

Fusing Pre-existing Knowledge and Machine Learning for Enhanced Building Thermal Modeling and Control

Présentée le 23 février 2024

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'automatique 3
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Loris DI NATALE

Acceptée sur proposition du jury

Prof. A. Karimi, président du jury
Prof. C. N. Jones, Dr B. Svetozarevic, directeurs de thèse
Prof. Z. Nagy, rapporteur
Dr J. Drgoňa, rapporteur
Prof. O. Fink, rapporteuse

*“There’s no sense in being precise when you
don’t even know what you’re talking about.”*

— John von Neumann

Acknowledgements

First, I would like to express my utmost gratitude to Prof. Colin Jones, Dr. Bratislav Svetozarevic, and Philipp Heer, who jointly offered me the opportunity to pursue this Ph.D. between the Urban Energy Systems Lab at Empa and the Laoratoire d'Automatique at EPFL. I cannot thank you enough for your flexibility and seamless collaboration over the years, which allowed me to explore many different topics. It was always a pleasure to meet with you and engage in fascinating deep discussions throughout the years, often extending work.

During the last one and a half years, I also had the privilege of working closely with Prof. Giancarlo Ferrari Trecate and Muhammad Zakwan. It opened me to a whole new way of doing research, and I am incredibly grateful for everything I learned with you. It truly felt like I was part of your group.

Whether at Empa, EPFL, NCCR events, or conferences, the positive working atmosphere always made me feel at home, rendering this four-year-long journey amazing. I can confidently say that most of my “work colleagues” have become genuine friends, and I will leave with countless cherished memories. A special thanks to Anna, Clara, Daniele, Hanmin, JS, Luca, Mahrokh, Nathan, Paul, Roland, Tingting, Vaibhav, Varsha, Wouter, and Zak for making my stay a little bit more special.

Lastly, none of this would have been possible without the unwavering support from my friends and family, not just during this latest endeavor but also long before. Thanks a lot for giving me a break from work throughout the years. A special thanks to the *As'soiffés* for the unforgettable weekends and holidays, Louis for the most rejuvenating winter retreats one could hope for, and Christine for the everyday support.

And of course, papa, maman, Margaux, I would be nothing without you.

Lausanne, January 29, 2024

Loris Di Natale

Abstract

Buildings play a pivotal role in the ongoing worldwide energy transition, accounting for 30% of the global energy consumption. With traditional engineering solutions reaching their limits to tackle such large-scale problems, data-driven methods and Machine Learning (ML) tools are gaining momentum. In particular, Neural Networks (NNs) are becoming prominent, both for modeling tasks or as control policies in Deep Reinforcement Learning (DRL) agents.

Despite their remarkable achievements, NNs however suffer from poor generalization to *unseen* data and may fail to adhere to the fundamental laws of physics. Consequently, the first part of this thesis focuses on merging physical insights into NNs, proposing the novel Physically Consistent Neural Network (PCNN) architecture. In PCNNs, a *physics-inspired* module leveraging established domain expertise runs in parallel to a *black-box* NN to ensure the model is aligned with the principles of physics. Applying PCNNs to multi-zone building thermal modeling, we prove that they are consistent with the laws of thermodynamics *by design*, as required, while simultaneously achieving state-of-the-art modeling performance among data-driven methods on a case study.

The second part of this thesis starts by discussing the characteristics of an *ideal* building controller, identifying model-free DRL control policies as strong candidates. We then illustrate how DRL agents can not only significantly surpass baseline controllers but also achieve near-optimal performance. Finally, we propose to enforce expert-designed rules on DRL agents to avoid suboptimal decisions and accelerate learning. Collectively, these investigations on single-zone temperature case studies point toward the potential of DRL agents being deployed from scratch in buildings and autonomously acquiring near-optimal behaviors within complex environments in a reasonable amount of time, bypassing the need for engineering-heavy control solutions.

Despite their versatile capabilities, however, the *black-box* nature of NNs may not be ideal in practice. To tackle this issue, the last part of this thesis focuses on using automatic backpropagation for System Identification (SI), extending beyond building-specific contexts. We introduce SIMBa, a general-purpose SI toolbox leveraging ML tools to identify *structured linear* state-space models from data. SIMBa facilitates the seamless incorporation of prior domain expertise while simultaneously ensuring model stability and achieving impressive performance across various SI tasks from both simulated and real-world data. Finally, we present one extension of SIMBa to identify irreversible port-Hamiltonian dynamics, creating nonlinear models that inherently adhere to the laws of thermodynamics and paving the way for the identification of general structured nonlinear systems through the power of backpropagation.

Abstract

Altogether, this thesis investigates diverse strategies to merge prior knowledge and ML techniques, encompassing both the adaptation of NNs to align with underlying physics and the utilization of automatic backpropagation to extract structured models from data. Overall, our results hint at the effectiveness of merging both worlds, leveraging the large-scale capabilities of ML tools to solve complex problems while anchoring their solutions in the foundational expertise of domain-specific knowledge.

Keywords: *Machine Learning, Deep Reinforcement Learning, System Identification, Prior knowledge integration, Building thermal modeling, Building control.*

Résumé

Les bâtiments jouent un rôle essentiel dans la transition énergétique puisqu'ils représentent 30% de la consommation énergétique mondiale. Les solutions d'ingénierie traditionnelles atteignant leurs limites pour résoudre de tels problèmes de grande échelle, les méthodes basées sur les données et l'Apprentissage Machine (AM) progressent. Les Réseaux de Neurones (RNs), notamment, suscitent un intérêt grandissant, que ce soit pour des tâches de modélisation ou en tant que contrôleurs entraînés par Apprentissage par Renforcement (AR).

Malgré leurs avantages, les RNs ne sont cependant pas capables de réagir à de *nouvelles* données en général et peuvent ne pas respecter les lois de la physique. Par conséquent, la première partie de cette thèse introduit les RNs Physiquement Cohérents (RNPCs). Ils contiennent deux modules en parallèle, le premier s'assurant le respect des lois physiques et le second comprenant un RN pour s'ajuster aux données mesurées. En appliquant les RNPCs à une étude de cas de modélisation thermique de bâtiment, nous prouvons qu'ils sont cohérents avec les lois de la thermodynamique, comme requis, tout en atteignant la meilleure précision parmi les méthodes testées.

