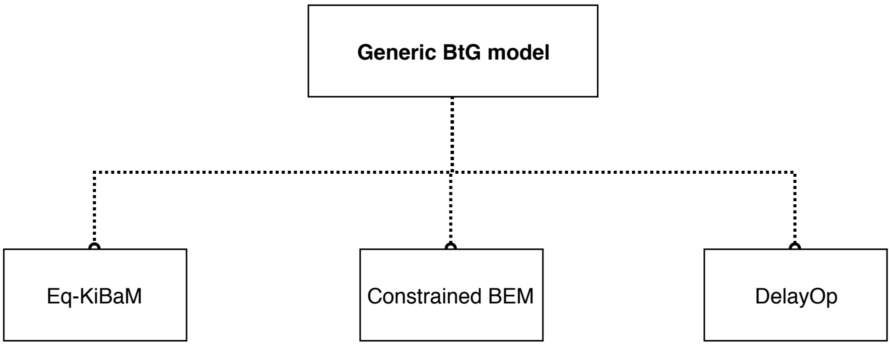


Figure 7.13 – A generic building-to-grid data structure proposition

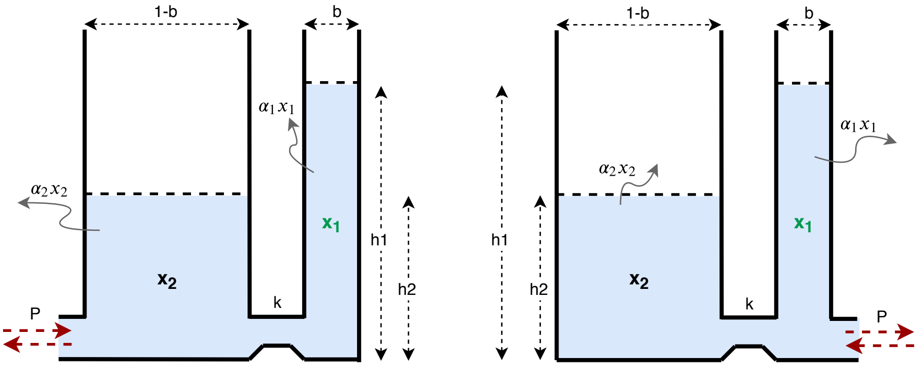


Figure 7.14 – Generic KiBaM model for BtG application (left) Zone conditioned through the air (right) Zone conditioned through the mass/water

encompassed by the BEM model, the proposed generic building model also includes electric vehicles and deferrable loads. Figure 7.13 depicts such a model made of three distinct sets of data structures:

- A set of *Equivalent Kinetic Battery Models* (Eq-KiBaM) for HVAC-conditioned zones with thermal constraints.
- A set of *Constrained BEM* for electric vehicles whose batteries SoC are constrained at leaving time.
- A set of *Delayed Operation* (DelayOp) for deferrable loads whose starting time can be flexibly shifted.

Equivalent kinetic battery model

Simulations presented in Section 7.2.3 highlighted the limitation of the BEM model when applied to a model-based controller. Indeed, the first-order equation governing the BEM's SoC is unable to capture the storage effect in the building mass that can later be used to heat/cool the air in the zones. The KiBaM formulation, presented in Chapter 1, lends itself ideally to model this intermediary effect. Indeed, it intends to represent the *recovery effect* happening in lead-acid and other chemical batteries, an effect that prevents a battery from using all the stored charges when plugged into a load.

When applied to buildings' HVAC systems, the same principle as the simple BEM holds: the equivalent KiBaM-based battery (Eq-KiBaM) is used to model the flexibility of a conditioned zone to deviate from its baseline. Unlike the BEM framework, two configurations exist for the *Eq-KiBaM* as shown in Figure 7.14. For both configurations, variable x_1 and x_2 account for air-related charges and mass-related charges, respectively. Therefore, the capacity linked to x_1 is determined by user preferences and the capacity linked to x_2 depends on physical constraints. On the one hand, the zone can be heated/cooled through its mass that will subsequently transfer to air that impacts the occupants. This is represented in Figure 7.14 (left) where the HVAC system exchanges heat with x_2. Heated floor/radiator/ water pipes linked to HPs fall into this category. On the other hand, the zone can be heated/cooled directly through the air that will subsequently transfer to its surrounding mass. This is represented in Figure 7.14 (right) where the HVAC system exchanges heat with x_1. Common air-based AC, electrical heaters, and HPs fall into this category.

Considering a zone conditioned through its air, the *Eq-KiBaM* equations can be written as follows:

$$\begin{cases} \frac{dx_1(t)}{dt} &= -\alpha_1 x_1(t) + k(h_2(t) - h_1(t)) + P(t) \\ \frac{dx_2(t)}{dt} &= -\alpha_2 x_2(t) - k(h_2(t) - h_1(t)) \end{cases} \tag{7.18}$$

where α_1 and α_2 (s^{-1}) are the self-discharge coefficient of the first and second well, respectively, and h_1 and h_2 (s^{-1}) are the heights of the first and second well, respectively. The "conductance" parameter, k, models the charge transfer flow between the two wells.

Given that $x_1 = bh_1$ and $x_2 = (1-b)h_2$, Eq. (7.18) can be re-written as:

$$\begin{cases} \frac{dx_1(t)}{dt} &= -\alpha_1 x_1(t) + k(\frac{x_2(t)}{1-b} - \frac{x_1(t)}{b}) + P(t) \\ \frac{dx_2(t)}{dt} &= -\alpha_2 x_2(t) - k(\frac{x_2(t)}{1-b} - \frac{x_1(t)}{b}) \end{cases} \tag{7.19}$$

Adopting a matrix-form notation of Eq. (7.19) leads to the following formulation of the *Eq-KiBaM* framework:

$$\begin{cases} \underline{\sigma}(t) \le P_b(t) \le \overline{\sigma}(t) \\ b\underline{C} \le x_1(t) \le b\overline{C} \\ (1-b)\underline{C} \le x_2(t) \le (1-b)\overline{C} \\ \frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} -\alpha_1 - \frac{k}{b} & \frac{k}{(1-b)} \\ \frac{k}{b} & -\alpha_2 - \frac{k}{(1-b)} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} P(t) \end{cases} \tag{7.20}$$

The procedure for parameter identification in the system denoted by Eq. (7.20) is identical to the one in the BEM system ^{5}, although slightly more complex as the equations are coupled. These parameters are α_1, α_2, $x_{0,1}$, $x_{0,2}$, k, b, and C, in addition to maximum charging power $\overline{\sigma}$ and maximum discharging power $\underline{\sigma}$. Therefore, a total of nine parameters must be gathered for each zone of the building with the *Eq-KiBaM* framework, against five for the BEM.

Constrained battery equivalent model

The presented BEM formulation also fits for modeling the dynamics of the battery of an electric vehicle. However, an EV's battery is meant to be moving with the vehicle, hence depriving the building's EMS of the flexible entity during absence periods $\mathscr{T}_{out}$. In addition, the EV's battery cannot generally be used for providing energy back to the building's loads, which translates into a positive or null battery power P_b. The battery is also expected to be charged to a minimum acceptable state, C_e, from a given instant t_e. The corresponding *constrained* BEM model varies from the BEM by adding the following constraints on the optimization variables that the aggregator will use:

$$\begin{cases} \text{Eq. (7.6)} \\ P_b \geq 0 \\ x_b[t_e] = C_{end} \\ P_b[t_o] = 0 \; \forall \, t_o \in \mathscr{T}_{out} \end{cases} \tag{7.21}$$

Beyond its application to EVs, the data structure represented by Eq. (7.21) can also include a hot water tank linked to an electrical heater for domestic hot water use. For each forecast water demand period, the corresponding BEM's SoC should be at its highest to cope with the occupant's unknown demand for hot water.

Delayed operation

Deferrable loads, presented in Chapter 1, can hardly be modeled through continuous equations such as those of the BEM. This is because of the temporal link between the predefined power consumption of the load profile and the discrete nature of the decision variable on the starting time of the load. The group of loads that have the ability to be shifted in time thus, fall into the model of *Delayed Operation*:

$$p_d[h] = \begin{cases} \mathscr{P}_d[h - t_d^*] & \text{if } t_d^* \leq h \leq t_d^* + |\mathscr{P}_d| \\ 0 & \text{otherwise} \end{cases} \tag{1.22}$$

Once started, the deferrable load cannot be interrupted and is then considered an uncontrollable load at the next update of the building's flexibility capacity model.

Table 7.4 – Summary of existing and proposed building-to-grid data structures

Name	Entities	Applications	Advantages & Limitations
Demand bidding curves [185]	All	- Local RES production matching - Aggregator bidding in market	✓ Independent of flexible entity (black-box) ✓ Structure compatible with existing energy markets × Need for forecast price signals × No receding horizon: only applies at current time
Flexibility contract [143]	Large HVAC, EWH	- Short-term ramp-up/down (freq. reg.) - Short-term load balancing	✓ Few MPC computations ✓ Ready to be used by utilities/aggregators × Need for forecast price signals × High sensitivity to disturbances
Battery equivalent model [173]	HVAC	- Local RES production matching - Medium-term (24h) energy arbitrage - Short-term ramp-up/down (freq. reg.) - Current DR programs	✓ Compact structure (5 parameters) ✓ Simple and well-known equations × Too simple for advanced building control (MPC) × Need one set of parameters per zone
Proposed Generic BtG model	HVAC, EWH, EVs, shiftable loads	*Same as BEM's*	✓ Works with all existing flexible entities ✓ Better precision for building-level MPC ✓ Easy to integrate in grid-side optimization framework × Large amount of parameters

7.4 Conclusion

Building-to-grid (BtG) models are essential structures for large scale integration of flexible demand into the future smart-grid. They can be applied in a wide range of areas, from a locally

---

[5] It is worth noting that the *Eq-KiBaM* boils down to the BEM framework if $b = 1$ or $k = 0$.

isolated microgrid to a larger energy market where aggregators bid on their behalf, as well as to electricity retailers. In this chapter, we have identified three relevant BtG data structures of radically different natures. Table 7.4 gives an overview of the reviewed BtG models, their grid-level applications, and a list of advantages/limitations for each of them. In addition to a theoretical discussion, simulations on a common commercial model highlighted the features and limitations of these frameworks. Then, we proposed an extension of the BEM model based on KiBaM, dubbed *Eq-KiBaM*, to capture the effect of thermal mass storage in a building's zones. A *generic BtG model* was envisioned to gather a set of *Eq-KiBaM* parameters, constrained BEM parameters for EVs and EWH modeling, and *DelayOp* model parameters to represent deferrable loads.

Despite their great results on simulated environments, one must bear in mind that the studied frameworks and the proposed framework all depend on the knowledge of a building model; they build their data structures and parameters entirely upon this model. As for the demand bidding curves and flexibility contracts, the model must be compatible with the MPC solver in which it is included. As an exception, the BEM and the *Eq-KiBaM* may allow a more complex model that will be mapped to the linear model of these frameworks. Relying on a model and environmental data forecast might lead to a variation between the building's forecast consumption/flexibility and its actual one. Furthermore, these building-to-grid data structures also rely on the building's EMS and BDMS to practically control its entities, which might be another source of error. As a consequence, this could translate into a discrepancy between the grid-side agent's command on power consumption and what the building will really consume.

The frameworks presented in this section completely change the way DR programs will be designed. Instead of relying on inaccurate methods solely based on historical consumption data to build a possible baseline, the buildings themselves can compute this baseline. As the baseline forecast is done at the building premises, it holds much more information and allows for advanced models to be used. Then, the presented frameworks offer a means for the grid-level agents to shape this baseline according to their needs. A generic building model, such as the one presented in this section, is envisioned to be a dynamic set of data structures that changes over time, depending on the environment in which the building evolves and especially the humans occupying it.

Conclusion and Outlook

Smart-buildings, as cyber-physical structured entities, have the potential to support the smart-grid in balancing daily energy production and demand. However, current incarnations of grid-connected smart-buildings are still rare and generally do not extend beyond local projects, specific to their particular test site. In this thesis, we provided a set of tools, frameworks, and models to foster the wider adoption of smart-buildings as active assets in the power systems.

In Chapter 2, we presented an integrated building simulator, named Virtualization Engine (*vEngine*), that plugs in to existing Building Data Management Systems (BDMSs). As BDMSs become increasingly present in buildings, an integrated simulator, deployed close to the heart of the smart-building, represents a valuable tool for performing realistic user-centric simulations, computing costs of retrofitting, and assessing the grid-flexibility potential, directly at the smart-building site. The engine is readily available to the research community, enabling users to add more entities in modular and structured ways. The multi-state load modeling presented in Chapter 3 was proven to perform better than state-of-the-art binary modeling, when applied to the energy disaggregation of a limited set of loads. As part of a building's consumption will still remain non-controllable (driven by the user), accurate energy disaggregation methods are primordial in helping a building's occupants understanding its energy usage.

Chapters 4 and 5 described how the smart-buildings could practically respond to traditional energy price changes and other Demand Response (DR) signals. In Chapter 4, we presented our Open Energy Management System (OpenEMS): an open-source platform compatible with BDMSs that enables the deployment of energy management strategies in smart-buildings. The use of this tool can save time which can then be re-invested in the development of robust advanced energy management methods for both simulations and real-life implementation. Chapter 5 focused on MPC when applied to a commercial building and highlighted the importance of dealing appropriately with peak demand charges, often neglected in the literature. We introduced an innovative incremental MPC method which better balances energy and demand costs in its objective function. Although subject to the same pricing structure, we illustrated that slightly different energy management strategies on similar buildings led to different grid behaviors. Utilities and policies must, therefore, bear in mind this heterogeneity when designing electricity prices in the presence of smart-buildings.

Chapters 6 and 7 dealt with the practical management of multiple smart-buildings in proactive communities and at the level of aggregators. In Chapter 6, we presented an innovative blockchain-based framework which enabled the decentralized management of a community of smart-buildings in the presence of Renewable Energy Sources (RES). Such a decentralized framework is envisioned to promote and accelerate the inclusion of smart-buildings into a smart-grid in a bottom-up fashion, unlike top-down policies which might take more time to appear in a near future. Yet, the blockchain technology still needs to mature further to become a large scale viable solution.
Finally, Chapter 7 presented a BtG data structure that encompassed most of the flexible entities encountered in buildings, with the goal of suggesting a unified model to be used by aggregators.

Grid-Oriented Smart-Buildings: Future Outlook

Data-driven control: the future of MPC?

Throughout this thesis, model-based energy management was intensively used either for optimization of a single building's assets or for a whole community. Nonetheless, MPC as applied to buildings has not yet been adopted broadly and is, so far, mainly limited to test field demonstrators. This comes from the complexity of deploying such a system and the corresponding cost for both the physical components and the engineering time. Moreover, one should ideally use a robust MPC and/or stochastic MPC to cope with the inherent disturbances affecting the system [227], which have not been tackled in this thesis. Additionally, as it intrinsically relies on a given model and a forecast of ambient conditions, the optimal path identified by the MPC is prone to drift away from the real building behavior and, therefore, will require engineering tweaks specific to the site under control.

Data-driven methods have recently been tested in the field of smart-buildings. Beyond their use for MPC parameter identification [19], Machine Learning (ML) methods have the potential to replace the need of a deterministic model in predictive control as they learn autonomously the relationship between building's inputs and its state. More robust to noise and less computationally heavy than MPC, a ML-based predictive control, such as Reinforcement Learning (RL), can lead to faster on-line optimization for smart-building [228]. Yet, these methods are still in the early stages of smart-buildings and will need to prove their added value to be more readily accepted. Tools like OpenEMS presented in Chapter 4 will be useful for the proof-of-concepts of data-driven algorithms.

The need of normalized building models

To carry out simulations for this thesis, we used various building models, which we mostly made ourselves, and different datasets for parameter identification. Even though they represented realistic buildings and were useful for energy management algorithms comparison purposes, reproducing the identical building models and environments represents a challenging time-consuming task. Most grid-oriented building models found in the literature follow the same trend of using a customized model. Despite the existence of public models like the Modelica Building Library, the MATLAB Thermal Model, or the ANSI/ASHRAE/IES Standard 90.1, that specify the building envelope and the HVAC system parameters, too many liberties can be taken in practice concerning the user behavior model, the ambient conditions, simulation timesteps, control parameters, etc.