La deuxième partie de ce travail commence par aborder les caractéristiques d'un contrôleur de bâtiment *idéal*, identifiant l'AR comme un candidat intéressant. Nous illustrons ensuite comment l'AR peut non seulement surpasser significativement les contrôleurs de référence, mais aussi atteindre des performances quasi-optimales. Enfin, nous proposons d'imposer des règles conçues par des experts lors de l'apprentissage pour éviter les décisions sous-optimales et accélérer la convergence. Dans l'ensemble, les cas d'étude simplifiés analysés indiquent le potentiel de l'AR pour apprendre à des RNs à acquérir autonomement des comportements quasi-optimaux dans des environnements complexes et en un laps de temps raisonnable.

Malgré leurs capacités polyvalentes, cependant, la nature *boîte noire* des RNs peut ne pas être idéale en pratique. Pour résoudre ce problème, la dernière partie de cette thèse se concentre sur l'utilisation d'outils développés pour les RNs appliqués à l'identification de modèles physiques traditionnels, pour les bâtiments mais pas seulement. Nous présentons SIMBa, qui identifie des modèles linéaires *structurés* à partir de données grâce à ces outils. Enfin, nous proposons une extension de SIMBa pour identifier des modèles port-Hamiltonien irréversibles, modèles non linéaires qui respectent intrinsèquement les lois de la thermodynamique. Collectivement, ces investigations ouvrent la voie à l'identification de systèmes non linéaires structurés grâce aux outils développés pour les RNs.

Dans l'ensemble, cette thèse explore diverses stratégies pour fusionner les connaissances préalables et les techniques d'AM. Elle englobe à la fois l'adaptation des RNs pour les aligner

Résumé

sur la physique sous-jacente et l'utilisation d'outils développés pour les RNs pour identifier des modèles traditionnels à partir de données. Collectivement, nos résultats suggèrent l'efficacité du mélange des deux mondes, exploitant les capacités à grande échelle des outils d'AM pour résoudre des problèmes complexes tout s'assurant que les solutions respectent les connaissances des experts du domaine.

Mots-clés : *Apprentissage machine, Apprentissage par renforcement, Identification de systèmes, Intégration de connaissances expertes, Modélisation thermique de bâtiments, Contrôle de bâtiments.*

Contents

Acknowledgements	i
Abstract (English/Français)	iii
Acronyms	ix
1 Introduction	1
1.1 Decreasing the energy intensity of the building sector	1
1.2 Marrying Neural Networks and traditional methods	4
1.3 Putting everything together	9
2 Physically Consistent Neural Networks for thermal building modeling	11
2.1 The need for physically consistent Neural Networks	11
2.2 Towards prior knowledge-infused Neural Network building thermal models . .	15
2.3 Physically Consistent Neural Networks	19
2.4 Presentation of the case study	30
2.5 Performance analysis and comparison to other methods	38
2.6 Conclusion and outlook	54
3 Prospects and hurdles of Deep Reinforcement Learning for building control	61
3.1 The potential of model-free Deep Reinforcement Learning	61
3.2 Preliminaries	75
3.3 Deep Reinforcement Learning can achieve near-optimal performance	80
3.4 Constraining agents and accelerating convergence	89
3.5 Conclusion and outlook	99
4 Leveraging automatic differentiation for system identification	105
4.1 Towards structured stable linear system identification	106
4.2 Free parametrizations of Schur matrices	110
4.3 The SIMBa toolbox	114
4.4 Benchmarking SIMBa through numerical experiments	119
4.5 Discussion of SIMBa's potential and limitations	135
4.6 A nonlinear extension: Physically Consistent Neural ODEs	137
4.7 Conclusion and outlook	146

Contents

5	Concluding remarks	149
A	Appendices - Chapter 2	151
A.1	Resistance-capacitance building model	151
A.2	Proofs of the main theoretical results	153
A.3	Solar irradiation preprocessing	155
A.4	Details on the data processing	156
A.5	Linear model identification	157
A.6	Additional single-zone PCNN results	158
A.7	Visualization of predictions	160
A.8	X-PCNN gradients	160
A.9	Number of numerical gradient values	163
B	Appendices - Chapter 4	165
B.1	Proof of Proposition 3	165
B.2	Proof of Corollary 2	166
B.3	Proof of Proposition 4	167
B.4	Proof of Corollary 3	168
B.5	Proof of Proposition 5	168
B.6	Proof of Proposition 6	170
B.7	Proof of Corollary 4	170
B.8	Temperature computation	171
B.9	Proof of Proposition 7	171
	Bibliography	173
	Curriculum Vitae	191

Acronyms

AD	Automatic Differentiation
ARIMA	AutoRegressive Integrated Moving Average
ARX	AutoRegressive model with eXogenous inputs
BEV	Battery Electric Vehicle
BO	Bayesian Optimization
BPTT	BackPropagation Through Time
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DeePC	Data-enabled Predictive Control
DL	Deep Learning
DPC	Differentiable Predictive Control
DRL	Deep Reinforcement Learning
EA	Efficient Agent
GD	Gradient Descent
GPU	Graphical Processing Unit
HVAC	Heating, Ventilation, and Air Conditioning
IAQ	Indoor Air Quality
ICNN	Input Convex Neural Network
IL	Imitation Learning
IPH	Irreversible port-Hamiltonian
LMI	Linear Matrix Inequality
LP	Linear Program
LS	Least Squares
LSTM	Long Short-Term Memory
LTl	Linear Time-Invariant
MAE	Mean Absolute Error

Acronyms

MAPE	Mean Absolute Percentage Error
MBDRL	Model-Based Deep Reinforcement Learning
ML	Machine Learning
MPC	Model Predictive Control
MSE	Mean Squared Error
NODE	Neural Ordinary Differential Equations
NN	Neural Network
ODE	Ordinary Differential Equation
PARSIM	PARsimonious Subspace Identification Method
PC-NODE	Physically Consistent Neural Ordinary Differential Equation
PCNN	Physically Consistent Neural Network
PEM	Prediction Error Method
PiML	Physics-inspired Machine Learning
PiNN	Physics-inspired Neural Network
PiDRL	Physics-inspired Deep Reinforcement Learning
PV	Photovoltaic
RBC	Rule-Based Control
RC	Resistance-Capacitance
ReL	Residual Learning
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RS	Reward Shaping
SI	System Identification
SIM	Subspace Identification Method
SIMBa	System Identification Methods leveraging Backpropagation
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic
TL	Transfer Learning
UMAR	Urban Mining and Recycling

1 Introduction

In the global shift towards sustainable energy to combat climate change, the emergence of new technologies will play a pivotal role. As energy systems become increasingly interconnected and, as a result, more intricate, expert- and engineering-based methods to optimize their operations indeed reach their limits. Conversely, purely data-driven solutions, particularly those relying on Neural Networks (NNs), have gained popularity in recent years but still lack performance and safety guarantees for real-world applications. Hybrid methods fusing prior knowledge and emerging technologies are thus bound to become leaders of the transition. These approaches have the potential to address complex problems while retaining the valuable insights acquired from decades of understanding and describing physical systems.

1.1 Decreasing the energy intensity of the building sector

As of 2022, building operations accounted for 30% of the global final energy demand and 26% of the global energy-related carbon emissions¹ worldwide [1]. Notably, almost half of that energy was solely dedicated to space and water heating [2], with an additional approximately 10% contributed by space cooling.² Collectively, almost 60% of the total energy usage in buildings — equivalent to 18% of the global final energy consumption — can hence be traced back to space and water heating as well as space cooling operations, making them primary targets for energy consumption reduction investigations.

Large-scale electrification of energy systems — coupled with a phase-out of fossil-based electricity generation technologies — has been identified as one of the most promising pathways for their decarbonization [4]. Alternatively, instead of modifying the energy supply mix, one can directly intervene at the building level to decrease the associated energy demand. This typ-

¹This can be broken down into 8% directly stemming from buildings and the other 18% being indirect emissions linked to the production of electricity and heat used in buildings.

²Approximated from the fact that cooling energy demand has been steadily growing since 2018 when it was responsible for roughly 7% of the total building energy budget, i.e. one-fifth of the electricity consumption in buildings [3], which itself represents 35% of the total building energy consumption [1].

ically involves the construction of more efficient buildings and appliances [5], the retrofit of old edifices [6], or the introduction of advanced control methods in existing infrastructures [7].³ This thesis focuses on the last option, aiming to provide **insights into data-driven methods to decrease the energy intensity of existing buildings through smart control algorithms**.

1.1.1 The need for advanced building control algorithms

While prior research has demonstrated the potential for substantial energy savings by adjusting building temperature setpoints [8], such considerations always have to be weighed against the comfort of the occupants. Indeed, it is imperative to maintain indoor temperatures within an acceptable range, as extremes can lead to discomfort [9], and energy minimization should not come at the expense of occupant well-being [10]. Remarkably, each person might perceive thermal comfort differently, resulting in personalized preferences [11]. To make matters worse, both objectives are usually conflicting, with higher levels of thermal comfort often coming at the price of additional energy consumption — and the relationship between the amount of energy used and the subsequent comfort of the occupants is complex, giving rise to highly nontrivial trade-offs.

Beyond the preferences and thermal comfort requirements of the occupants rendering the building energy optimization problem challenging in general, their behavior also directly influences the amount of energy needed to maintain satisfactory indoor conditions. In commercial buildings, different occupancy patterns can lead to energy consumption variations from 30% to 150%, for example [12]. On top of that, buildings can be significantly impacted by external weather conditions, especially if not well insulated. Adding to the complexity, each building is unique and hence requires a tailored controller — unlike industrial processes, for example, where the same solution can be applied repeatedly once it has been optimized.

Altogether, this calls for control solutions able to minimize building energy consumption without compromising occupant comfort, regardless of the circumstances or the specific characteristics of the building, a very challenging control problem in general [13].

1.1.2 The rise of data-driven methods

Despite the known advantages of advanced control methods, the building automation industry still mainly relies on Rule-Based Controllers (RBCs) [7, 14]. However, RBCs are *reactive* controllers, i.e., they cannot anticipate environmental changes, and hence generally perform suboptimally [15]. Furthermore, manually tuning them to achieve good performance — or *retuning* them when the operating conditions change — is highly time-consuming [16].

³Note that advanced control methods will also be required to maximize the utility of private Photovoltaic (PV) electricity production and subsequently decrease the electricity demand of buildings.

A model-based paradigm

These shortcomings of RBCs can be addressed with *proactive* control methods, often leveraging Model Predictive Control (MPC), which can simultaneously achieve impressive energy savings and thermal comfort improvement over standard baselines [17, 18]. MPC relies on a model of the building under control and disturbance predictions to anticipate environmental changes and find optimal control inputs [7]. More recently, high-fidelity models have also been used as *simulators* to train (Deep) Reinforcement Learning ((D)RL) agents — which learn via trial and error [19] — before deploying them in physical buildings [14]. Overall, accurate building thermal models are thus nowadays of paramount importance in building control applications.

Constructing a model from first principles and calibrating its parameters to achieve good performance is, however, a time-consuming and engineering-heavy endeavor [20, 21]. To alleviate this workload, data-driven modeling methods gained considerable momentum in the past years, taking advantage of the increasing amount of data collected in buildings [22]. Traditionally, one uses data to identify the parameters of simplified physics-based models, leading to the *gray-box* modeling paradigm. Nevertheless, this parameter identification process is generally nontrivial [23] and yields a trade-off between model complexity and accuracy [24].

Alternatively, statistical patterns can directly be derived from data using Machine Learning (ML) tools, adopting a *black-box* modeling approach and bypassing the need for engineering altogether [25]. Following their successes in a wide variety of tasks [26], NNs have recently been applied to thermal building modeling as well [25]. Although they achieve impressive performance, however, vanilla NNs are completely physics-agnostic, which can lead to spurious behaviors in practice [27]. Further work is hence required to design black-box models consistent with known system properties, typically stemming from the underlying physical laws, before such data-driven paradigms achieve widespread adoption in practical applications.

A model-free vision

Despite the accomplishments of model-based building control methods and the latest advances in data-driven modeling,⁴ we argue in Chapter 3 that *model-free* DRL algorithms provide a valid alternative with interesting potential for widespread adoption. They could indeed bypass the need for accurate models altogether, avoiding the associated pitfalls and paving the way for building-agnostic yet well-performing control solutions. Furthermore, they are ideal candidates for controllers being deployed *from scratch* in buildings, potentially removing the need for engineering throughout the entire process.

However, because of the slow thermal dynamics of buildings⁵ and the high sample com-

⁴This includes our investigations in Chapters 2 and 4.

⁵A new control input is usually applied every 10–15 min due to the high time constant of thermal dynamics [28].