Therefore, the research community studying energy in buildings needs a normalized building model, similar to the IEEE 30-bus for power system simulation, compatible across all simulation platforms. The BOPTEST project [229], financed by DOE, aims to fill this gap. It intends to create a framework which allows for comparing advanced control strategies across a standard set of buildings and climates. This would decouple the controller from the simulator, and provide the developer with a smart-building model, thus seen as a black-box, interfaced through an API (cloud-based or local instances).

Humans at the heart of the smart-buildings

Research in grid-connected smart-buildings and DR generally focuses on thermal models and pushes the modeling of human behavior into the background. Yet, the occupants of buildings (especially residential) influence the building state in a complex way. In this thesis, we have followed the traditional approach which considers a pre-defined schedule for user-driven loads, occupants' actions on their environment, and comfort preferences. It would be worth using existing, more advanced models, better capturing the relationship between the humans and their surrounding environment [230]. The distributed and modular nature of the Virtualization Engine described in Chapter 2 makes it a suitable candidate with which to use ML for advanced modeling of humans behavior.

To a larger extent, the smart-building will be a reality only with the active involvement of its occupants. Smart-meters, progressively being rolled out worldwide, represent the first step towards the BtG interface and will allow building owners to better monitor their energy consumption. However, the investment into sensor/actuator hardware and data management systems, the backbones of smart-buildings, depends solely on the initiative of the building's owner/occupant(s). Incentives will therefore be needed to promote large-scale investment in smart-building equipment and, most importantly, synchronize them with emerging smart-grid standards and data structures.

Energy policies: the key enabler for smart-buildings development

The lack of proper grid policies is often referred to as the major barrier to Demand Side Management (DSM) and DR [11, 157]. Closely linked with the aforementioned human barrier, energy policies are envisioned to incentivize building owners to invest in new technology for smart-buildings. However, policies are currently falling behind and do not reflect the need in DR of the future smart-grid. They should be updated to better redefine retail electricity price structures, and to promote the development of microgrids in remote places or harsh environments, local markets, and RES-oriented DR programs. The price of electricity remains the most widespread vector in shaping a smart-building's power profile, as demonstrated in Chapters 4, 5, and 6. However, same building types may have different controllers and logics (cf. Chapter 5). The intense electrification to meet world decarbonization objectives will increase the share of controllable loads, and therefore lead to a higher price responsiveness. It is, therefore, of utmost importance that price structures are properly designed, to account for different building behavior and environments. Dynamic prices which reflect local supply should be favored against static time-of-use contracts, either via traditional utilities or local aggregators.

Nevertheless, future smart-buildings and microgrids may raise social issues if the energy policies are not properly designed. Indeed, smart-community and microgrid technologies will reduce the dependency on the global grid, hence cutting part of the revenue. However, they are generally only available to the rich segment of the population; it would, therefore, leave the other poorer segment of the population, with an old grid relying on fossil fuel that would increase prices to cope with the reduced demand.

Concluding remarks

This PhD dissertation has demonstrated through a set of developed tools, models, and frameworks, that the smart-building represents a valuable asset in supporting the smart-grid in its daily task to balance supply and demand. Properly managed, flexible entities available in existing buildings can circumvent the need for investing in additional generation and storage facilities in the grid. The BtG integration will be ensured through efficient models and data structure that will bridge both domains. As the smart-building technology becomes ready, energy policies will have to both follow and foresee the rollout of a distributed renewable energy supply and flexible demand. Additionally, bottom-up decentralized frameworks can already optimize the use of local energy resources.

A Appendix

A.1 Building Models

Most of the models used in this thesis are derived from either a physical phenomenon or a equivalent RC model. In both cases, the resulting model is continuous and expresses the derivative of the state. The discretization process of a continuous A^c, B_u^c, B_d^c, C^c LTI model consists in identifying the following matrices A, B_u, B_d, C, given a discretization step T_s:

$$\begin{cases} \dot{\boldsymbol{x}}(t) & = A^c\, \boldsymbol{x}(t) + B_u^c\, \boldsymbol{u}(t) + B_d^c\, \boldsymbol{d}(t) \\ \boldsymbol{T}(t) & = C^c\, \boldsymbol{x}(t) \end{cases} \Rightarrow \begin{cases} \boldsymbol{x}[t+1] & = A\, \boldsymbol{x}[t] + B_u\, \boldsymbol{u}[t] + B_d\, \boldsymbol{d}[t] \\ \boldsymbol{T}[t] & = C\, \boldsymbol{x}[t] \end{cases}$$

Exact identification of discrete matrices can be done by looking at the exact solution of the continuous system:

$$x(t) = e^{A^c t} x(0) + \int_0^t e^{A^c (t-\tau)} B_c u(\tau) d\tau$$

In practice, the Python *control* toolbox allows to get the discrete LTI system through the Zero-Order Hold (ZOH) method.

A.1.1 Minergie residential model

Building description and geometry

Table A.1 summarizes the values of the parameters used in Section 2.3 for the building thermal model (cf Fig. 2.9)

The Minergie building model created in this study is artificially made of 3 distinct zones (Fig. A.1). It intends to represent an average Swiss independent villa, hosting a family. The equivalent zones regroup rooms sharing similar perturbation effect (solar radiation, load and human heat gain) and comfort preference:

Table A.1 – Minergie building - RC equivalent parameters

Parameter	Value	Parameter	Value	Parameter	Value
R_{z121} [K/W]	2.500e-2	C_{z1} [J/K]	2.724e5	R_{z1f} [K/W]	2.800e-3
R_{z122} [K/W]	2.500e-2	C_{z2} [J/K]	3.891e5	R_{z2f} [K/W]	1.900e-3
R_{z1a1} [K/W]	1.640e-2	C_{w12} [J/K]	4.500e5	R_{fr} [K/W]	1.400e-3
R_{z1a2} [K/W]	1.640e-2	C_{w1a} [J/K]	1.646e6	R_{rs} [K/W]	1.720e-3
R_{z2a1} [K/W]	1.370e-2	C_{w2a} [J/K]	1.896e6	C_{wr} [J/K]	2.092e6
R_{z2a2} [K/W]	1.370e-2	C_f [J/K]	6.822e6	C_{ws} [J/K]	3.139e5

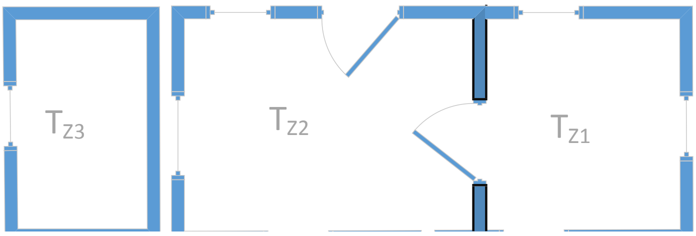


Figure A.1 – 3-zone segmentation model

- Zone 1 - Bedrooms, where the temperature comfort is more tight when occupied and the human effect is lower on average.
- Zone 2 - Common living part (living room, hall, kitchen) where lots of perturbations appear and the temperature comfort are similar but more relaxed than in zone 1.
- Zone 3 - The user seldom stays there and the comfort is not required (garage, etc).

For sake of simplicity, Zone 1 (bedrooms) is at the same level as Zone 2, whereas commonly it should be upstairs. While this hypothesis does not model the multi-floor thermal interaction, this simplifies the floor-heating system design common for both zones. These zones are not physically present as depicted in Fig. A.1, but instead represent the aggregation of the rooms that share the same properties (user occupation, comfort properties, etc).

We decided on the following dimension for the whole building: total length of 20 meters, total width of 12 meters, and height of 2.5 meters. The chosen areas for each zone and the amount of windows are displayed in Table A.2. Windows all have an area of $1.5m^2$, and an internal wall of an equivalent length of 24 meters separates zone 1 and zone 2.

Walls and windows properties

The standard Minergy [68] specifies U-values ($W/m^2.K$) requirements for walls and windows:

Table A.2 – Minergie building zones area

Parameter	Zone 1	Zone 2	Zone 3	Total
floor area (m^2)	84	120	36	240
windows	6	4	2	12

Table A.3 – Minergie building wall properties

Wall type	Th. capacity ($J/K.kg$)	Th. conductivity ($W/m^2.K$)	Density (kg/m^3)
Exterior concrete	2200	1.95	1040
Interior	1000	0.33	50

- External wall: $u_{wall} = 0.16\ W/m^2.K$
- External window: $u_{window} = 0.9\ W/m^2.K$

However these values solely model the thermal conductivity of the material and don't take into account the air renewal. The equivalent U-value of an interface separating two zones must therefore add the heat losses due to air renewal to the one due to material conductivity. This can be expressed as [1]:

$$U_{eq} = \sum_i A\ u_i + \frac{1}{3} NV$$

where A is the wall surface, u_i represent the U-value of each component of the interface, N is the air renewal rate [h^{-1}], and V is the zone volume. N is fixed to 0.6 h^{-1} for a good air quality.

The thermal capacity of an external wall making up the building envelope depends on the wall material and its geometry. Layers of concrete, brick, and glass wool are used to form a total thickness 0.2 meter and a equivalent U-value of 0.25 $W/m^2.K$. Concerning the interface between the two internal zones (zone 1 and zone 2), glass wool has been chosen, with a thickness of 0.15 meter. Thermal properties are shown in Table A.3.

Hydronic heating system

The hydronic heating system encompasses the underground water pipes circuit and the floor that releases the heat from the water to the zones occupied by humans. The water pipes contain water that is heated by a AWHP on a specific portion of the circuit (referred to as the supply circuit) and slowly transfer their heat to the floor tiles in the building. The total mass of water in contact with the floor through the pipes (referred to as the return circuit) depends on the pipe diameter and length [231].

---

[1] The coefficient $\frac{1}{3}$ is derived from the air density, air heat capacity and hours to seconds conversion

Table A.4 – Minergie building floor properties

Floor type	Th. capacity ($J/K.kg$)	Th. conductivity ($W/m^2.K$)	Density (kg/m^3)
Ceramic tiles	880	0.7	3800

We assume a supply temperature of $50°C$, a return temperature of $40°C$. The length of the pipes in each zone will therefore dictate the amount of power released in the building. From the MoPEC 2014 [232], it can be found that a maximum of $25\frac{W}{m^2}$ is allowed for heating. Therefore, from the table provided in [231], the following water pipe length is derived for both zones:

- Zone 1: a maximum of $25 \times 0.35 \times 240W = 2100W$ can be provided, therefore a maximum length of 100m of pipes, leading to a water flow of 200 l/hr. That means a total volume of $100 \times \pi \times (\frac{0.02}{2})^2 = 0.0314m^3 = 31.42l$. The resistance to the floor $R_{z1f} = \frac{0.018}{100\times\pi\times d} = 0.0028\frac{K}{W}$.
- Zone 2: a maximum of $25 \times 0.5 \times 240W = 3000W$ can be provided, therefore a maximum length of 150m of pipes, leading to a water flow of 300 l/hr. That means a total volume of $150 \times \pi \times (\frac{0.02}{2})^2 = 0.0471m^3 = 47.12l$. The resistance to the floor $R_{wf1} = \frac{0.018}{150\times\pi\times d} = 0.0019\frac{K}{W}$.

Combining both heating pipe circuits, a total of about $80l$ of water is flowing through the return water circuit. One can fairly imagine a supply water volume of $20l$, hence a total water circuit of 100 liters[2].

The equivalent thermal resistance modeling the water exchange between supply and return depends on water flow and heat capacity:

$$\frac{1}{R_{th}} = \dot{m} \times c_p = 500\frac{l}{h} \times 4184\frac{J}{K\,kg} = 581.11 \rightarrow R_{th} = 0.00172\frac{K}{W}$$

The heat stored in the water pipes must then be transferred to a material with a large thermal capacity and good conductivity. Stone/Ceramic and slate tiles are some of the best flooring for underfloor heating, due to their high thermal mass and good conductivity [233]. Heat coming from the water pipes can therefore quickly transfer to the surface. The thermal properties of ceramic tiles are shown in Table A.4.

Given that the floor thickness is fixed to 0.075 meter:

- Zone 1 ($A = 84m^2$): $R_{fz} = 0.0034\frac{W}{K}$ and $C_{floor} = 2.8 \times 10^6\frac{J}{K}$
- Zone 2 ($A = 120m^2$): $R_{fz} = 0.0024\frac{W}{K}$ and $C_{floor} = 4.0128 \times 10^6\frac{J}{K}$

---

[2] In practice, these values have been tweaked through trial and error simulations, and lead to a higher water volume.

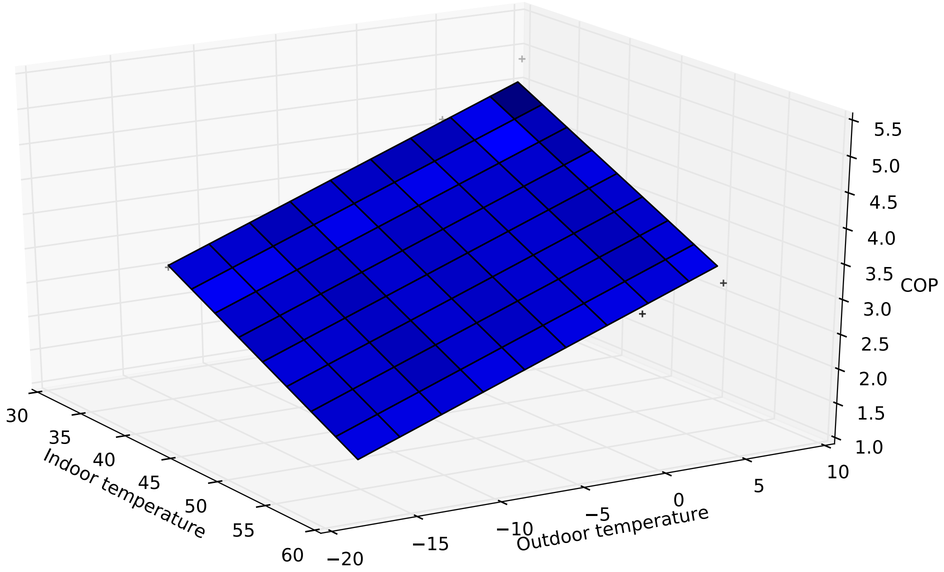


Figure A.2 – Air-to-water heat pump COP model (Indoor temperature = supply water temperature)

Air-Source to Water Heat Pump

The residential model Air-to-water Heat Pump Kita M [234] has been chosen in this study. Coefficients extracted through linear regression (Python *sklearn* toolbox) for the AWHP model are: $\text{cop}_0 = 7.010$, $\alpha = -0.0794$, and $\beta = 0.0896$. The R^2-value of the fit is 0.952, despite the small dataset provided by the manufacturer. Fig. A.2

Water tank and heater

A Westinghouse Durable 316l Stainless Steel Tank 80 Gallon (302 liters) equipped with a 4500-Watt Electric Water Heater has been elected for the Minergie house. The tank is 1,7526m high and has a diameter of 0.6m. It stores and heats water ranging from 50 $°C$ C to 80 $°C$ with a thermal efficiency of 98%. The standby loss is 0.57 %/h leading to a U-value of 1 $W/K.m^2$.

Electric Vehicle

The Minergie building hosts a Tesla Model S electric vehicle, providing an autonomy of 490 km. The chemical battery has a capacity equal to $75kWh(SoC_{max}$ = 72.5kWh) and a maximum charging power of 7kW. The charging efficiency is taken as 90%.

As a residential vehicle is most of the weekday out of the dwelling, the battery is out of the building model during that period. It is assumed that the vehicle is available everyday during t^{ev}_{start} and t^{ev}_{stop}

The vehicle arrives at the dwelling premise with an initial SoC at t^{ev}_{start} and should ideally be fully charged at t^{ev}_{stop}. It is then assumed that the dweller has the choice to offer the battery as an ESS. Alternatively, the EV might only be considered as a load to charge.

The document [34] uses US data for modeling the starting charging time and the distance range of use. From the latter, the initial state-of-charge can be derived.