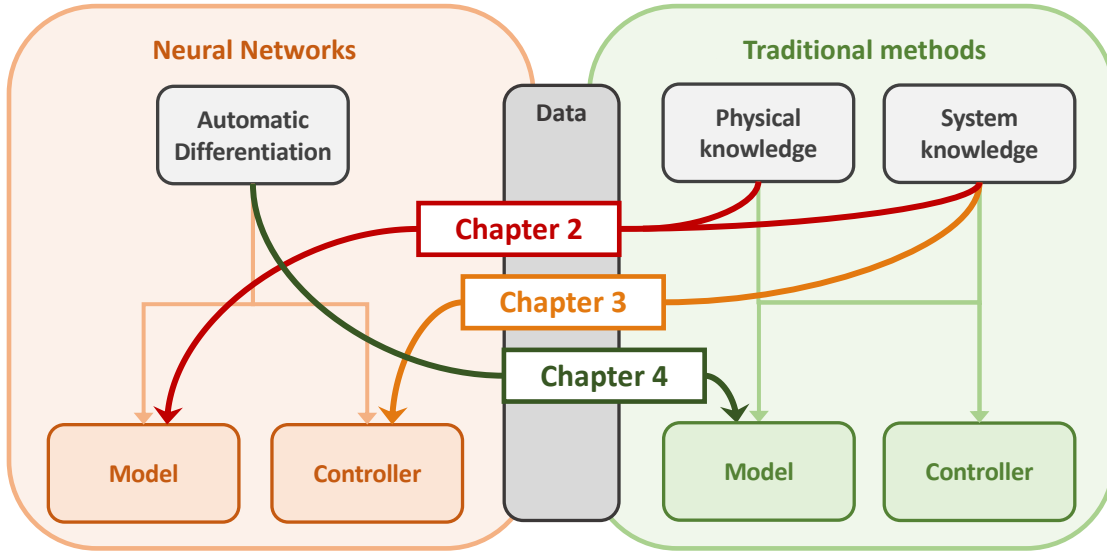


Figure 1.1: Schematic representation of this thesis, which lies at the intersection of traditional methods and Neural Networks, leveraging data to integrate both approaches. We first propose to introduce prior physical and system knowledge into NN-based models and controllers in Chapters 2 and 3, respectively, before bringing the automatic differentiation paradigm for structured system identification in Chapter 4.

plexity of DRL algorithms [29], it often takes months for a vanilla model-free DRL policy to converge [20, 30]. Furthermore, DRL agents rely on exploration to find optimal actions and might hence incur unacceptable discomfort for the occupants or high energy bills during the learning phase [31]. Additional work is thus needed to create DRL agents rapidly converging toward effective solutions and satisfying the comfort needs of the occupants at all times to promote widespread acceptance.

1.2 Marrying Neural Networks and traditional methods

As depicted in Figure 1.1, vanilla NNs habitually rely on Automatic Differentiation (AD) and the backpropagation algorithm to optimize some performance criterion on the measured data without requiring any prior knowledge about the system to model or control. On the other hand, traditional methods generally build upon expert knowledge of the system and/or the underlying physical laws to design *structured* models or controllers, which are then often calibrated using data.

To tackle the aforementioned building modeling and control challenges, this thesis provides insights into hybrid data-driven methods fusing NNs, physical knowledge, and system properties for modeling or controller design. It thus lies at the intersection of these paradigms and is separated into three main Chapters as sketched in Figure 1.1.

Specifically, to exemplify the power of approaches combining the strengths of both NNs and

traditional methods, we first investigate how to alleviate the system- and physics-agnosticism of NN models through prior knowledge integration in Chapter 2. Similarly, we then analyze how to incorporate expert intuition into model-free DRL control policies in Chapter 3. Collectively, these examinations allow us to enforce desired properties on NNs, moving away from vanilla architectures. Conversely, Chapter 4 looks at traditional System Identification (SI) techniques and investigates how to leverage AD to solve challenging SI tasks. These analyses showcase the benefits of leveraging ML tools to address complex problems that typically exceed the capabilities of traditional approaches.

Chapter 2: Physically Consistent Neural Networks for thermal building modeling

Despite their remarkable performance, NNs also come with significant challenges: they are infamous for their brittleness and can fail spectacularly on previously unseen data [27]. This is critical for control-oriented models: if the NN model fails to capture the underlying physical laws, the associated controller might subsequently make spurious decisions [32]. Indeed, when a building model does not capture a physically meaningful relationship between cooling power inputs and temperatures, for example, this can mislead a controller to turn on the air conditioning when it is snowing outside because it *thinks* — according to the model — that this will increase the indoor temperature, similarly to what was observed in [30, 33].

The field of Physics-inspired ML (PiML) has recently emerged to tackle these challenges and bridge the gap between physics-grounded yet limited methods and highly expressive but physics-agnostic ML models [34, 35]. The majority of these recent advancements hinge on Physics-inspired NNs (PiNNs), which typically integrate a physical loss term in addition to the conventional data-driven counterpart. This *steers* NNs towards solutions that not only fit the data well but also align with the underlying physical laws [36]. However, these rules are not enforced but rather encouraged through the additional loss term and PiNNs may still fall short of consistently adhering to them even after extensive training.

To guarantee adherence to the required laws at all times, researchers have hence investigated various ways to directly encode physics in NNs *by design*, giving rise to Lagrangian NNs [37] or Hamiltonian NNs [38], among others. **However, such a tailored architecture has never been applied to building thermal modeling, where compliance with the laws of thermodynamics is required to ensure energy transfers and heat gains are captured accurately.**

Main contributions

In response to the identified need for thermodynamically consistent NNs, Chapter 2 presents one potential solution, dubbed **Physically Consistent Neural Networks (PCNNs)**, specifically applied to building thermal modeling. The key idea is to let a physics-inspired module run in parallel to an NN, the former guaranteeing compliance with the laws of physics and the latter capturing highly nonlinear behaviors. This chapter is heavily inspired by the following papers:

Chapter 1. Introduction

- [39] **Loris Di Natale**, Bratislav Svetozarevic, Philipp Heer, and Colin Jones. Physically consistent neural networks for building thermal modeling: theory and analysis. *Applied Energy*, 325:119806, 2022.
- [40] **Loris Di Natale**, Bratislav Svetozarevic, Philipp Heer, and Colin Jones. Towards scalable physically consistent neural networks: An application to data-driven multi-zone thermal building models. *Applied Energy*, 340:121071, 2023.

In the examined case studies, PCNNs achieve performance on par with vanilla NNs despite their constrained architecture to follow the laws of thermodynamics. On the other hand, they surpassed other physically consistent data-driven methods by 20–30%. Although these investigations were limited to a single building, they hint that PCNNs can indeed achieve state-of-the-art modeling performance among data-driven methods while respecting the underlying physical laws *by design*, thereby alleviating the engineering burden of traditional physics-based approaches. Notably, **the modularity of PCNNs could allow them to be applied beyond building thermal modeling, paving the way towards generic hybrid methods merging NNs and prior expert knowledge for physical system modeling.**

Chapter 3: Prospects and hurdles of Deep Reinforcement Learning for building control

After discussing the use of NNs for thermal modeling in-depth in Chapter 2, we turn to NN controllers trained via DRL in Chapter 3. We first identify seven key characteristics of an *ideal* building controller, namely optimality, robustness to disturbances, constraint satisfaction, adaptability, scalability, transferability, and convergence speed. After thoroughly comparing with other control methods, we argue that *model-free* DRL agents are well-positioned for widespread adoption according to these requirements. Indeed, they can circumvent the challenges associated with the intricate design of accurate models, leading to good adaptability, scalability, and transferability properties [41, 42].

On the other hand, while DRL agents have been compared to MPC controllers in [13, 43, 44], for example, their *optimality gap* — how close to the optimal performance they are — is seldom discussed. Furthermore, as highlighted in Section 1.1.2, vanilla DRL policies suffer from slow convergence speed and might behave inadequately during the exploration phase, i.e., violate the comfort of the occupants. **These concerns naturally hinder real-world experiments and call for data-efficient constrained DRL solutions** [14].

Main contributions

Following the need for advanced building control methods to decrease the energy intensity of the sector discussed in Section 1.1, Chapter 3 proposes characteristics of an *ideal* building controller and a contrastive analysis of some of the existing methods in light of these requirements. Having recognized model-free DRL agents as promising candidates, we then offer

insights to address some of the critical unresolved questions about these controllers, drawing inspiration from the works in:

- [45] **Loris Di Natale**, Bratislav Svetozarevic, Philipp Heer, and Colin Jones. Near-optimal deep reinforcement learning policies from data for zone temperature control. In *2022 IEEE 17th International Conference on Control & Automation (ICCA)*. IEEE, 2022.
- [46] **Loris Di Natale**, Bratislav Svetozarevic, Philipp Heer, and Colin Jones. Computationally Efficient Reinforcement Learning: Targeted Exploration leveraging Simple Rules. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 2334–2339. IEEE, 2023.

Specifically, we show that model-free DRL agents exhibit the potential to achieve near-optimal performance in various settings. Furthermore, they can be constrained to avoid critical failures and converge within a reasonable timeframe. Although these results have yet to be confirmed in different case studies, they hint at the **potential of system-agnostic DRL control policies to ensure the comfort of the occupants while learning to minimize energy consumption from scratch and without engineering overhead, a first step towards generic and widely applicable building controllers.**

Chapter 4: Leveraging automatic differentiation for system identification

Instead of introducing prior knowledge in NNs, either for modeling or control purposes, Chapter 4 turns the problem around and investigates how to leverage ML tools to help traditional SI approaches [47]. Therein, we are particularly interested in methods allowing the integration of desired system properties in the identified model, a nontrivial task with traditional tools. Throughout this chapter, we argue that casting the SI problem in an ML framework can help mitigate some of the associated issues.

For example, in the common case of discrete-time Linear Time-Invariant (LTI) state-space SI, we frequently require the identified model to be stable [48]. Notably, stability can be enforced by modifying the state-space matrices *a posteriori* [49], but this correction might incur significant performance loss [50]. Alternatively, one can leverage *free parametrizations* of stable matrices, such as in [51], to ensure stability *by design*. These methods, however, can only identify generic matrices; there is no mechanism to integrate prior knowledge about the system beyond stability. This may become crucial in practical applications where the state-space matrices are known to have specific sparsity patterns, for example [52]. Conversely, prior knowledge of the system matrices can be enforced through the COSMOS framework [53], for example, but at the expense of stability guarantees.

To make matters worse, extending beyond linear state-space SI to enforce more generic system properties — typically stemming from physical laws — generally adds further complexity to the problem. Indeed, while fitting linear models to minimize the one-step-ahead prediction error simplifies to a Least Squares (LS) optimization problem [54], identifying other types of dynamical models or minimizing the multi-step-ahead prediction error often becomes

Chapter 1. Introduction

more intricate [55]. Overall, **identifying generic systems by minimizing the multi-step-ahead prediction error while incorporating desired structural properties like stability, sparsity, or adherence to physical principles remains challenging.**

Main contributions

Motivated by these shortcomings of existing methods, Chapter 4 introduces the **open-source system-agnostic SIMBa (System Identification Methods leveraging Backpropagation) toolbox**. It leverages automatic differentiation to simultaneously optimize the multi-step-ahead prediction error, ensure the stability of the identified model, and allow for prior knowledge integration for linear state-space SI. Subsequently, we present an extension of SIMBa to identify nonlinear systems while maintaining thermodynamic consistency using Irreversible port-Hamiltonian (IPH) modeling. This chapter is heavily influenced by the following papers:

- [56] **Loris Di Natale**,[†] Muhammad Zakwan,[†] Bratislav Svetozarevic, Philipp Heer, Giancarlo Ferrari Trecate, and Colin Jones. Stable linear subspace identification: A Machine Learning approach. *Submitted to ECC 2024, arXiv:2311.03197*, 2023.
- [57] **Loris Di Natale**,[†] Muhammad Zakwan,[†] Philipp Heer, Giancarlo Ferrari Trecate, and Colin Jones. SIMBa: System Identification Methods leveraging Backpropagation. *Submitted to IEEE Transactions on Control Systems Technology. arXiv:2311.13889*, 2023.
- [58] Muhammad Zakwan,[†] **Loris Di Natale**,[†] Bratislav Svetozarevic, Philipp Heer, Colin Jones, and Giancarlo Ferrari Trecate. Physically consistent neural ODEs for learning multi-physics systems. *IFAC-PapersOnLine* 56(2), 5855-5860, 2023.

[†] Authors contributed equally.

Across thorough numerical experiments, our findings indicate that SIMBa outperforms traditional stable state-space SI methods by more than 25% in the majority of instances. In specific applications, the performance gains can exceed 90%. Similar conclusions are drawn for the nonlinear extension leveraging IPH dynamics, which significantly outperforms classical methods. Collectively, these investigations highlight the potential of ML tools to help scale traditional SI methods to more complex problems. This introduces a **novel paradigm for the identification of structured models from data.**

Credit assignment

Chapter 4 stems from a highly fruitful collaboration with Muhammad Zakwan, with the first two authors of the ensuing papers [56–58] equally sharing the workload. While all the results are reported here for completeness, any merit or credit is thus shared between the two authors. In general, Muhammad Zakwan spearheaded the theoretical contributions while the author of this thesis led the software development and numerical investigations. Note that since the text in this chapter is largely inspired by co-authored papers [56–58], portions of it will likely appear in Muhammad Zakwan’s thesis in a similar fashion.