A.1.2 Commercial building model

The commercial building model used in Part II (Advanced integration of Smart-Buildings into the Smart-Grid) has been provided by the Automatic Control Laboratory at EPFL [235]. It initially comes from a detailed Energyplus model that was subsequently linearized via the OpenBuild toolbox developed in the same laboratory [46]. The resulting model can therefore be written as:

$$\begin{aligned} \boldsymbol{x}[t+1] &= A\,\boldsymbol{x}[t] + B_u\,\boldsymbol{u}[t] + B_d\,\boldsymbol{d}[t] \\ \boldsymbol{T}[t] &= C\,\boldsymbol{x}[t] \end{aligned}$$

where x represents the building state that does not have a straightforward physical meaning. Matrices A describes the losses in the system, B_u maps the HVAC input power to state change, B_d the disturbance to state change, and C the state to zone thermal comfort. These matrices result from the discretization of a continuous model, given a discretization step $dt = 20min$:

$$\boldsymbol{x_0} = \begin{bmatrix} -6.73\;10^{3} \\ 1.34\;10^{3} \\ -5.96\;10^{2} \\ 9.22 \\ -4.21\;10^{2} \\ -3.94\;10^{-1} \\ -1.23\;10^{1} \\ -3.01\;10^{1} \\ -7.55\;10^{-1} \\ 1.38\;10^{2} \end{bmatrix},$$

$$A = \begin{bmatrix}
9.90\;10^{-1} & -3.41\;10^{-2} & -3.50\;10^{-2} & 3.95\;10^{-4} & 1.79\;10^{-2} & 9.36\;10^{-5} & 9.89\;10^{-3} & 1.15\;10^{-3} & 8.01\;10^{-6} & 1.60\;10^{-3} \\
-2.95\;10^{-2} & 7.16\;10^{-1} & -2.58\;10^{-1} & 2.26\;10^{-3} & -7.20\;10^{-2} & -3.36\;10^{-4} & 8.80\;10^{-2} & -4.10\;10^{-2} & 2.18\;10^{-4} & 1.15\;10^{-2} \\
-3.22\;10^{-2} & -2.62\;10^{-1} & 7.25\;10^{-1} & 2.54\;10^{-3} & 2.03\;10^{-2} & 1.36\;10^{-3} & 1.30\;10^{-1} & -2.82\;10^{-2} & 4.50\;10^{-4} & 2.22\;10^{-2} \\
3.81\;10^{-4} & 2.33\;10^{-3} & 2.54\;10^{-3} & 9.76\;10^{-1} & -2.03\;10^{-3} & 9.50\;10^{-2} & -1.88\;10^{-3} & -9.56\;10^{-5} & 1.67\;10^{-2} & -3.75\;10^{-4} \\
1.89\;10^{-2} & -6.29\;10^{-2} & 2.32\;10^{-2} & -2.06\;10^{-3} & 5.61\;10^{-1} & -7.97\;10^{-4} & -9.52\;10^{-2} & -1.66\;10^{-1} & 5.71\;10^{-4} & -4.54\;10^{-2} \\
9.42\;10^{-5} & -2.49\;10^{-4} & 1.43\;10^{-3} & 9.50\;10^{-2} & -1.09\;10^{-3} & 5.14\;10^{-1} & 1.84\;10^{-4} & -1.34\;10^{-3} & -1.40\;10^{-1} & -4.03\;10^{-4} \\
7.10\;10^{-3} & 8.21\;10^{-2} & 1.17\;10^{-1} & -1.71\;10^{-3} & -8.10\;10^{-2} & 2.76\;10^{-4} & 7.51\;10^{-1} & -1.28\;10^{-2} & -1.03\;10^{-4} & -1.20\;10^{-1} \\
-1.41\;10^{-3} & -4.62\;10^{-2} & -4.23\;10^{-2} & 1.53\;10^{-4} & -1.30\;10^{-1} & -1.05\;10^{-3} & -2.09\;10^{-2} & 7.32\;10^{-1} & 4.58\;10^{-4} & -1.10\;10^{-1} \\
-3.45\;10^{-6} & 2.13\;10^{-4} & 4.14\;10^{-4} & 1.67\;10^{-2} & 7.67\;10^{-4} & -1.40\;10^{-1} & -1.39\;10^{-4} & 4.42\;10^{-4} & 7.51\;10^{-1} & -7.16\;10^{-4} \\
1.77\;10^{-4} & 2.23\;10^{-3} & 4.72\;10^{-3} & -6.04\;10^{-5} & -1.05\;10^{-2} & -2.80\;10^{-4} & -2.45\;10^{-2} & -1.61\;10^{-3} & -6.14\;10^{-4} & 6.69\;10^{-1}
\end{bmatrix}$$

Figure A.3 – BLR building and casino microgrid

$$B_u = \begin{bmatrix} -1.29 & -5.13 & -1.28 \\ 1.04 & -2.36\,10^{1} & 1.07 \\ -1.86 & -1.64\,10^{1} & -1.95 \\ 4.09 & 1.20\,10^{-1} & -3.98 \\ 8.49 & -3.85 & 8.39 \\ -1.02\,10^{1} & -8.06\,10^{-3} & 1.03\,10^{1} \\ 2.40 & 2.33 & 2.37 \\ 1.95 & -9.06\,10^{-1} & 1.95 \\ -9.55\,10^{-1} & 5.80\,10^{-3} & 9.58\,10^{-1} \\ 5.21\,10^{-1} & 1.42\,10^{-1} & 5.23\,10^{-1} \end{bmatrix}, B_d = \begin{bmatrix} -1.65 & -3.62\,10^{-1} & -6.10 \\ -7.96\,10^{-2} & 3.28\,10^{-2} & -2.16 \\ -1.05 & -7.09\,10^{-2} & -6.02 \\ 1.92\,10^{-2} & -1.57\,10^{-2} & 1.06\,10^{-1} \\ 2.06 & 1.06\,10^{-1} & 1.17\,10^{1} \\ 1.02\,10^{-2} & 8.01\,10^{-4} & 5.87\,10^{-2} \\ -7.85\,10^{-1} & -3.19\,10^{-1} & 1.29\,10^{-1} \\ -9.03\,10^{-1} & -2.63\,10^{-1} & -1.84\,10^{-1} \\ -4.12\,10^{-3} & -1.31\,10^{-2} & -7.60\,10^{-3} \\ 1.92 & 1.40\,10^{-1} & -8.36\,10^{-1} \end{bmatrix}, \quad C = \begin{bmatrix} -3.29\,10^{-3} & -6.39\,10^{-3} & -3.28\,10^{-3} \\ 1.14\,10^{-3} & -2.84\,10^{-2} & 1.17\,10^{-3} \\ -3.64\,10^{-3} & -2.26\,10^{-2} & -3.73\,10^{-3} \\ 4.19\,10^{-3} & 1.87\,10^{-4} & -4.01\,10^{-3} \\ 1.41\,10^{-2} & -3.67\,10^{-3} & 1.40\,10^{-2} \\ -1.22\,10^{-2} & -1.95\,10^{-6} & 1.24\,10^{-2} \\ 2.90\,10^{-3} & 5.81\,10^{-3} & 2.86\,10^{-3} \\ 3.09\,10^{-3} & -2.52\,10^{-3} & 3.09\,10^{-3} \\ -2.15\,10^{-3} & 1.72\,10^{-5} & 2.15\,10^{-3} \\ 1.45\,10^{-3} & 1.06\,10^{-3} & 1.45\,10^{-3} \end{bmatrix}^T$$

This model represents the thermal dynamic of a three-zone building, each containing an HVAC system and subject to internal heat gain (loads and humans), solar radiation, and outside air temperature disturbances.

A.1.3 Blue Lake Rancheria test site

BLR is a city located in Humbold Country in the north of California, United States. Among other things, the city hosts tribal government offices, a hotel & casino, and a gas station with a convenience store. Due to its location, the area is frequently subject to intense natural disaster such as forest fires and heavy rainstorms, often causing power outages. Loosing the electrical power coming from the coast could therefore endangered the isolated community.

To this end, Schatz Energy Research Center (SERC) successfully deployed a microgrid technology in 2016. A total capacity of $420kw_{ac}$ of PV array locally can be used to power the local loads or charge the 500 kW/950 kWh battery energy storage system. A microgrid management system allows for an optimized autonomous control over a forecasted planning horizon, leading to microgrid protection, control, data acquisition, and forecasting. The project received many awards for RES integration and its resulting resiliency. In 2017 a nearby wildfire isolated BLR from the main grid, and the microgrid could successfully fulfil the need of local energy demand. In addition to its resiliency, the local RES and smart microgrid management decrease greenhouse gas emission and the annual electricity costs. More information can be found in the final report [236].

Basic building energy management strategies were enabled in that first project, in the form of discrete priority load shedding. Nevertheless, further optimization could be performed through controlling the loads consuming the local production. A second project led by SERC in collaboration with LBNL, called *Solar+ Optimizer* aims to develop a generic toolkit for commercial building energy management. The ongoing Electric Program Investment Charge (EPIC) project started in 2018 and intends to provide:

- A hardware toolkit for designing integrated *Solar+* packages.
- An open-source software for controlling the technology.
- A set of site targeting guidelines for helping optimize which sites are the best candidates for investment given the local conditions and the potential for coordination.

When taken together, the main outcomes of the *Solar+* project focus on the interconnection between advanced control in commercial buildings, solar, and energy storage system. Ideally, the project should constitute a set of market-ready technologies that could be deployed all over California, in order to improve resiliency and decrease greenhouse gas emissions.

The primary target represents gas stations and convenience stores, for their canopy can easily host solar. Fig. A.3 shows the gas station and convenience store (top), the 420 kW solar installation microgrid deployed during the first project (bottom left), and the conditioned goods sold in the store (bottom right). A first part of the *Solar+ Optimizer* project consisted in deploying 60 kW of solar as depicted in Fig. A.4.

A 174 kWh battery was further installed behind the convenience store and wired to the rest of the system to complete the local microgrid. That microgrid can be islanded from the main grid (PG&E) at any moment via to a SEL Relay and PCC breaker. Table A.5 provides detailed information for each of the components.

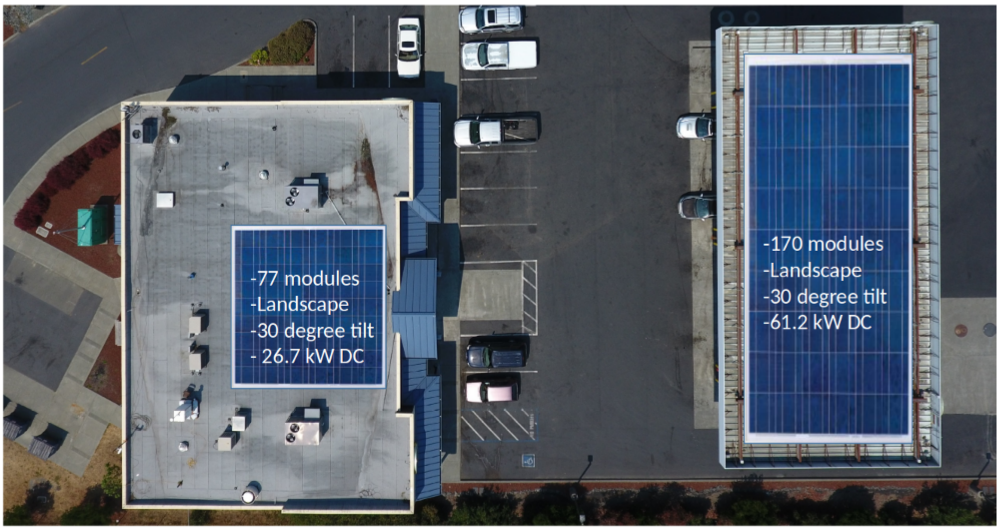


Figure A.4 – *Solar+* project: PV installation on gas station and convenience store

Solar+ Optimizer

The main innovative research lays in the development of the *Solar+ Optimizer* block that drives the controllable loads. Sharing similarities with the Open Building Data Management System (OpenBMS) and OpenEMS presented in Chapter 2, the software leverages ambient forecast (weather, prices, etc), sensors data, and building model in order to optimally drive the flexible entities setpoints. Fig. A.5 depicts the interconnection between the *Solar+ Optimizer* and the surrounding modules/services of the project. eXtensible Building Operating System (XBOS) [60], the core of the system, connects the various Smart-Building components together. The module offers the following features:

- Real-time monitoring of building sensors - it collects data from smart-sensor to monitor the state of the building. Services data such as weather forecast and utility DR signals can also be pushed to the building operating system.
- Control of building actuators - it relays high-level commands to low-level hardware, agnostic to the protocol of the latter.
- Collection, modeling, and analytics of building data - the module stores the building metadata and understands high level queries for further analysis.
- Advanced management and coordination of building systems/subsystems - the module expose its data through PUB/SUB protocol to higher level modules that can leverage the building state and metadata information.

The building operating system relies on a BOSSWAVE data bus to convey data from one sub-module to another. The bus mainly interconnects hardware drivers for sensor/actuator

Table A.5 – Blue Lake *Solar+* demonstrator: new installed hardware

Name	Data	Comment
PV array/modules SunPower	$P_{nom} = 60.120 kW$	Av. efficiency 22.2% Sunny Tripower CORE1 PV system invertor CK Johnson Rack substructure and IronRidge PV Rack
Battery storage system Tesla Powerpack2	Cap. 174 kWh Power 109 kW	140 kVA inverter
Refrigeration controllers Parker	3 controllers	/
Programmable thermostat Venstar	1 thermostat	/
Power meters AcuEnergy	2 meters	Acuvim II-D-RCT-P1 (2 comm modules) AXM-WEB (6 CTs) RCT16-1000
Islanding real-time automation controller SEL	/	/

communication, various databases for data management, and higher-level applications. More information can be found in the documentation [237].

Constituting the main application of XBOS in the frame of the *Solar+ Optimizer* project, the **MPC module** continuously orchestrates the building power flows. The application is a Python process based on the MPCPy package [153] that periodically calls XBOS API to update the necessary exodata and the state of the building model used for control. The predictive controller can then solve the economic optimization problem to drive the thermostat set points, fridge & freezer set points, and battery power. The particularity of MPCPy lays in the use of Modelica, a equation-based and object-oriented modeling language. The tool JModelica leverages the derived building model to practically solve the constrained optimization problem. Beyond its optimization usefulness for MPC purpose, the tool can also perform simulation and analysis, which can be of interest for user interface.

The Modelica-based system used for representing the whole *Solar+* project site is depicted in Fig. A.6 (left). The bloc diagram shows the relationship between the input variables and the output variables of the system, made of a thermally-driven building, a PV array, and a battery. The input variables range from uncontrollable parameters (radiation, outside temperature, etc) to control variable (battery charge/discharge, freezer state, etc).

A lumped capacitance models the thermal behavior of the building, Fig. A.6 (right). Three conditioned spaces make up the entire convenience store model: the store zone conditioned by the RTU, a first food zone conditioned by the refrigerator system, and a second food zone conditioned by the freezer. Due to a 24h per day presence and the need to keep food cool, the

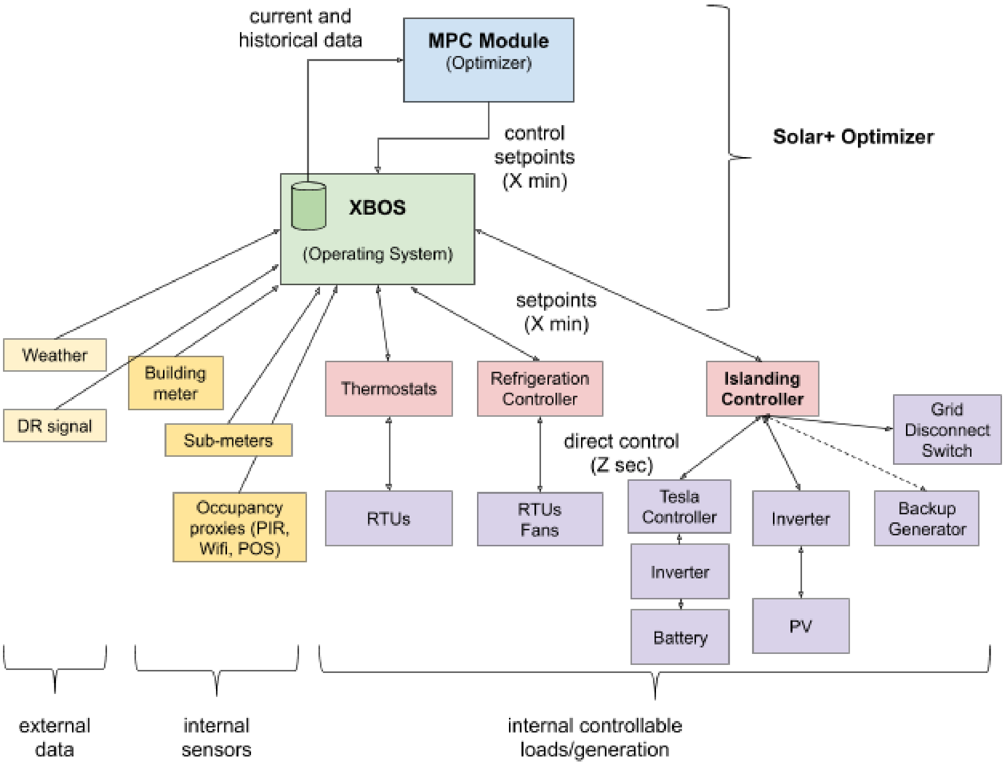


Figure A.5 – *Solar+ Optimizer* software architecture and its connection with microgrid entities

temperature constraints must be ensured continuously without any setback.