1.3 Putting everything together

While Chapters 2 and 3 are devoted to building energy consumption reduction applications, we stress here that the methods presented therein may be applied to different fields. They exemplify possibilities to leverage prior and system knowledge to enforce desired properties on NNs, enhancing their reliability and enabling potential widespread real-world applications of this emerging technology. In contrast, Chapter 4 underscores the often untapped potential of ML tools, especially the automatic differentiation framework, to help traditional SI methods tackle previously hard-to-grasp problems. We postulate that analogous techniques to those employed in SIMBa could be utilized to support the integration of ML tools into the design of traditional control methods, addressing the missing link in Figure 1.1.

Remarkably, since NNs also rely on backpropagation during training, they could be seamlessly incorporated into SIMBa, typically to capture unmodeled effects in parallel with the known parametrized dynamics. Interestingly, in that case, SIMBa would recover the PCNN architecture discussed in Chapter 2, with a physics-inspired module and an NN running in parallel. In other words, enforcing desired system properties on standard NNs or, conversely, starting from a traditional physics-grounded model and introducing an NN in parallel to capture complex dynamics both give rise to similar final model architectures.

Although we do not discuss it in detail throughout this work, similar remarks can be made for controllers. For example, we focus on ensuring that NN-based control policies follow some ground rules at all times through computationally inexpensive modifications in Chapter 3, which is conceptually related to the modified PCNN architecture in the modeling case. Conversely, one could start from a known controller — to ensure minimal performance guarantees — and subsequently enhance its performance by adding an NN in parallel, in a similar vein to what is proposed in Chapter 4 for models.

Overall, this thesis provides two alternative perspectives on hybrid methods, either starting from the point of view of ML or control engineers and complementing it with the other perspective, and highlights their similarity. **Altogether, our investigations show the efficacy of integrating traditional and emerging methods to achieve and exceed state-of-the-art performance while ensuring desired properties are respected.**

2 Physically Consistent Neural Networks for thermal building modeling

Given the importance of accurate thermal building models discussed in Section 1.1.2, this chapter is devoted to the development of a novel Physically Consistent Neural Network (PCNN) architecture. PCNNs merge traditional physics-based insights and Neural Networks to simultaneously retain physical consistency and achieve state-of-the-art performance among data-driven methods while naturally scaling to large-scale multi-zone buildings.

2.1 The need for physically consistent Neural Networks

Fueled by the ever-growing available computing capacity and amount of data being collected in various applications, Machine Learning recently entered the Deep Learning (DL) era [26]. Indeed, NNs with hundreds of thousands of parameters are nowadays routinely trained [59]. While their complex architectures allow them to achieve state-of-the-art performance on very different tasks [60–64], deep NNs also come with significant practical challenges, typically stemming from their *data inefficiency* and *lack of generalization* to unseen data [27, 65, 66]. NNs indeed often require millions of samples to be trained accurately, leading to heavy computational burdens. Furthermore, they might subsequently fail to provide meaningful solutions on new data they were not trained on, i.e., fail to generalize. **Throughout this chapter, we aim to analyze this generalization issue, proposing the novel PCNN architecture as one potential solution.**

2.1.1 The generalization issue of Neural Networks

First observed in 2013, this intriguing NN behavior is best visualized on image recognition and classification tasks, where adding small perturbations — indistinguishable to the human eye — on the input image can change the decision of a state-of-the-art NN from a “dog” to a “camel”, for example [66]. Similarly, NNs can attain superhuman performance on image recognition tasks and yet fail when the background changes [67, 68]. This led to the development of adversarial DL, where people look for different ways to fool NNs, exemplifying the brittleness

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

of their predictions [69, 70].

More subtle and maybe more worryingly, NNs can learn *shortcuts* [27], which means they might *fit the training data well without fundamentally understanding the problem*, hence failing to generalize. For example, they can generate captions without ever looking at the corresponding images [71]. In a similar vein, an NN could detect pneumonia from X-ray scans with good accuracy only by looking at hospital-specific tokens and correlating it with each hospital's pneumonia prevalence, never examining the lungs [72]. While these are only a few examples — more details can be found in [27] —, they clearly indicate how NNs can find ways to perform extremely well without fundamentally solving the task at hand. These flawed models are however unable to generalize and cannot be deployed in real-world applications since we have no means to know how they will react to new conditions.

To circumvent this generalization issue, researchers often rely on better data sets that cover the entire spectrum of inputs and allow NNs to react to any situation. This however requires vast computational resources and is only possible in fields where a significant amount of data is available, such as for tasks related to natural language processing [60] or images [73]. Additionally, to ensure some level of generalization, practitioners typically separate the data into training and validation sets, the former being used to train the network and the latter to assess its performance on unseen data to avoid *overfitting* the training data [74]. However, classical NNs cannot be robust to input modifications that do not exist in the entire data set.

Implications for thermal building models

Although PCNNs might be applied to model various physical systems, we are mainly interested in models able to predict the evolution of the temperature inside a building over time throughout this chapter. As discussed above, even though NNs achieve state-of-the-art performance on such time series modeling tasks [75], we cannot trust classical NNs to perform well in all situations and grasp the underlying physical laws. They might indeed violate the laws of thermodynamics despite achieving high accuracy during training, something problematic for control-oriented applications, where the controller subsequently needs to capture the impact of heating and cooling correctly, as mentioned in Section 1.2.

To make matters worse, even if several years of building operation data are available, one will always face an input coverage problem. Indeed, buildings are usually inhabited and operated in a typical fashion to maintain a comfortable temperature for the occupants — heating when it gets cold in winter and cooling when it gets hot in summer. Most data sets are hence inherently incomplete and we cannot hope to learn robust NNs that grasp the effect of heating in summer, for example [20, 30, 33].

This is illustrated in Figure 2.1, where one can compare the temperature predictions of a classical linear physics-based Resistance-Capacitance (RC) model, a classical Long Short-Term Memory network (LSTM), and a PCNN under different heating and cooling power inputs

in one thermal zone. Interestingly, the plotted LSTM achieved a superior accuracy than both other models on the training data — overfitting it, as detailed in Section 2.5.1 and Table 2.2 — but clearly failed to capture the impact of heating and cooling. This hints that **only measuring the accuracy of predictions of NNs might sometimes hide spectacularly flawed behaviors they picked up**, similarly to what was observed in [27].

2.1.2 Introducing physics-based prior knowledge

In general, classical NNs suffer from *underspecification*, as reported in a large-scale study from Google [76], which might explain their brittleness. As a promising countermeasure, researchers started to include prior knowledge — also known as *inductive biases* — into NNs to facilitate their training and improve their performance. This led to the success of the CNN, RNN, and graph NN families, among others, which are specially designed to capture spatial invariance [77], temporal dependencies [78] and structural relations [79] in the data, respectively. When interested in modeling a physical system, such as the thermal dynamics of a building, we often know the underlying physical laws and can hence similarly look to impose constraints on NNs to help them learn meaningful solutions.

In recent years, pioneered by the physics-guided NNs of Karpatne et al. [80, 81] and the more general physics-informed DL framework originally proposed by Raissi et al. [82–84], *Physics-informed* or *Physics-inspired* NN (PiNN) designs flourished [85–87]. While many works modify the loss function of NNs to steer the learning towards physically meaningful solutions [36, 88], these schemes cannot provide any guarantee about the final model respecting the desired constraints. Furthermore, measurement errors or unmeasured heat losses, for example, can corrupt data samples, which might consequently not follow the expected physical laws exactly, making it hard for NNs to simultaneously drive the data- and physics-based losses to zero [89].

To avoid these issues, more systematic approaches directly alter the networks’ architecture to ensure the underlying physical laws are followed *by design*, typically capturing Lagrangian or Hamiltonian dynamics [37, 38, 65]. Additionally, since the desired properties are hard-coded in such models, the loss function does not need to be altered, which avoids common pitfalls of classical PiNNs, such as the difficult trade-off between the accuracy and the physical consistency of the model, which can also increase the amount of data needed [90, 91].

Despite this progress, NNs tailored to capture the laws of thermodynamics — a requirement of building thermal models — were never developed. To this end, we propose the novel PCNN architecture in this chapter, which includes existing knowledge of the system at its core. The main idea is to introduce a physics-inspired module capturing known physical dynamics in parallel to the main NN, injecting an inductive bias in PCNNs such that they do not need to learn everything from data, but only what we cannot easily characterize *a priori*.

Methodologically, PCNNs are close to the physics-interpretable shallow NNs, where the inputs are also processed by two parallel modules, one to retain physical exactness when possible

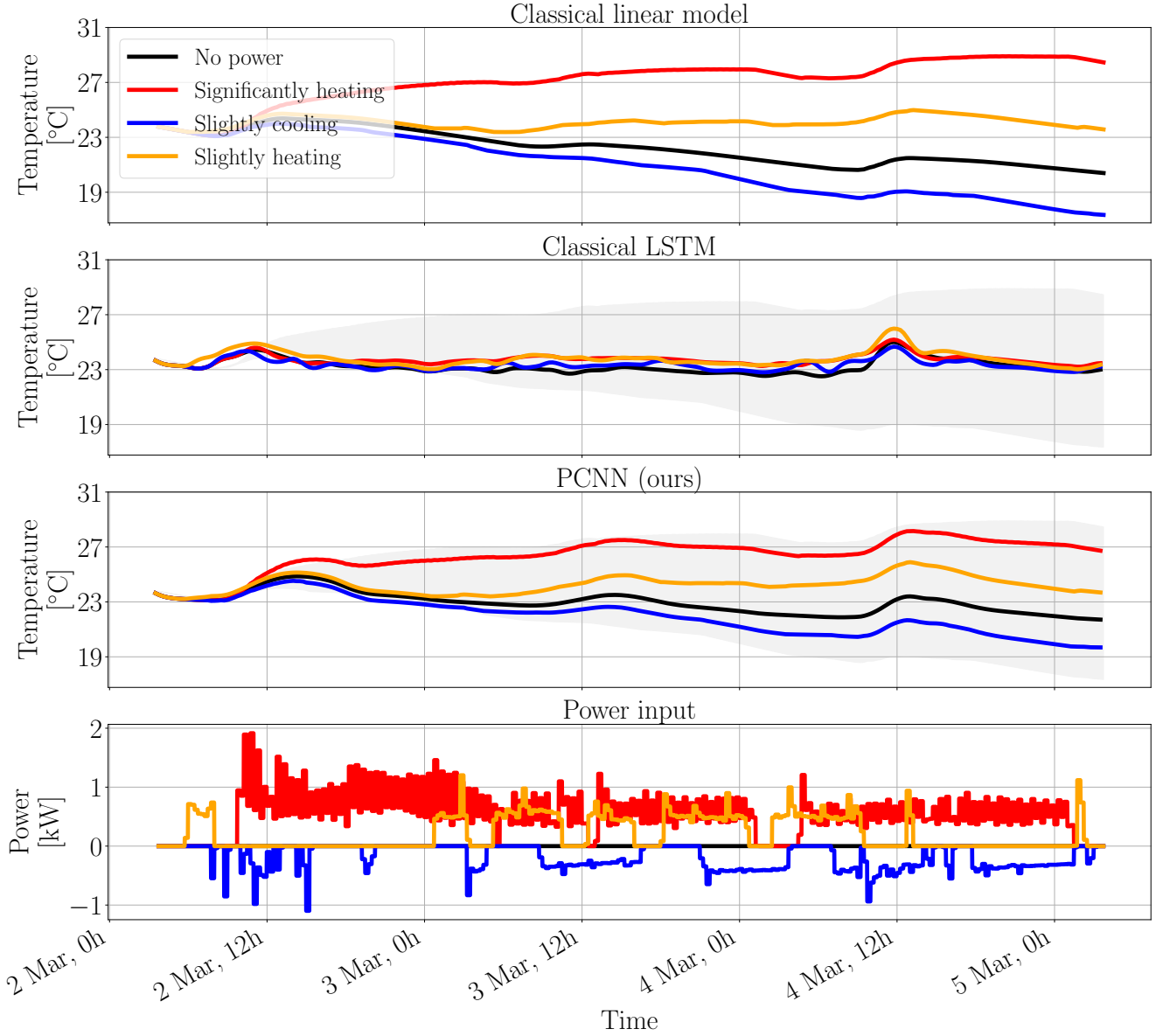


Figure 2.1: Temperature predictions of a linear physically consistent model and the proposed PCNN compared to a classical LSTM under different control inputs. The gray-shaded areas represent the span of the linear model predictions to provide a visual comparison with both black-box methods. While the LSTM presents a lower training error than the PCNN (see Section 2.5.1), indicating a good fit to the data, it does not capture the impact of the different heating/cooling powers applied to the system, e.g., predicting higher temperatures when cooling is on than when heating is. The specific structure of PCNNs introduced in Section 2.3, on the other hand, allows them to retain physical consistency, similarly to classical physics-based models, while improving the prediction accuracy (see Section 2.5.1).