The RTU system is composed of a 2-stage cooling unit and a 1-stage gas heating unit. In cooling mode, the maximum power cannot exceed 29.3 kW and the load is assumed to work with a fixed COP of 2.59; in heating mode, the capacity is 24.91 kW with a 0.8 efficiency. Concerning the refrigerator, it can draw a maximum power capacity of 5.861 kW with a cooling COP of 1.3; The freezer cooling capacity is 6.096 kW with COP 1.7. The modeling approach of refrigerator and freezer is similar to the RTU system assuming constant capacities and efficiencies.

The input cooling/heating signals can continuously take values from 0 to 1. Concerning the battery, the SoC can not go lower than 25% and cannot exceed 100%.

Outside disturbances influence the PV output and building zone temperatures. The web service Dark Sky provide the necessary forecast to the MPC module through XBOS, such as ambient temperature, relative humidity, and solar radiation. The internal loads releasing heat inside the convenience store are mainly slot machines, whose consumption is supposed to be constant.

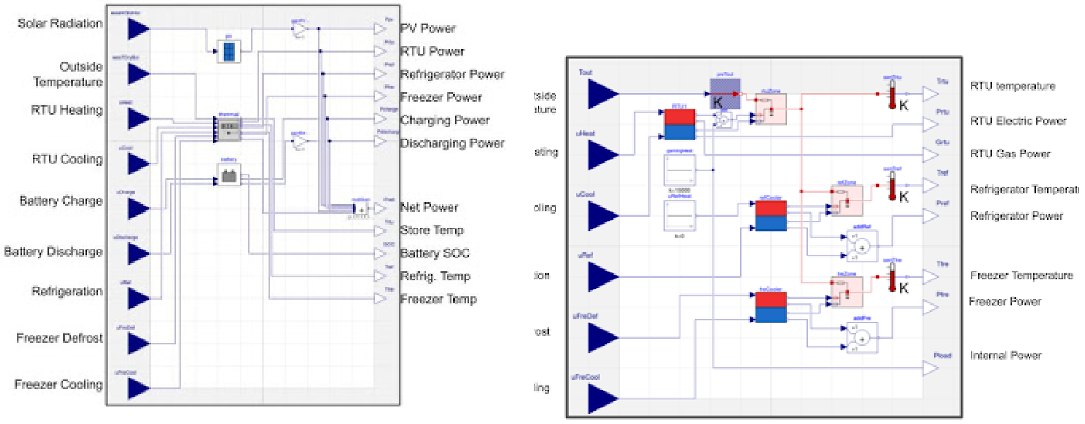


Figure A.6 – *Solar+ Optimizer* software: Modelica bloc diagram representation of (left) system model (right) thermal building model

A.2 Virtualization Engine

A.2.1 The vEngine in emulation mode

Health-check module

When emulating the behavior of a non-existing entity, the *vEngine* competes with physical middleware that run in real-time. For the emulated entities to properly work, the *vEngine* should ensure that all of them have a fair share of the computational power and no one starve. Any excessive computational starvation would entail an inappropriate behavior of the virtual component, inducing errors and non-natural delays at the BDMS premise. Moreover, as the *vEntities* are practically implemented as *uTread*, their execution only depend on the internal code and cannot therefore be controlled by a third party - like the OS.

These reasons have led to the development of the *Health-check module*, an independent module that runs into the *vMid* with the mission to monitor the pool of *vEntities* and ensure they behave appropriately. By periodically sending a special message "health report" to target *vEntities*, the *Health-check module* reconstructs the state of the pool. A target *vEntity i* will respond to a "health report" signal with the following data, computed internally:

- Computational time Q_i: the time elapsed from entity wake-up to the release of CPU.
- Effective sleeping time Δt_i: the time elapsed from a CPU release to the next CPU acquisition.
- Expected sleeping time Δt_i^a: the sleeping time explicitly specified by the *vEntity i* itself just before releasing the CPU.

Algorithm 8 Health Check module logic

```
procedure GET_POOL_STATE
    $\mathcal{U}_{vE} \leftarrow \{i \mid (\alpha_i - \overline{\alpha}) \geq \sigma_{\text{fair}}\}$
    if $\mathcal{U}_{vE}$ is not empty then
        for all j in $\{vEs\} \setminus \mathcal{U}_{vE}$ do
            $\bar{d}_j \leftarrow \overline{\alpha} \sum_{i \notin \mathcal{U}_{vE}} Q_i + \sum_{i \in \mathcal{U}_{vE}} \alpha_i \, Q_i - \alpha_j \, Q_j$
            if $\bar{d}_j \geq d_j^{max}$ then
                REDUCE_CPU_USE($\forall i, \; i \in \mathcal{U}_{vE}$)
                break
            end if
        end for
    end if
end procedure
procedure REDUCE_CPU_USE(i)
    if $vE_i$ has a simpler model then
        REDUCE_COMPLEXITY(i)
    else
        $\Delta t_i \leftarrow Q_i / \overline{\alpha}$
    end if
end procedure
```

- Other useful metrics, like the amount of received and emitted events, etc.

Given these measured quantities, the delay experienced by *vEntity* i can be expressed as $\Delta t_i - \Delta t_i^a$, and must ideally be kept below a threshold $\overline{d_i}$. Such delays depend on many dynamic factors, like the model of the *vEntities*, the hosting hardware, and the pool instance. This is the role of the *Health-check module* to keep them within bound, regardless these conditions. It does it by following Algorithm 8. The chosen approach consists in punishing the *uTread*s that are judged to be unfair with respect to the rest of the pool. To measure unfairness, one must first define the relative CPU use α_i of a *uTread* and the mean fair CPU share $\overline{\alpha}$:

$$\alpha_i = \frac{Q_i}{Q_i + \Delta t_i^a}$$
$$\overline{\alpha} = \frac{\sum_i Q_i}{\sum_i Q_i + \Delta t_i^a}$$

From these data computed by the *Health-check module*, the set $\mathcal{U}_{vE}$ of entities that are *unfair* with respect to the other can be created. An entity is considered as *unfair* when its relative CPU share α_i exceeds the mean *fair* CPU share $\overline{\alpha}$ by more than σ_{fair}. If it exists at least one *unfair* entity, vE_k, in the *vEntities* pool A, the resulting introduced delay to its peers is measured. The quantity $\tilde{d}_j$ represents the worst case delay experienced by vE_j where $vE_j \in A \wedge k \neq j$, weighted by each entity's α_j. It aims to be compared with the maximum acceptable delay $\overline{d_j}$.

If $\tilde{d}_j$ exceeds the $\overline{d_j}$ threshold, it means that the unfairness of some entities like vE_k impacts the vE_j peers. Hence, the *Health Check* module forces them to better cooperate.

The *uTread*s have at their disposal a useful mechanism to release the CPU usage, even before the full completion of their computational step: the possibility to "sleep" and give the hand to a waiting *uTread* whenever they want. This context switching can be useful during a computational heavy function, and the *vEntity* has therefore the choice to slice its execution into various chunks of time. By doing so, it is less likely to starve other entities and hence being penalized by the *Health-check module*. A more radical solution consists in changing the model in use to a simplified, less computational greedy one. These two mechanisms are enforced by the *Health-check module* through the method $reduce\_cpu\_use()$, whenever an entity is judged to be unfair, according to the method $get\_pool\_state()$.

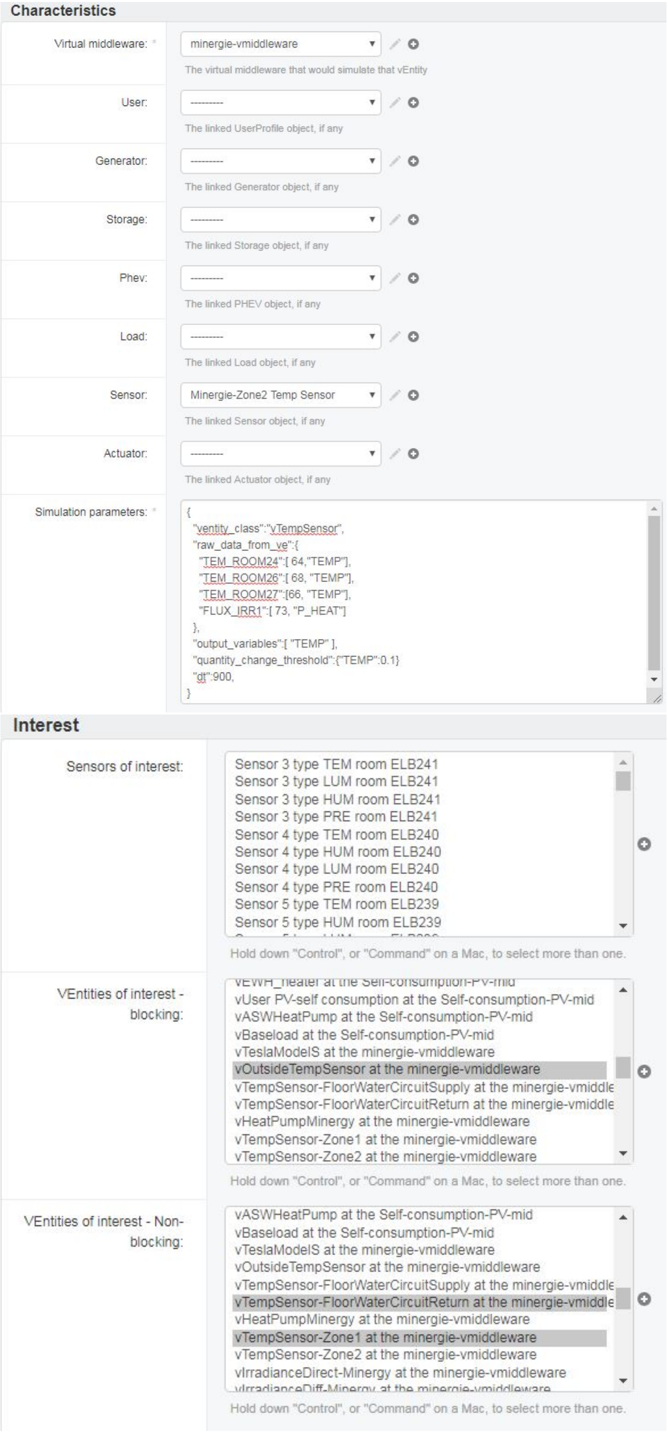
Virtual network

Delays and packet losses represent a big challenge in a cyber-physical system in which many sensors and actuators communicate with the main data management system. In this context, an emulated virtual network called *vNetwork* was developed in [13], [59] to emulate packet losses and delays virtually hampering the communication between the *vEngine* and the BDMS. Spawn as a kernel process and integrative part of the *vEngine*, it intends to replicate ICT flaws depending on the type of virtual network the user wants to emulate.

A.2.2 OpenBMS example

Fig. A.7 depicts how a virtual entity is practically encoded in the OpenBMS [1]. The example consists in a virtual temperature sensor based on the *vTempSensor* class model. As the OpenBMS is initially designed for an actual physical building, each virtual entity must be mapped to an element in the database (the sensor "Minergie-Zone2 Temp Sensor" in this example). Simulation parameters helps connecting the theoretical thermal model with parameters contained in the BDMS, such as the interfaces between rooms (3 in this case). The output data "TEMP" will be emitted only when it changes by more than $0.1\,^{\circ}C$, and the default wake-up delay is set to 15 minutes.

Concerning the dependencies, three tables are practically used. The first one allows the virtual entities to get data from actual physical sensors, but is left empty for this simulation. The second table specifies the set of blocking relationships, meaning that the entity won't run its simulation step until it receives data from that entity. The third and last table lists the non-blocking relationship, meaning that the data coming from these entities will be used at the next iteration.



Characteristics

Virtual middleware: * minergie-vmiddleware
The virtual middleware that would simulate that vEntity

User: ---------
The linked UserProfile object, if any

Generator: ---------
The linked Generator object, if any

Storage: ---------
The linked Storage object, if any

Phev: ---------
The linked PHEV object, if any

Load: ---------
The linked Load object, if any

Sensor: Minergie-Zone2 Temp Sensor
The linked Sensor object, if any

Actuator: ---------
The linked Actuator object, if any

Simulation parameters: *

```
{
 "ventity_class":"vTempSensor",
 "raw_data_from_ve":{
  "TEM_ROOM24":[ 64,"TEMP"],
  "TEM_ROOM26":[ 68, "TEMP"],
  "TEM_ROOM27":[66, "TEMP"],
  "FLUX_IRR1":[ 73, "P_HEAT"]
 },
 "output_variables":[ "TEMP" ],
 "quantity_change_threshold":{"TEMP":0.1}
 "dt":900,
}
```

Interest

Sensors of interest:
Sensor 3 type TEM room ELB241
Sensor 3 type LUM room ELB241
Sensor 3 type HUM room ELB241
Sensor 3 type PRE room ELB241
Sensor 4 type TEM room ELB240
Sensor 4 type HUM room ELB240
Sensor 4 type LUM room ELB240
Sensor 4 type PRE room ELB240
Sensor 5 type TEM room ELB239
Sensor 5 type HUM room ELB239
Hold down "Control", or "Command" on a Mac, to select more than one.

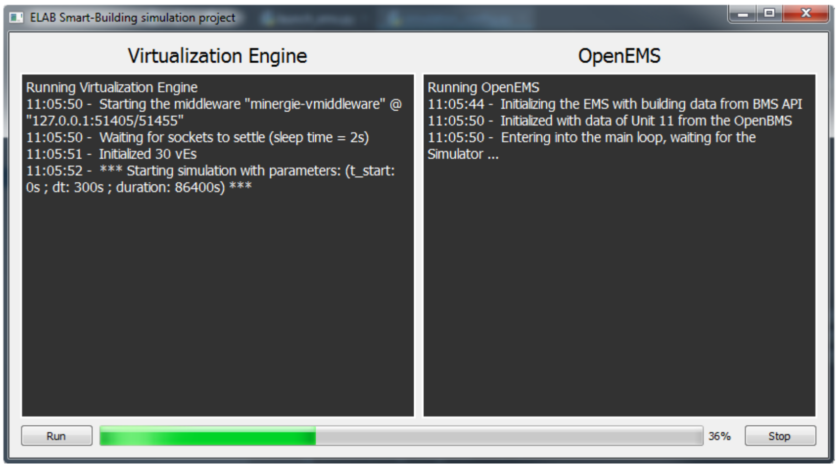
VEntities of interest - blocking:
vUser PV-self consumption at the Self-consumption-PV-mid
vASWHeatPump at the Self-consumption-PV-mid
vBaseload at the Self-consumption-PV-mid
vTeslaModelS at the minergie-vmiddleware
vOutsideTempSensor at the minergie-vmiddleware
vTempSensor-FloorWaterCircuitSupply at the minergie-vmiddle
vTempSensor-FloorWaterCircuitReturn at the minergie-vmiddle
vHeatPumpMinergy at the minergie-vmiddleware
vTempSensor-Zone1 at the minergie-vmiddleware
vTempSensor-Zone2 at the minergie-vmiddleware
Hold down "Control", or "Command" on a Mac, to select more than one.

VEntities of interest - Non-blocking:
vASWHeatPump at the Self-consumption-PV-mid
vBaseload at the Self-consumption-PV-mid
vTeslaModelS at the minergie-vmiddleware
vOutsideTempSensor at the minergie-vmiddleware
vTempSensor-FloorWaterCircuitSupply at the minergie-vmiddle
vTempSensor-FloorWaterCircuitReturn at the minergie-vmiddle
vHeatPumpMinergy at the minergie-vmiddleware
vTempSensor-Zone1 at the minergie-vmiddleware
vTempSensor-Zone2 at the minergie-vmiddleware
vIrradianceDirect-Minergy at the minergie-vmiddleware
Hold down "Control", or "Command" on a Mac, to select more than one.

Figure A.7 – BDMS example of virtual entity encoding (top) Principal characteristics (bottom) Dependencies

A.3 ELAB Smart-Building Simulator

The *ELAB Smart-Building simulator* is a Python project developed for smart-building simulation and energy management. It relies on the Virtualization Engine and the OpenEMS



```
Running Virtualization Engine
11:05:50 -  Starting the middleware "minergie-vmiddleware" @
"127.0.0.1:51405/51455"
11:05:50 -  Waiting for sockets to settle (sleep time = 2s)
11:05:51 -  Initialized 30 vEs
11:05:52 -  *** Starting simulation with parameters: (t_start:
0s ; dt: 300s ; duration: 86400s) ***
```

```
Running OpenEMS
11:05:44 -  Initializing the EMS with building data from BMS API
11:05:50 -  Initialized with data of Unit 11 from the OpenBMS
11:05:50 -  Entering into the main loop, waiting for the
Simulator ...
```

Figure A.8 – GUI of the ELAB Smart-Building simulation project

presented in Chapter 2 and 4, respectively. On the one hand, the Virtualization Engine emulates the building entities present in the building and their interactions. It allows a variable simulation time step and an overall bottom-up modeling of electrical entities as well as occupants, thermal propagation, sensors, and actuators. On the other hand, the OpenEMS periodically updates its own model of the building to change the set-points of the controllable entities and offers a user-friendly data analysis interface. The code is open-source and available at [116].

Fig. A.7 displays the user interface of the ELAB Smart-Building simulator, used in Chapter 4. Upon instantiation, the environment fetches data of the simulated building from the OpenBMS's API. Then, it spawns in parallel instances of the Virtualization Engine and the OpenEMS, providing them the necessary information to setup their own environment and connects to the OpenBMS. Once both processes are ready, the user can trigger the simulation and will see a user-friendly log file being updated as the simulation goes on.

A.3.1 Open Building Data Management System

The data coming from the Virtualization Engine to the OpenEMS must be abstracted due for the technology-agnostic feature of the designed EMS, and vice versa. This role is taken by the OpenBMS whose architecture can be seen in Figure A.9. In the bottom layer lays the low-level hardware front-end which are then interfaced by a embedded middleware agent depending on their protocol. These agents are distributed and all connect to the real-time server for data acquisition and forwarding. Data are efficiently stored in a time-series database in addition to a database gathering metadata about the building itself. These two databases can be accessed through a RestAPI interface by external third-parties. Furthermore, sockets with external application may be opened for a fast acquisition of sensed data.

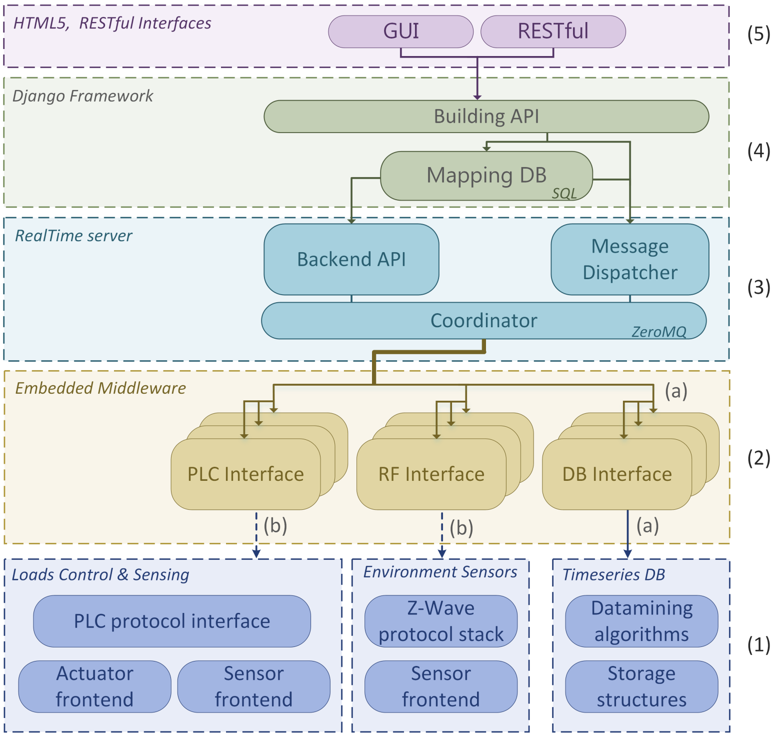


Figure A.9 – The BMS architecture used in this thesis, schema from [1]

A.3.2 Open Energy Management System

Figure A.10 shows the UML diagram on which the OpenEMS relies to generate its internal model of the building and its components. The hierarchical set of energy-related entities are based on the state-of-the-art building model presented in Chapter 1. These entities are located in a set of space (interior or exterior) that are mapped to the "Room" class. As there spaces must generally be conditioned, a class "Comfort" stores a list of BMS's sensors and actuators that connect to this space as well as user preference (e.g., temperature limits or reference). Finally, the class "Interface" links two "Room" instances with each other and provide the necessary RC model parameters and other useful properties. It's worth noting that occupant's actions are implicitly stored in a set of stochastic model parameters for every concerned component (e.g., a probabilistic set of arrival/leaving time of the EV or a Load Profile structure for loads).

A.4 Non-Controllable Loads Model

Figure A.11 shows activity clustering for two other appliances of the ECO dataset in addition to the three others presented in Chapter 3.

A.5 Battery Equivalent Model

The BEM violation time is derived as follow, for a positive value of u_i:

$$\left\{\begin{array}{ll} \dot{x}(t) & = -\alpha x(t) + u_i \\ x(t=0) & = x_0 \\ x(t=t_{v,i}) & = C \end{array}\right. \rightarrow \left\{\begin{array}{ll} x(t) & = a + b\, e^{-\alpha t} \\ a+b & = -x_0 + u_i \\ a + b\, e^{-\alpha t_{v,i}} & = -\alpha C + u_i \end{array}\right. \tag{A.1}$$

$$\left\{\begin{array}{ll} a & = \frac{u_i}{\alpha} \\ b & = \frac{x_0 - u_i}{\alpha} \\ e^{-\alpha t_{v,i}} & = \frac{C - \frac{u_i}{\alpha}}{b} \end{array}\right. \tag{A.2}$$

and since $x_0 \leq C$, $t_{v,i}$ exists only if $u_i \geq \frac{C}{\alpha}$.

In case of discharge, i.e. $u_i < 0$, the battery SoC tends towards $-C$:

$$\left\{\begin{array}{ll} \dot{x}(t) & = -\alpha x(t) + u_i \\ x(t=0) & = x_0 \\ x(t=t_{v,i}) & = -C \end{array}\right. \rightarrow \left\{\begin{array}{ll} x(t) & = a + b\, e^{-\alpha t} \\ a+b & = -x_0 + u_i \\ a + b\, e^{-\alpha t_{v,i}} & = \alpha C + u_i \end{array}\right. \tag{A.3}$$

$$\left\{\begin{array}{ll} a & = \frac{u_i}{\alpha} \\ b & = \frac{x_0 - u_i}{\alpha} \\ e^{-\alpha t_{v,i}} & = \frac{-C - \frac{u_i}{\alpha}}{b} \end{array}\right. \tag{A.4}$$

and since $x_0 \geq -C$, $t_{v,i}$ exists only if $u_i \leq \frac{-C}{\alpha}$.

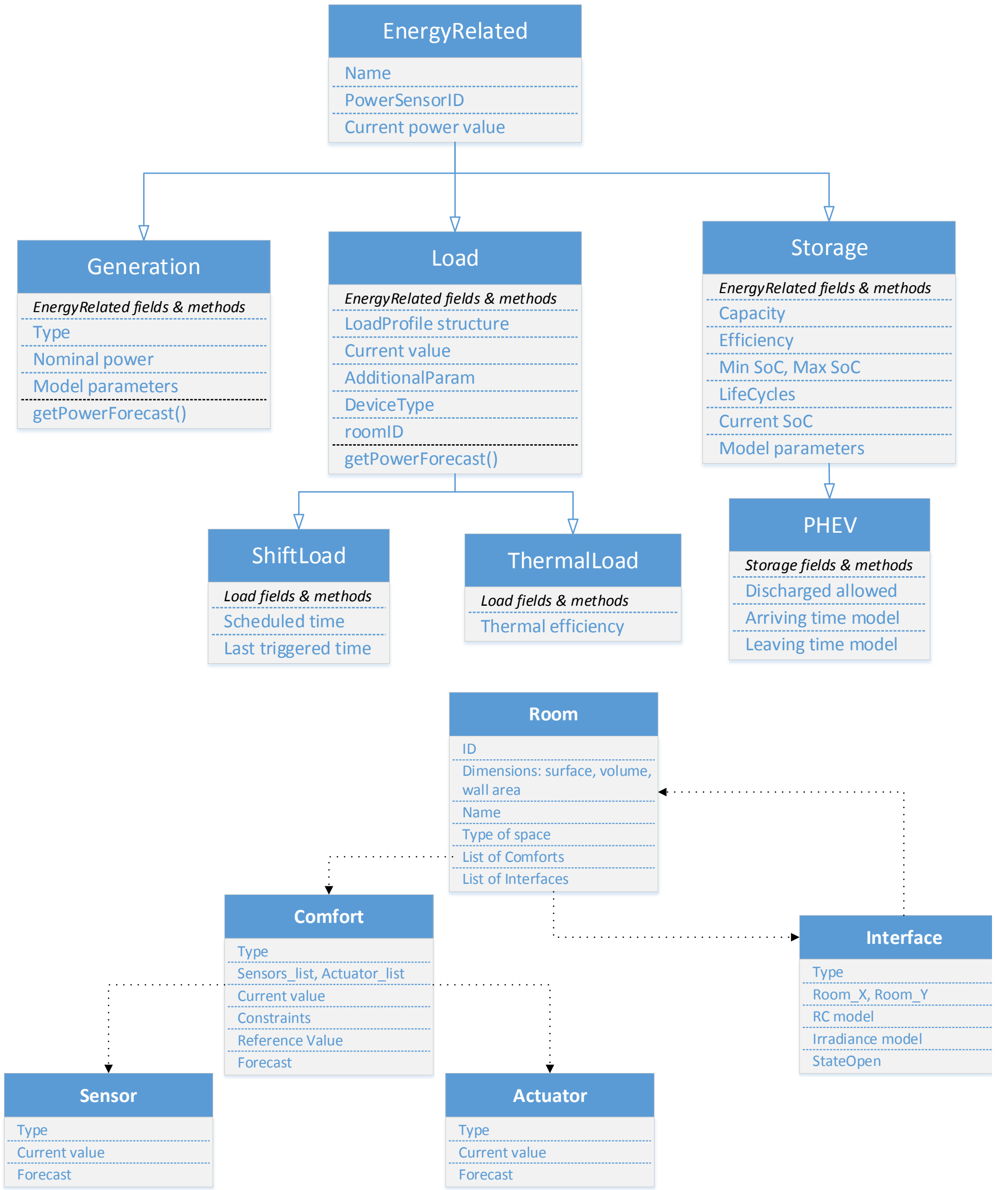


Figure A.10 – UML class diagrams of OpenEMS (top) Energy-related classes (bottom) Zone-related classes

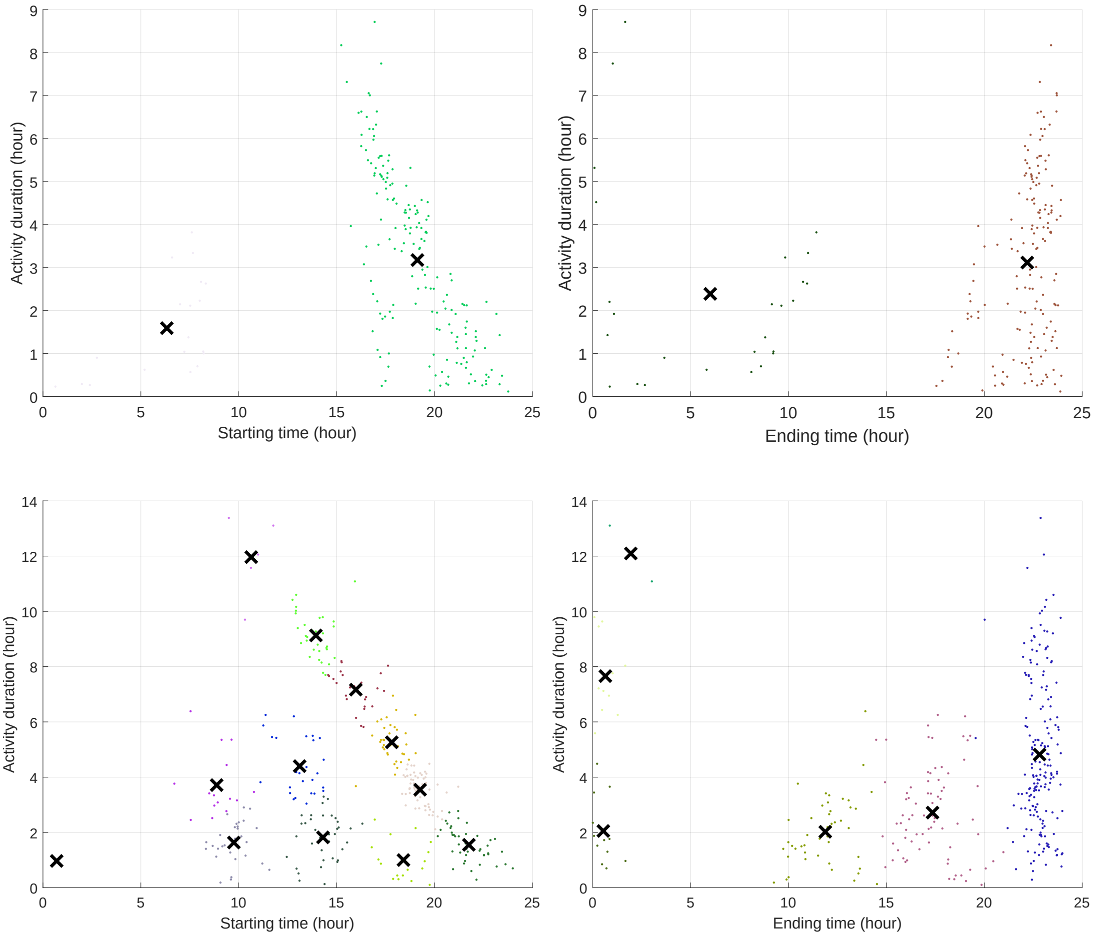


Figure A.11 – Activity clustering: duration time versus (left) starting time (right) ending time. The top figures refer to a living room lamp, and the bottom figures to the TV in the ECO dataset

Bibliography

[1] G. Lilis, G. Conus, and M. Kayal, “A Distributed, Event-driven Building Management Platform on Web Technologies,” in *1st International Conference on Event-Based Control, Communication, and Signal Processing*, Krakow, 2015.

[2] European Commision, “EU climate action,” https://ec.europa.eu/clima/citizens/eu_en, accessed: 2019-10-27.

[3] California Legislative Information, “Senate Bill No. 100,” https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201720180SB100, accessed: 2019-10-27.

[4] B. Kroposki, B. Johnson, Y. Zhang, V. Gevorgian, P. Denholm, B. M. Hodge, and B. Hannegan, “Achieving a 100% Renewable Grid: Operating Electric Power Systems with Extremely High Levels of Variable Renewable Energy,” *IEEE Power and Energy Magazine*, vol. 15, no. 2, pp. 61–73, 2017.

[5] L. Kristov, “The Bottom-Up (R)Evolution of the Electric Power System: The Pathway to the Integrated-Decentralized System,” *IEEE Power and Energy Magazine*, vol. 17, pp. 42–49, 2018.

[6] X. Li and J. Wen, “Review of building energy modeling for control and operation,” *Renewable and Sustainable Energy Reviews*, vol. 37, pp. 517–537, 2014.

[7] G. Lilis, G. Conus, N. Asadi, and M. Kayal, “Towards the next generation of intelligent building: An assessment study of current automation and future IoT based systems with a proposal for transitional design,” *Sustainable Cities and Society*, vol. 28, pp. 473–481, 2016.

[8] H. Farhangi, “The Path of the Smart Grid,” *IEEE power & energy magazine*, vol. 8, pp. 18–28, 2010.

[9] U.S. Department of Energy, “Benefits of Demand Response in Electricity Markets and Recommendations for Achieving Them,” 2006.

[10] S. Nolan and M. O’Malley, “Challenges and barriers to demand response deployment and evaluation,” *Applied Energy*, vol. 152, pp. 1–10, 2015.

[11] N. Good, K. A. Ellis, and P. Mancarella, “Review and classification of barriers and enablers of demand response in the smart grid,” *Renewable and Sustainable Energy Reviews*, vol. 72, no. November 2016, pp. 57–72, 2017.

[12] E. Vrettos, “Control of Residential and Commercial Loads for Power System Ancillary Services,” Ph.D. dissertation, 2016.

[13] G. Lilis, “A Scalable and Secure System Architecture for Smart Buildings,” Ph.D. dissertation, 2017.

[14] ASHRAE, *ASHRAE HANDBOOK*, 2013.

[15] G. Fraisse, C. Viardot, O. Lafabrie, and G. Achard, “Development of a simplied and accurate building model based on electrical analogy,” *Energy and Buildings*, vol. 34, pp. 1017–1031, 2002.

[16] S. Wang, “Optimal simplified thermal models of building envelope based on frequency domain regression using genetic frequency domain regression using genetic algorithm,” *Energy and Buildings*, 2007.

[17] B. Lehmann, D. Gyalistras, M. Gwerder, K. Wirth, and S. Carl, “Intermediate complexity model for Model Predictive Control of Integrated Room Automation,” *Energy & Buildings*, vol. 58, pp. 250–262, 2013.

[18] E. Rodríguez Jara, F. J. Sánchez de la Flor, S. Álvarez Domínguez, J. L. Molina Félix, and J. M. Salmerón Lissén, “A new analytical approach for simplified thermal modelling of buildings: Self-Adjusting RC-network model,” *Energy and Buildings*, vol. 130, pp. 85–97, 2016.

[19] D. H. Blum, K. Arendt, L. Rivalin, M. A. Piette, M. Wetter, and C. T. Veje, “Practical factors of envelope model setup and their effects on the performance of model predictive control for building heating, ventilating, and air conditioning systems,” *Applied Energy*, vol. 236, no. November 2018, pp. 410–425, 2019.

[20] M. Wetter, “Multizone building model for thermal building simulation in modelica,” *Submitted to: Modelica conference 2006*, pp. 517–526, 2006.

[21] G. Goddard, J. Klose, and S. Backhaus, “Model development and identification for fast demand response in commercial hvac systems,” *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 2084–2092, 2014.

[22] R. McDowall, *Fundamentals of HVAC Systems: SI Edition.* American Society of Heating, Refrigerating and Air-Conditioning Engineers eLearning, 2007. [Online]. Available: https://books.google.ch/books?id=6FQGW-NsancC

[23] C. Verhelst, D. Axehill, C. N. Jones, and L. Helsen, “Impact of the cost function in the optimal control formulation for an air-to-water heat pump system,” *Simulation*, vol. 1, no. 1, pp. 1–21.

[24] P. Dolan, M. Nehrir, and V. Gerez, “Development of a monte carlo based aggregate model for residential electric water heater loads,” *Electric Power Systems Research*, vol. 36, no. 1, pp. 29 – 35, 1996.

[25] L. Paull, D. MacKay, H. Li, and L. Chang, “A water heater model for increased power system efficiency,” *Canadian Conference on Electrical and Computer Engineering*, no. 1, 2009.

[26] L. Paull, H. Li, and L. Chang, “A novel domestic electric water heater model for a multi-objective demand side management program,” *Electric Power Systems Research*, vol. 80, no. 12, pp. 1446–1451, 2010.

[27] M. H. Nehrir, R. Jia, D. A. Pierre, and D. J. Hammerstrom, “Power management of aggregate electric water heater loads by voltage control,” *2007 IEEE Power Engineering Society General Meeting, PES*, no. 1, 2007.

[28] Z. Xu, R. Diao, S. Lu, J. Lian, and Y. Zhang, “Modeling of electric water heaters for demand response: A baseline PDE model,” *IEEE Transactions on Smart Grid*, vol. 5, no. 5, pp. 2203–2210, 2014.

[29] S. Koch, “Demand response methods for ancillary services and renewable energy integration in electric power systems,” Ph.D. dissertation, 2012.

[30] T. Hubert and S. Grijalva, “Realizing Smart Grid Benefits Requires Energy Optimization Algorithms at Residential Level,” *ISGT 2011*, pp. 2–9, 2011.

[31] M. Rastegar, M. Fotuhi-firuzabad, and F. Aminifar, “Load commitment in a smart home,” *Applied Energy*, vol. 96, pp. 45–54, 2012.

[32] M. Jongerden and B. Haverkort, “Which battery model to use?” *IET Software*, vol. 3, no. 6, p. 445, 2008.

[33] E. Vrettos, K. Lai, F. Oldewurtel, and G. Andersson, “Predictive Control of buildings for Demand Response with dynamic day-ahead and real-time prices,” *Control Conference (ECC), 2013 European*, pp. 2527–2534, 2013.

[34] S. Zambrano-Perilla, G. Ramos, and D. Celeita, “Modeling and impacts of plug-in electric vehicles in residential distribution systems with coordinated charging schemes,” *International Review on Modelling and Simulations*, vol. 9, no. 4, pp. 227–237, 2016.

[35] D. S. H. Chan and J. C. H. Phang, “Analytical Methods for the Extraction of Solar-Cell Single-and Double-Diode Model Parameters from I-V Characteristics,” *IEEE Transactions on Electron Devices*, vol. 34, no. 2, pp. 286–293, 1987.

[36] D. Sera, R. Teodorescu, and P. Rodriguez, “PV panel model based on datasheet values,” *IEEE International Symposium on Industrial Electronics*, no. 4, pp. 2392–2396, 2007.

[37] H. Tian, F. Mancilla-David, K. Ellis, E. Muljadi, and P. Jenkins, "A cell-to-module-to-array detailed model for photovoltaic panels," *Solar Energy*, vol. 86, no. 9, pp. 2695 – 2706, 2012.

[38] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. Buhl, Y. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, and J. Glazer, "Energyplus: creating a new-generation building energy simulation program," *Energy and Buildings*, vol. 33, no. 4, pp. 319 – 331, 2001.

[39] A. S and C. C, "A comparison of energyplus to doe-2.1e: Multiple cases ranging from a sealed box to a residential building," *Energy Systems Laboratory (http://esl.tamu.edu)*. [Online]. Available: http://hdl.handle.net/1969.1/90773

[40] S. Klein, J. A. Duffie, and W. A. Beckman, "Trnsys – a transient simulation program." *ASHRAE Transaction*, p. 623–633, 1976.

[41] University of Strathclyde, UK, "Esp-r, a modelling tool for building performance simulation," http://www.esru.strath.ac.uk/Programs/ESP-r.htm, 2010.

[42] M. E, "The home energy saver: Interactive energy information and calculations on the web," *Center for Building Science News(4):1-2. LBL/PUB-731.*

[43] M. Malhotra, P. Im, G. K. Accawi, M. P. Ternes, and M. MacDonald, "Multifamily tool for energy audits (multea) engineering manual (version 1)," 2018.

[44] M. Wetter, "Co-Simulation of Building Energy and Control Systems with the Building Controls Virtual Test Bed ," *Journal of Building Performance Simulation*, vol. 4, no. 3, pp. 185–203, 2011.

[45] W. Bernal, M. Behl, T. Nghiem, and R. Mangharam, "Mle+: a tool for integrated design and deployment of energy efficient building controls," in *BuildSys '12 Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. ACM New York, NY, USA 2012, 2012, pp. 123–130.

[46] T. T. Gorecki, F. a. Qureshi, and C. N. Jones, "OpenBuild : An Integrated Simulation Environment for Building Control," in *2015 IEEE Conference on Control Applications (CCA)*, no. 7320826, 2015, pp. 1522–1527.

[47] G. Engel, A. S. Chakkaravarthy, and G. Schweiger, "Co-simulation between trnsys and simulink based on type155," in *Software Engineering and Formal Methods*, A. Cerone and M. Roveri, Eds. Springer International Publishing, 2018, pp. 315–329.

[48] "Simscape block libraries," https://www.mathworks.com/help/physmod/simscape/index.html, 2018.

[49] M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica buildings library," *Journal of Building Performance Simulation*, vol. 7, no. 4, pp. 253–270, 2014.

[50] G. Morganti, A. M. Perdon, G. Conte, and D. Scaradozzi, “Multi-Agent System Theory for Modelling a Home Automation System,” in *10th International Work-Conference on Artificial Neural Networks*, 2009, pp. 585–593.

[51] B. De Carolis, G. Cozzolongo, S. Pizzutilo, and V. L. Plantamura, “Agent-Based Home Simulation and Control,” in *Foundations of Intelligent Systems*, 2005, no. April 2016, pp. 404–412.

[52] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and X. Zhang, “The umass intelligent home project,” 01 1999, pp. 291–298.

[53] M. Hu, F. Li, M. Haghnevis, Y. Khodadadegan, L. M. Sanchez, S. Wang, X. Zhuang, and T. Wu, “AN AGENT BASED SIMULATION FOR BUILDING ENERGY SYSTEM MODELING,” in *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, 2010, pp. 1–8.

[54] A. G. Lez-Briones, F. De La Prieta, M. S. Mohamad, S. Omatu, and J. M. Corchado, “Multi-agent systems applications in energy optimization problems: A state-of-the-art review,” *Energies*, vol. 11, no. 8, pp. 1–28, 2018.

[55] J. Park, M. Moon, S. Hwang, and K. Yeom, “CASS: A Context-Aware Simulation System for Smart Home,” in *5th ACIS International Conference on Software Engineering Research, Management & Applications*, 2007.

[56] J. Venkatesh, B. Aksanli, J. C. Junqua, P. Morin, and T. S. Rosing, “HomeSim: Comprehensive, smart, residential electrical energy simulation and scheduling,” in *International Green Computing Conference Proceedings*, june 2013, pp. 1–8.

[57] B. Wang and J. S. Baras, “HybridSim: A modeling and co-simulation toolchain for cyber-physical systems,” *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pp. 33–40, 2013.

[58] G. Lilis, O. Van Cutsem, and M. Kayal, “Building Virtualization Engine: a Novel Approach Based on Discrete Event Simulation,” in *2nd International Conference on Event-Based Control, Communication, and Signal Processing*, 2016.

[59] ——, “A High-Speed Integrated building emulation engine based on discrete event simulation,” *Journal of Systems Architecture*, vol. 92, pp. 53–65, 2018.

[60] G. Fierro and D. E. Culler, “Xbos: An extensible building operating system,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-197, Sep 2015. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-197.html

[61] G. Lilis and O. Van Cutsem, “vmiddleware: a real time building emulation engine,” https://gitlab.com/georgekav/ib_backend_virtualization.git, 2017.

[62] M. Drăgoicea, L. Bucur, and M. Pătraşcu, “A Service Oriented Simulation Architecture for Intelligent Building Management,” in *Exploring Services Science*, 2013, pp. 14–28.

[63] F. Akgul, *ZeroMQ.* Packt Publishing, 2013.

[64] P. O. Siebers, C. M. Macal, J. Garnett, D. Buxton, and M. Pidd, “Discrete-event simulation is dead, long live agent-based simulation!” *Journal of Simulation*, vol. 4, no. 3, pp. 204–210, 2010.

[65] A. Gustafsson, “Threads Without the Pain,” *Queue*, vol. 3, no. 9, 2005.

[66] C. T. Clack, “Modeling solar irradiance and solar PV power output to create a resource assessment using linear multiple multivariate regression,” *Journal of Applied Meteorology and Climatology*, vol. 56, no. 1, pp. 109–125, 2017.

[67] A. Sharda and S. Kumar, “Heat transfer through windows : A Review Heat Transfer Through Windows : a Review,” *International Conference on Advances in Mechanical Engineering*, no. August 2009, 2017.

[68] F. Beyeler, N. Beglinger, and U. Roder, “Minergie: The Swiss Sustainable Building Standard,” *Innovations: Technology, Governance, Globalization*, vol. 4, no. 4, pp. 241–244, 2009.

[69] Swiss confederation, “Low energy buildings in switzerland? buildings with minergie certificate (sfoe),” https://www.geo.admin.ch/en/news/datasetoftheweek.detail.news.html/geo-internet/datasetoftheweek2018/datasetoftheweek20180213.html, 2018.

[70] “Meteo Swiss,” https://www.meteosuisse.admin.ch, accessed: 2019-08-13.

[71] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini, “The ECO Data Set and the Performance of Non-Intrusive Load Monitoring Algorithms,” *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pp. 80—-89, 2014.

[72] O. Van Cutsem, G. Lilis, and M. Kayal, “Automatic multi-state load profile identification with application to energy disaggregation,” *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pp. 1–8, 2018.

[73] A. H. Helen, S. Greenberg, and E. Huang, “One size does not fit all: applying the transtheoretical model to energy feedback technology design,” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 927–936, 2010.

[74] M. Hayn, V. Bertsch, and W. Fichtner, “Electricity load profiles in Europe: The importance of household segmentation,” *Energy Research and Social Science*, vol. 3, pp. 30–45, 2014.

[75] Y. Wang, Q. Chen, C. Kang, and Q. Xia, “Clustering of Electricity Consumption Behavior Dynamics toward Big Data Applications,” *IEEE Transactions on Smart Grid*, vol. 3053, no. c, pp. 1–1, 2016.

[76] H. Shi, M. Xu, and R. Li, “Deep Learning for Household Load Forecasting-A Novel Pooling Deep RNN,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2018.

[77] G. W. Hart, “Nonintrusive Appliance Load Monitoring,” *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[78] Y. W. Law, T. Alpcan, V. C. S. Lee, A. Lo, S. Marusic, and M. Palaniswami, “Demand response architectures and load management algorithms for energy-efficient power grids: A survey,” in *2012 Seventh International Conference on Knowledge, Information and Creativity Support Systems*, 2012, pp. 134–141.

[79] M. Baranski and J. Voss, “Detecting patterns of appliances from total load data using a dynamic programming approach,” *Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004*, pp. 327–330, 2004.

[80] L. Chuan, A. Ukil, and S. Member, “Modeling and Validation of Electrical Load Profiling in Residential Buildings in Singapore,” *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2800 – 2809, 2014.

[81] A. A. Abdelsalam, H. A. Gabbar, F. Musharavati, and S. Pokharel, “Dynamic aggregated building electricity load modeling and simulation,” *Simulation Modelling Practice and Theory*, vol. 42, no. February 2016, pp. 19–31, 2014.

[82] A. S. Bouhouras, G. T. Andreou, A. N. Milioudis, and D. P. Labridis, “Signature of residential low voltage loads,” *2012 IEEE International Conference on Industrial Technology, ICIT 2012, Proceedings*, pp. 89–94, 2012.

[83] E. B. Alzate, N. H. Mallick, and J. Xie, “A High-Resolution Smart Home Power Demand Model and Future Impact on Load Profile in Germany,” *IEEE International Conference Power & Energy (PECON)*, pp. 53–58, 2014.

[84] M. Zeifman and K. Roth, “Nonintrusive appliance load monitoring: Review and outlook,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 1, pp. 76–84, 2011.

[85] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, “Non-intrusive Load Monitoring approaches for disaggregated energy sensing: A survey,” *Sensors*, vol. 12, no. 12, pp. 16 838–16 866, 2012.

[86] J. Liao, G. Elafoudi, L. Stankovic, and V. Stankovic, “Power Disaggregation for Low-sampling Rate Data,” *NILM Work. 2014*, no. 18800, 2014.

[87] P. Ferrez and P. Roduit, “Non-intrusive appliance load curve disaggregation for service development,” *ENERGYCON 2014 - IEEE International Energy Conference*, pp. 813–820, 2014.

[88] S. Barker, M. Musthag, D. Irwin, and P. Shenoy, “Non-Intrusive Load Identification for Smart Outlets,” *SmartGridComm 2014*, pp. 548–553, 2014.

[89] M. Baranski and J. Voss, “Genetic algorithm for pattern detection in NIALM systems,” *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, no. 2, pp. 3462–3468, 2004.

[90] G. Tang, K. Wu, J. Lei, and J. Tang, “A simple model-driven approach to energy disaggregation,” *2014 IEEE International Conference on Smart Grid Communications, SmartGridComm 2014*, pp. 566–571, 2015.

[91] M. Wytock and J. Z. Kolter, “Contextually Supervised Source Separation with Application to Energy Disaggregation,” *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1–10, 2013.

[92] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han, “Unsupervised Disaggregation of Low Frequency Power Measurements,” *Proceedings of the SIAM Conference on Data Mining*, pp. 747–758, 2011.

[93] T. Zia, D. Bruckner, and A. Zaidi, “A hidden Markov model based procedure for identifying household electric loads,” *IECON Proceedings (Industrial Electronics Conference)*, pp. 3218–3223, 2011.

[94] O. Parson, S. Ghosh, M. Weal, and A. Rogers, “Non-intrusive load monitoring using prior models of general appliance types,” *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 356–362, 2012.

[95] Z. Kolter, T. Jaakkola, and J. Z. Kolter, “Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation,” *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. XX, pp. 1472–1482, 2012.

[96] S. Pattem, “Unsupervised disaggregation for non-intrusive load monitoring,” *Proceedings - 2012 11th International Conference on Machine Learning and Applications, ICMLA 2012*, vol. 2, pp. 515–520, 2012.

[97] M. J. Johnson and A. S. Willsky, “Bayesian Nonparametric Hidden Semi-Markov Models,” *The Journal of Machine Learning Research*, vol. 14, pp. 673–701, 2013.

[98] A. Marchiori, D. Hakkarinen, Q. Han, and L. Earle, “Circuit-level load monitoring for household energy management,” *IEEE Pervasive Computing*, vol. 10, no. 1, pp. 40–48, Jan. 2011.

[99] R. Jia, Y. Gao, and C. J. Spanos, “A fully unsupervised non-intrusive load monitoring framework,” *2015 IEEE International Conference on Smart Grid Communications, SmartGridComm 2015*, pp. 872–878, 2016.

[100] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake, “Leveraging smart meter data to recognize home appliances,” *2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012*, pp. 190–197, 2012.

[101] Y. Kim, T. Schmid, Z. M. Charbiwala, and M. B. Srivastava, "ViridiScope," *Proceedings of the 11th international conference on Ubiquitous computing - Ubicomp '09*, no. January, p. 245, 2009.

[102] A. G. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. P. O'Hare, "Real-Time Recognition and Profiling of Appliances through a Single Electricity Sensor," *BT - Proceedings of the Seventh Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2010*, pp. 279–287, 2010.

[103] D. Bergman, D. Jin, J. Juen, N. Tanaka, C. Gunter, and A. Wright, "Nonintrusive load-shed verification," *IEEE Pervasive Computing*, vol. 10, no. 1, pp. 49–57, 2011.

[104] P. Ducange, F. Marcelloni, and M. Antonelli, "A novel approach based on finite-state machines with fuzzy transitions for nonintrusive home appliance monitoring," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1185–1197, 2014.

[105] R. Kohlenberg, T. Phillips, and W. Proctor, "A behavioral analysis of peaking in residential electrical-energy consumers1," *Journal of Applied Behavior Analysis - J APPL BEHAV ANAL*, vol. 9, pp. 13–18, 04 1976.

[106] Z. Wang and G. Zheng, "The application of mean-shift cluster in residential appliance identification," *Proceedings of the 30th Chinese Control Conference*, pp. 3111–3114, 2011.

[107] V. Hautamäki, S. Cherednichenko, I. Kärkkäinen, T. Kinnunen, and P. Fränti, *Improving K-Means by Outlier Removal.* Springer Berlin Heidelberg, 2005, pp. 978–987.

[108] S. K. K. Ng, J. Liang, and J. W. M. Cheng, "Automatic appliance load signature identification by statistical clustering," *Advances in Power System Control, Operation and Management (APSCOM 2009)*, pp. 1–6, 2009.

[109] M. Dong, P. C. M. Meira, W. Xu, and C. Y. Chung, "Non-intrusive signature extraction for major residential loads," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1421–1430, 2013.

[110] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[111] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus External cluster validation indexes," *International Journal of Computers and Communications*, vol. 5, no. 1, pp. 27—-34, 2011.

[112] J. Z. Kolter and M. J. Johnson, "REDD : A Public Data Set for Energy Disaggregation Research," *Proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, no. 1, pp. 1–6, 2011.

[113] A. Monacchi, D. Egarter, W. Elmenreich, S. D'Alessandro, and A. M. Tonello, "GREEND: an energy consumption dataset of households in italy and austria," *SmartGridComm 2014*.

[114] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, "BLUED: a fully labeled public dataset for Event-Based Non-Intrusive load monitoring research," *Proceedings of the 2nd Workshop on Data Mining Applications in Sustainability*, Aug. 2012.

[115] W. Kleiminger, T. Staake, and S. Santini, "Occupancy Detection from Electricity Consumption Data," *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pp. 10:1–10:8, 2013.

[116] O. Van Cutsem, "OpenEMS: a python open-source energy management system platform for smart-buildings," https://gitlab.com/olivancu/OpenEMS/tree/develop, 2017.

[117] ——, "EMS strategies for OpenEMS tools," https://c4science.ch/diffusion/5154/ems-strategies.git, 2018.

[118] P. Palensky and D. Dietrich, "Demand side management: Demand response, intelligent energy systems, and smart loads," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[119] A. I. Dounis and C. Caraiscos, "Advanced control systems engineering for energy and comfort management in a building environment — A review," *Renewable and Sustainable Energy Reviews*, vol. 13, pp. 1246–1261, 2009.

[120] A. Barbato and A. Capone, "Optimization models and methods for demand-side management of residential users: A survey," *Energies*, vol. 7, no. 9, pp. 5787–5824, 2014.

[121] B. Paris, J. Eynard, S. Grieu, T. Talbert, and M. Polit, "Heating control schemes for energy management in buildings," *Energy & Buildings*, vol. 42, no. 10, pp. 1908–1917, 2010.

[122] J. S. Vardakas, N. Zorba, and C. V. Verikoukis, "A survey on demand response programs in smart grids: Pricing methods and optimization algorithms," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 152–178, 2015.

[123] J. Clarke, J. Cockroft, S. Conner, J. Hand, N. Kelly, R. Moore, T. O'Brien, and P. Strachan, "Simulation-assisted control in building energy management systems," *Energy and Buildings*, vol. 34, no. 9, pp. 933–940, 2002.

[124] S. Caron and G. Kesidis, "Incentive-based Energy Consumption Scheduling Algorithms for the Smart Grid," *Smart Grid Communications (SmartGridComm)*, pp. 391–396, 2010.

[125] H. Shakouri G. and K. Aliyeh, "Multi-objective cost-load optimization for demand side management of a residential area in smart grids," *Sustainable Cities and Society*, vol. 32, pp. 171–180, 2017.

[126] G. Xiong, C. Chen, S. Kishore, and A. Yener, "Smart (In-home) Power Scheduling for Demand Response on the Smart Grid," *ISGT 2011*, pp. 1–7, 2011.

[127] A. H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320–331, 2010.

[128] N. Kumaraguruparan, H. Sivaramakrishnan, and S. S. Sapatnekar, "Residential task scheduling under dynamic pricing using the multiple knapsack method," in *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*, 2012, pp. 1–6.

[129] P. Du and N. Lu, "Appliance Commitment for Household Load Scheduling," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 411–419, 2011.

[130] X. Chen, T. Wei, and S. Hu, "Uncertainty-aware household appliance scheduling considering dynamic electricity pricing in smart home," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 932–941, 2013.

[131] C. Vivekananthan, Y. Mishra, and F. Li, "Real-Time Price Based Home Energy Management Scheduler," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 2149–2159, 2015.

[132] Z. Chen, S. Member, L. Wu, and Y. Fu, "Real-Time Price-Based Demand Response Management for Residential Appliances via Stochastic Optimization and Robust Optimization," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1822–1831, 2012.

[133] S. Althaher, P. Mancarella, and J. Mutale, "Automated Demand Response From Home Energy Management System Under Dynamic Pricing and Power and Comfort Constraints," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1874–1883, 2015.

[134] N. Gudi, L. Wang, and V. Devabhaktuni, "A demand side management based simulation platform incorporating heuristic optimization for management of household appliances," *International Journal of Electrical Power and Energy Systems*, vol. 43, no. 1, pp. 185–193, 2012.

[135] A. Afram and F. Janabi-Sharifi, "Theory and applications of HVAC control systems - A review of model predictive control (MPC)," *Building and Environment*, vol. 72, pp. 343–355, 2014.

[136] W. J. Cole, K. M. Powell, E. T. Hale, and T. F. Edgar, "Reduced-order residential home modeling for model predictive control," *Energy and Buildings*, vol. 74, pp. 69–77, 2014.

[137] P. Radecki and B. Hencey, "Online Model Estimation for Predictive Thermal Control of Buildings," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1414–1422, 2016.

[138] D. Sturzenegger, D. Gyalistras, V. Semeraro, M. Morari, and R. S. Smith, "BRCM Matlab Toolbox : Model Generation for Model Predictive Building Control," *2014 American Control Conference*, pp. 1063–1069, 2014.

[139] A. Mirakhorli and B. Dong, "Occupancy behavior based model predictive control for building indoor climate—A critical review," *Energy and Buildings*, vol. 129, no. November 2017, pp. 499–513, 2016.

[140] C. Finck, R. Li, and W. Zeiler, "Economic model predictive control for demand flexibility of a residential building," *Energy*, vol. 176, pp. 365–379, 2019.

[141] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, no. October 2017, pp. 199–216, 2018.

[142] F. A. Qureshi, T. T. Gorecki, and C. N. Jones, "Model predictive control for market-based demand response participation," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 19, pp. 11 153–11 158, 2014.

[143] "Model predictive control approach to online computation of demand-side flexibility of commercial buildings HVAC systems for Supply Following," *Proceedings of the American Control Conference*, pp. 1082–1089, 2014.

[144] M. D. Knudsen and S. Rotger-griful, "Combined Price and Event-based Demand Response Using Two-stage Model Predictive Control," *SmartGridComm 2015*, pp. 344–349, 2015.

[145] G. T. Costanzo, O. Gehrke, D. E. M. Bondy, F. Sossan, H. Bindner, J. Parvizi, and H. Madsen, "A coordination scheme for distributed model predictive control: Integration of flexible DERs," *2013 4th IEEE/PES Innovative Smart Grid Technologies Europe, ISGT Europe 2013*, 2013.

[146] L. A. Hurtado, P. H. Nguyen, and W. L. Kling, "Agent-based control for building energy management in the smart grid framework," *IEEE PES Innovative Smart Grid Technologies Conference Europe*, vol. 2015-January, no. January, pp. 1–6, 2015.

[147] B. Zhou, W. Li, K. W. Chan, Y. Cao, Y. Kuang, X. Liu, and X. Wang, "Smart home energy management systems: Concept, configurations, and scheduling strategies," *Renewable and Sustainable Energy Reviews*, vol. 61, pp. 30–40, 2016.

[148] S. R. Griful, U. Welling, and R. H. Jacobsen, "Multi-modal Building Energy Management System for Residential Demand Response," *2016 Euromicro Conference on Digital System Design (DSD)*, pp. 252–259, 2016.

[149] D.-m. Han and J.-h. Lim, "Design and Implementation of Smart Home Energy Management Systems based on ZigBee," *Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1417–1425, 2010.

[150] G. Bazydło and S. Wermiński, “Demand side management through home area network systems,” *International Journal of Electrical Power and Energy Systems*, vol. 97, no. October 2017, pp. 174–185, 2018.

[151] M. Pritoni and D. M. Auslander, *Operating Systems for Small/Medium Commercial Buildings*. Springer International Publishing, 2018, pp. 45–69.

[152] S. Katipamula, J. Haack, G. Hernandez, B. Akyol, and J. Hagerman, “Volttron: An open-source software platform of the future,” *IEEE Electrification Magazine*, vol. 4, pp. 15–22, 12 2016.

[153] D. Blum and M. Wetter, “MPCPy: An Open-Source Software Platform for Model Predictive Control in Buildings,” in *Proceedings of the 15th IBPSA*, 2017, pp. 1381–1390.

[154] B. P. Esther and K. S. Kumar, “A survey on residential Demand Side Management architecture, approaches, optimization models and methods,” *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 342–351, 2016.

[155] Mehdi Maasoumy and A. Sangiovanni-Vincentelli, *Smart Connected Buildings Design Automation: Foundations and Trends Mehdi*, 2016, vol. 8, no. 1-2.

[156] P. Palensky, S. Member, D. Dietrich, and S. Member, “Demand Side Management : Demand Response , Intelligent Energy Systems , and Smart Loads,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[157] P. Warren, “A review of demand-side management policy in the UK,” *Renewable and Sustainable Energy Reviews*, vol. 29, pp. 941–951, 2014.

[158] H. Hao, B. M. Sanandaji, K. Poolla, and T. L. Vincent, “A generalized battery model of a collection of Thermostatically Controlled Loads for providing ancillary service,” *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton 2013*, pp. 551–558, 2013.

[159] G. Ghatikar and R. Bienert, “Smart Grid Standards and Systems Interoperability : A Precedent with OpenADR,” *Grid-Interop 2011*, pp. 1–7, 2011.

[160] *Big-M method*. Boston, MA: Springer US, 2001, pp. 62–62. [Online]. Available: https://doi.org/10.1007/1-4020-0611-X_72

[161] L. Gurobi Optimization, “Gurobi optimizer reference manual,” http://www.gurobi.com, 2019.

[162] Epex Spot SE, “Swiss day ahead auction market data,” https://www.epexspot.com/en/market-data/dayaheadauction, 2019.

[163] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Berges, D. Culler, R. Gupta, M. B. Kjærgaard, M. Srivastava, and K. Whitehouse, “Brick: Towards a unified metadata schema for buildings,” *Proceedings*

of the 3rd ACM Conference on Systems for Energy-Efficient Built Environments, BuildSys 2016, pp. 41–50, 2016.

[164] O. Van Cutsem, D. Blum, M. Pritoni, and M. Kayal, “Comparison of MPC Formulations for Building Control under Commercial Time-of-Use Tariffs,” in *PowerTech 2019*, 2019, pp. 1–6.

[165] M. J. Michael, R. Matthew, R. Arthur, S. Maxine, and W. Steven, “Report of the building energy efficiency subcommittee, to the secretary of energy advisory board,” https://www.energy.gov/eere/buildings/building-performance-database, 2012.

[166] Office of Energy Efficiency & Renewable Energy, “About the commercial buildings integration program,” https://www.energy.gov/eere/buildings/about-commercial-buildings-integration-program, 2012.

[167] S. Nolan and M. O’Malley, “Challenges and barriers to demand response deployment and evaluation,” *Applied Energy*, vol. 152, pp. 1–10, 2015.

[168] U.S. Department of Energy, “Benefits of Demand Response in Electricity Markets and Recommendations for Achieving Them,” 2006.

[169] D. Kim and J. E. Braun, “Development, implementation and performance of a model predictive controller for packaged air conditioners in small and medium-sized commercial building applications,” *Energy and Buildings*, vol. 178, pp. 49–60, 2018.

[170] M. Jingran, S. J. Qin, L. Bo, T. Salsbury, J. Ma, and B. Li, “Economic model predictive control for building energy systems,” *2011 IEEE PES Innovative Smart Grid Technologies*, 2011.

[171] T. Salsbury, P. Mhaskar, and S. J. Qin, “Predictive control methods to improve energy efficiency and reduce demand in buildings,” *Computers and Chemical Engineering*, vol. 51, pp. 77–85, 2013.

[172] Z. Wang, A. Babak, and S. Ratnesh, “Stochastic Demand Charge Management for Commercial and Industrial Buildings,” *2017 IEEE Power & Energy Society General Meeting*, 2017.

[173] H. Hao, D. Wu, J. Lian, and T. Yang, “Optimal Coordination of Building Loads and Energy Storage for Power Grid and End User Services,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4335–4345, 2018.

[174] H. Huang and C. Roussac, “Predicting peak energy demand in commercial buildings under extreme conditions : by how much can we improve accuracy ?” *2018 ACEEE Summer Study on Energy Efficiency in Buildings*, pp. 1–12, 2018.

[175] S. H. Lee and T. Hong, “Leveraging zone air temperature data to improve physics-based energy simulation of existing buildings,” *15th IBPSA Conference*, pp. 528–535, 2017.

[176] U.S. Department of Energy, "Commercial reference buildings," https://www.energy.gov/eere/buildings/commercial-reference-buildings, accessed: 2018-10-30.

[177] National Renewable Energy Laboratory, "National solar radiation database," https://rredc.nrel.gov/solar/old_data/nsrdb/.

[178] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *European Control Conference (ECC)*, 2013, pp. 3071–3076.

[179] O. Van Cutsem, D. Ho Dac, P. Boudou, and M. Kayal, "Cooperative Energy Management of a Community of Smart-Buildings: a Blockchain approach," *International Journal of Electrical Power & Energy Systems*, vol. 117, p. 105643, 2020.

[180] R. Ma, H. H. Chen, Y. R. Huang, and W. Meng, "Smart grid communication: Its challenges and opportunities," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 36–46, 2013.

[181] A. H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 120–133, 2010.

[182] J. S. Vardakas, N. Zorba, C. V. Verikoukis, and S. Member, "A Survey on Demand Response Programs in Smart Grids : Pricing Methods and Optimization Algorithms," *IEEE COMMUNICATION SURVEYS & TUTORIALS*, vol. 17, no. 1, pp. 152–178, 2015.

[183] S. Kishore and L. V. Snyder, "Control Mechanisms for Residential Electricity Demand in SmartGrids," *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pp. 443–448, 2010.

[184] D. S. Callaway and I. A. Hiskens, "Achieving Controllability of Electric Loads," *Proceedings of the IEEE*, vol. 99, no. 1, 2010.

[185] T. Wei, S. Member, Q. Zhu, and N. Yu, "Proactive Demand Participation of Smart Buildings in Smart Grid," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1392–1406, 2016.

[186] J. Iria, F. Soares, and M. Matos, "Optimal supply and demand bidding strategy for an aggregator of small prosumers," *Applied Energy*, vol. 213, no. June 2017, pp. 658–669, 2018.

[187] X. Ayón, J. K. Gruber, B. P. Hayes, J. Usaola, and M. Prodanović, "An optimal day-ahead load scheduling approach based on the flexibility of aggregate demands," *Applied Energy*, vol. 198, pp. 1–11, 2017.

[188] W. Saad, Z. Han, H. V. Poor, and T. Başar, "Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 86–105, 2012.

[189] Y. Zhang and G. Augenbroe, “Optimal demand charge reduction for commercial buildings through a combination of efficiency and flexibility measures,” *Applied Energy*, vol. 221, no. March, pp. 180–194, 2018.

[190] M. Pilz and L. Al-Fagih, “Recent Advances in Local Energy Trading in the Smart Grid Based on Game–Theoretic Approaches,” *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 1363–1371, 2017.

[191] P. Chakraborty, E. Baeyens, P. P. Khargonekar, K. Poolla, and P. Varaiya, “Analysis of Solar Energy Aggregation under Various Billing Mechanisms,” *IEEE Transactions on Smart Grid*, 2018.

[192] N. Liu, X. Yu, C. Wang, C. Li, L. Ma, and ..., “Energy-sharing model with price-based demand response for microgrids of peer-to-peer prosumers,” *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3569–3583, 2017.

[193] B. Cornélusse, I. Savelli, S. Paoletti, A. Giannitrapani, and A. Vicino, “A community microgrid architecture with an internal local market,” *Applied Energy*, vol. 242, no. March, pp. 547–560, 2019.

[194] S. Nakamoto, “Bitcoin : A Peer-to-Peer Electronic Cash System,” *Cryptography Mailing list at https://metzdowd.com*, pp. 1–9, 2009.

[195] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. Mccallum, and A. Peacock, “Blockchain technology in the energy sector : A systematic review of challenges and opportunities,” *Renewable and Sustainable Energy Reviews*, 2019.

[196] Ethereum Foundation, “Ethereum’s white paper,” https://github.com/ethereum/wiki/wiki/White-Paper, 2014.

[197] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, and L. Orsini, “Designing microgrid energy markets A case study : The Brooklyn Microgrid,” *Applied Energy*, vol. 210, pp. 870–880, 2018.

[198] A. Hahn, R. Singh, C. C. Liu, and S. Chen, “Smart contract-based campus demonstration of decentralized transactive energy auctions,” *2017 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference, ISGT 2017*, 2017.

[199] D. Cutler, T. Kwasnik, S. Balamurugan, S. Booth, and B. S. National, “A Demonstration of Blockchain-based Energy Transactions between Laboratory Test Homes,” *2018 ACEEE Summer Study on Energy Efficiency in Buildings*, pp. 1–13, 2018.

[200] M. Pipattanasomporn, M. Kuzlu, and S. Rahman, “A Blockchain-based Platform for Exchange of Solar Energy: Laboratory-scale Implementation,” *Proceedings of the Conference on the Industrial and Commercial Use of Energy, ICUE*, vol. 2018-October, no. October, pp. 1–9, 2019.

[201] N. Z. Aitzhan and D. Svetinovic, “Security and Privacy in Decentralized Energy Trading Through Multi-Signatures , Blockchain and Anonymous Messaging Streams,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018.

[202] M. Meisel, L. Fotiadis, S. Wilker, A. Treytl, and T. Sauter, “Blockchain Applications In Microgrids: an overview of current projects and concepts,” *43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 6153–6158, 2017.

[203] E. Munsing, J. Mather, and S. Moura, “Blockchains for decentralized optimization of energy resources in microgrid networks,” in *1st Annual IEEE Conference on Control Technology and Applications*, 2017.

[204] S. Noor, W. Yang, M. Guo, K. H. van Dam, and X. Wang, “Energy Demand Side Management within micro-grid networks enhanced by blockchain,” *Applied Energy*, 2018.

[205] C. Pop, T. Cioara, M. Antal, I. Anghel, I. Salomie, and M. Bertoncini, “Blockchain Based Decentralized Management of Demand Response Programs in Smart Energy Grids,” *Sensors*, vol. 18, 2018.

[206] M. Gupta, *Blockchain for dummies.* John Wiley & Sons, 2017.

[207] S. Vergura, “A complete and simplified datasheet-based model of pv cells in variable environmental conditions for circuit simulation,” *Energies*, vol. 9, no. 5, 2016.

[208] web3.js, “Ethereum javascript api,” https://web3js.readthedocs.io/en/1.0/, 2019.

[209] Ethereum Javascript Community, “Ethereumjs,” https://ethereumjs.github.io/, 2018.

[210] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[211] C. Dannen, *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*, 1st ed. Berkely, CA, USA: Apress, 2017.

[212] KUL Energy Institute, “The current electricity market design in Europe,” *EI-FACT SHEET*, p. 4, 2015.

[213] F. Elghitani and W. Zhuang, “Aggregating a Large Number of Residential Appliances for Demand Response Applications,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5092–5100, 2018.

[214] California Public Utilities Commission, “Energy Division’s Evaluation of Demand Response Auction Mechanism,” Tech. Rep., 2019.

[215] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Cañizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke, G. A. Jiménez-Estévez, and N. D. Hatziargyriou, “Trends in microgrid control,” *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1905–1919, 2014.

[216] C. Wang, J. Yan, C. Marnay, N. Djilali, E. Dahlquist, J. Wu, and H. Jia, “Distributed Energy and Microgrids (DEM),” *Applied Energy*, vol. 210, pp. 685–689, 2018.

[217] T. Samad, E. Koch, and P. Stluka, “Automated Demand Response for Smart Buildings and Microgrids: The State of the Practice and Research Challenges,” *Proceedings of the IEEE*, vol. 104, no. 4, pp. 726–744, 2016.

[218] S. Kiliccote, M. A. Piette, and L. Berkeley, “Buildings-to-Grid Technical Opportunities: From the Buildings Perspective 1,” Tech. Rep., 2014.

[219] J. Hagerman, “Buildings-to-Grid Technical Opportunities: Introduction and Vision,” Tech. Rep., 2011.

[220] A. F. Taha, N. Gatsis, B. Dong, A. Pipri, and Z. Li, “Buildings-to-Grid Integration Framework,” *IEEE Transactions on Smart Grid*, vol. XX, no. OCTOBER, pp. 1–13, 2017.

[221] A. Ulbig, S. Member, and G. Andersson, “On Operational Flexibility in Power Systems,” *IEEE Power and Energy Society General Meeting*, p. 8, 2012.

[222] J. T. Huges, A. D. Domínguez-García, and K. Poolla, “Virtual battery models for load flexibility from commercial buildings,” *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2015-March, pp. 2627–2635, 2015.

[223] J. T. Hughes, A. D. Domínguez-García, and K. Poolla, “Identification of Virtual Battery Models for Flexible Loads,” *IEEE Transactions on Power Systems*, vol. 31, no. 6, pp. 4660–4669, 2016.

[224] D. H. Blum, T. Zakula, and L. K. Norford, “Opportunity cost quantification for ancillary services provided by heating, ventilating, and air-conditioning systems,” *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1264–1273, 2017.

[225] R. De Coninck and L. Helsen, “Quantification of flexibility in buildings by cost curves - Methodology and application,” *Applied Energy*, vol. 162, pp. 653–665, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2015.10.114

[226] M. Kohansal and H. Mohsenian-Rad, “A Closer Look at Demand Bids in California ISO Energy Market,” *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 3330–3331, 2016.

[227] B. Kouvaritakis and M. Cannon, “Stochastic Model Predictive Control,” *Encyclopedia of Systems and Control*, no. December, pp. 1–9, 2014.

[228] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, “On-line Building Energy Optimization using Deep Reinforcement Learning,” *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3698–3708, 2018.

[229] Department of Energy, “BOPTEST: Building Operations Testing Framework,” https://www.energy.gov/eere/buildings/boptest-building-operations-testing-framework, 2019.

[230] J. Chapman, P.-O. Siebers, and D. Robinson, “On The Multi Agent Stochastic Simulation Of Occupants In Buildings,” *Journal of Building Performance Simulation*, vol. 11, no. 5, pp. 604–621, 2018.

[231] Heat Web, “Underfloor Heating,” 2011. [Online]. Available: http://www.heatweb.com/techtips/Underfloor/underfloorheating.html

[232] Conférence des directeurs cantonaux de l’énergie, “Modèle de prescriptions énergétiques des cantons (MoPEC),” 2014. [Online]. Available: https://www.endk.ch/fr/documentation/batiments-mopec

[233] The Green Home, “What is the best flooring for underfloor heating?” 2013. [Online]. Available: http://thegreenhome.co.uk/heating-renewables/underfloor-heating/best-flooring-for-underfloor-heating/

[234] “Air to water Heat Pump Kita M,” https://www.templari.com/en/air-to-water-heat-pump-kita-m/, accessed: 2019-08-14.

[235] “Automatic Control Laboratory - EPFL,” https://www.epfl.ch/labs/la/, accessed: 2019-08-12.

[236] D. Carter, J. Zoellick, and M. Marshall, “Demonstrating a Secure, Reliable, Low-Carbon Community Microgrid at the Blue Lake Rancheria,” Tech. Rep. January, 2019.

[237] “eXtensible Building Operating System,” https://docs.xbos.io/, accessed: 2019-07-25.

Olivier Van Cutsem

Electrical Engineer

Personal data

Adress	Avenue de Montoie 37, 1007 Lausanne, Switzerland
Date of birth	25 February 1992
Nationality	Belgium (Swiss resident, Permit B)
Contact	+41 78 949 27 09, vancutsemolivier@gmail.com

Education

Sept. 2015 - *Present*	**PhD student**, *Ecole Polytechnique Fédérale de Lausanne (EPFL)*, Switzerland, supervisor: Pr. M. KAYAL. On Smart-Buildings and their integration into the Smart-Grid
May - Oct. 2018	**Visiting researcher**, *Lawrence Berkeley National Laboratory (LBNL)*, CA, US, supervisor: Rich Brown. Demand Response methods analysis and implementation with OpenADR for Solar+ Optimizer project
Spring 2015	**Master Thesis**, *Ecole Polytechnique Fédérale de Lausanne (EPFL)*, Switzerland. Design and implementation of a local network based on Power Line Communication for Smart-Buildings
2013 - 2015	**Master in Electrical Engineering**, *Université de Liège (ULg)*, Belgium. Specialized in Electricity and Electronics Science (Graduated *Magna Cum Laude*)
2010 - 2013	**Bachelor in Electrical Engineering**, *Université de Liège (ULg)*, Belgium. Specialized in Electricity, Electronics and Computer Science (Graduated *Magna Cum Laude*)
2004 - 2010	**Secondary Education Degree**, *Athenée Royal Air Pur*, Seraing, Belgium. Option Mathematics/Sciences

Experience

Jan. 2018 - *Present*	**Back-end Application Developer**, *Trokus*, Belgium, *www.trokus.be*. Back-end developer for the local food exchange platform Trokus
2015 - 2018	**Student projects supervisor**, EPFL. Conception and management of semester projects tailored to master students in Energy Management and Sustainability
2015 - 2018	**Student assistant**, EPFL, course: *Electronics II*, Prof M. Kayal & Dr. Koukab. Teaching exercises & laboratories for bachelor students in electronics
Summer 2014	**Company internship**, *Deltatec*, Belgium. Development of a video scaler using Vivado High Level Synthesis (HLS) for Virtex 6
2013 - 2016	**Private teacher**. Helping students preparing engineering admission exams and bachelor level teaching
2013 - 2014	**Student assistant**, ULg, courses: Electrical Engineering and Computer Science. Supervising laboratories for third-year students in Engineering Teaching basics of C programming to first-year students in Engineering

Computer skills

Basic	VHDL, Web design, Linux, Microcontrollers
Intermediate	C++, JAVA, Word/Excel/LaTeX, Jupyter/Pandas, Optimization solvers, Mobile App
Advanced	Python, C, Matlab

Languages

Mother tongue	**French**.
Advanced	**English**, *Highly proficient in spoken and written English.*
Good	**Spanish**, *Advanced understanding and good spoken/written skills.*
Basic	**Dutch, German**, *Basic understanding.*

Miscellaneous

Sports	Road cycling, trail running, racket sports, skiing
Hobbies	Travelling & culture discovering, green energy technology, music concerts, cinema & reading

Publications

Conference 2016	G. Lilis, O. Van Cutsem, and M. Kayal, "Building virtualization engine: a novel approach based on discrete event simulation," *2nd International Conference on Event-Based Control, Communication, and Signal Processing*, 2016.
Journal 2018	G. Lilis*, O. Van Cutsem*, and M. Kayal, "A high-speed integrated building emulation engine based on discrete event simulation," *Journal of Systems Architecture*, vol. 92, pp. 53-65, 2018 **the authors contributed equally.*
Conference 2018	O. Van Cutsem, G. Lilis, and M. Kayal, "Automatic multi-state load profile identification with application to energy disaggregation," *IEEE International Conference on Emerging Technologies and Factory Automation*, 2018.
Conference 2019	O. Van Cutsem, D. Blum, M. Pritoni, and M. Kayal, "Comparison of MPC formulations for building control under commercial time-of-use tariffs," *PowerTech 2019*.
Conference 2019	I. Bayram , O. Van Cutsem, J. Bigler, J-C. Fosse, & M. Kayal, "Demonstration of a smart villa energy monitoring platform in qatar," *UCET'19*, 2019.
Journal 2019	O. Van Cutsem, D. Ho Dac, P. Boudou, and M. Kayal, "Cooperative energy management of a community of smart-buildings: a blockchain approach," *International Journal of Electrical Power & Energy Systems*, pp 1-13, 2019, in production.

List of supervised projects

Sep 2015 - Jun 2016	"Smart Building Interfaces", *Semester project*, D. Davydov "Easy Dashboard for Smart-Buildings", *Semester project*, A. Bouchet "Smart Grid - Smart Building Interaction", *Semester project*, M. Dorokhova
Sep 2016 - Jun 2017	"Energy Comparison Tool", *Semester project*, T. Wyss and C. Raposo "User occupancy prediction using non-intrusive power consumption data", *Semester project*, O. Guessous "Building Simulation", *Semester project*, R. Glatt
Sep 2017 - Jun 2018	"MPC for Smart-Building", *Semester project*, N. Sierro "Decentralized Demand Response and Microgrids: A Blockchain approach", *Semester project*, D. Ho Dac
Sep 2018 - Aug 2019	"A Decentralized Algorithm for DR and RES with Blockchain technology", *Semester project*, P. Boudou "Energy Management for Smart Homes using Renewable Energy Sources", *Master thesis*, B. Ismaili "Assessing the potential CO2 reduction for the deployment of an Ecoblock in Oakland", *Master thesis*, M. Gaillet-Tournier