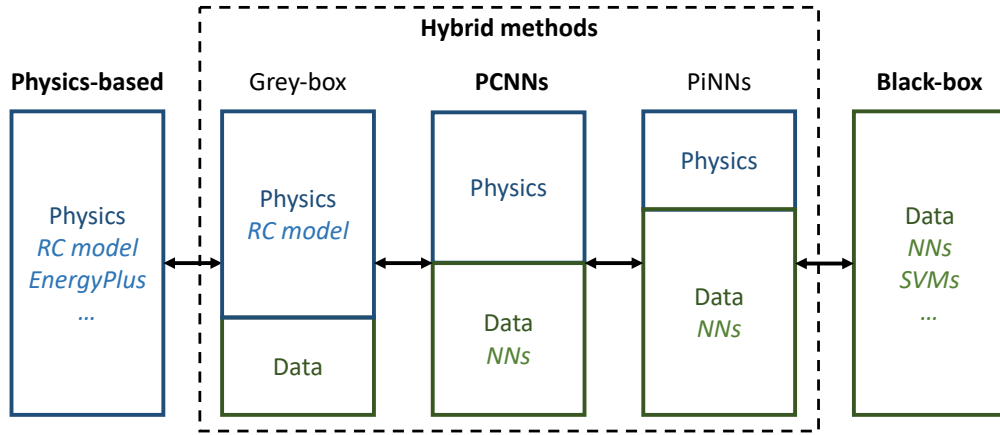


Figure 2.2: Structural differences between the different existing methods, starting from first principles on the left and using more and more data towards black-box approaches.

and one to capture nonlinearities through a shallow NN [92]. Also related in spirit to the PCNN architecture, Hu et al. introduced a specific learning pipeline, where the output of the forward NN is fed back through a physics-inspired NN structure to reconstruct the input and hence ensure the forward process retains physical consistency [93].

2.2 Towards prior knowledge-infused Neural Network building thermal models

Existing building thermal models can be broadly classified into three categories: physics-based, black-box, and hybrid methods, as pictured in Figure 2.2. Given the focus of this section on PiNNs and to emphasize differences with classical gray-box models, we furthermore split hybrid methods into two different parts in this literature overview. Due to the numerous works on building modeling, we only provide a short summary of the strengths and weaknesses of the various techniques herein, and more details can be found in dedicated reviews, such as in [22, 25, 94–100].

2.2.1 From first principles to data-driven models

Since the evolution of the temperature in a thermal zone is governed by the laws of thermodynamics, the most natural way to model it is to write down the corresponding Ordinary Differential Equations (ODEs) and then use custom solvers or discretization schemes to propagate them through time, such as in [101, 102]. These physics-based methods, also known as first principles or *white-box* models, dominated the field early on when the lack of available data hindered the development of data-driven models [94].

Since they are grounded in first principles, natural advantages of these approaches include

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

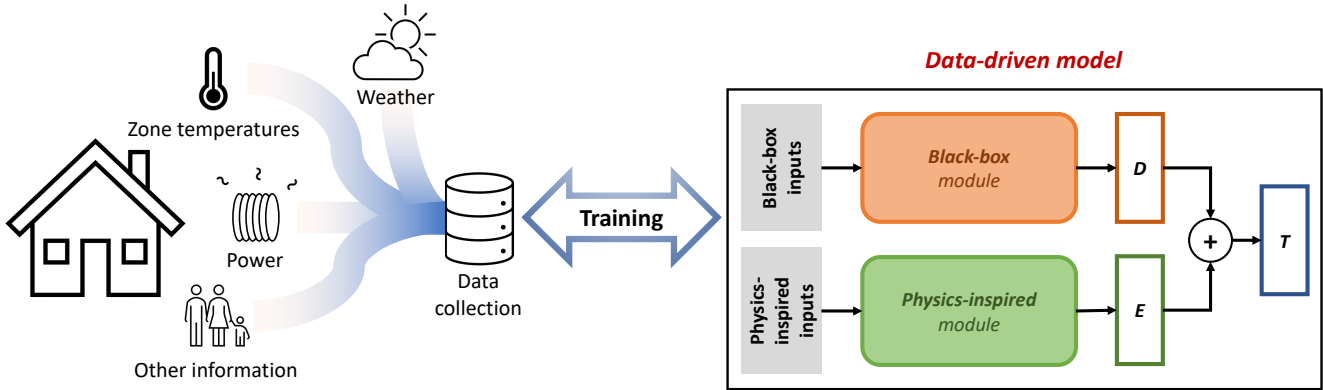


Figure 2.3: General pipeline of data-driven building thermal modeling frameworks, where data is collected from the real building, potentially stored in a database, and then used to train or calibrate a data-driven model. The picture on the right is a schematic representation of the proposed PCNN architecture detailed in Section 2.3

their *interpretability* and generalization [94, 97]. On the other hand, due to the complexity of detailed thermal models, assumptions and simplifications have to be made, such as in the choice of the ODEs, which can limit their accuracy [103]. Moreover, the more precision desired, the more knowledge and time are required to design the model and find the corresponding parameters — the coefficients of the ODEs —, typically concerning the building envelope and the HVAC system [104, 105].

To alleviate this engineering burden, allow more complex structures to be modeled, and accelerate the entire pipeline, custom modeling tools such as EnergyPlus, Modelica, TRNSYS, or IDA ICE are often used in practice [106–108]. Such detailed simulation tools however still require expert knowledge and access to many design parameters that are often not directly available [109], which makes them infamously hard to calibrate [21, 110, 111]. Moreover, while no training is required, solving the complex underlying ODEs to simulate each time step can entail a significant computational burden at run-time [112].

In recent years, owing to the growing amount of data collected in buildings, researchers started to employ data-driven approaches instead, bypassing the cumbersome procedures and expert knowledge required to set up classical physics-based models [100]. This gave rise to so-called gray- or black-box frameworks, both of which use historical data collected in buildings for calibration or training purposes, as sketched in Figure 2.3.

2.2.2 Black-box models

As opposed to white-box methods, black-box models do not rely on first principles but solely derive patterns from historical data. The most widely used approaches for building thermal

2.2 Towards prior knowledge-infused Neural Network building thermal models

modeling rely on multiple linear or support vector regressions, NNs, and ensembles, apart from the classical AutoRegressive Integrated Moving Average (ARIMA) models [100].

Black-box models are generally easier to deploy — since no expert knowledge is required at the design stage —, more flexible, and thus often more scalable than physics-based ones [25, 99, 113]. Furthermore, since they do not have to follow a predefined underlying architecture, black-box methods are generally more expressive, capable of capturing unknown nonlinear dynamics, and hence usually perform better [97]. Finally, they are generally easier to transfer from one building to another as similar model architectures can be used despite their different dynamics. Since all the parameters are learned from data, these approaches can indeed fit a large number of buildings simultaneously, such as in [114], where 1'000 households were automatically modeled with the same architecture.

On the other hand, black-box models often lack generalization guarantees outside of the data they are trained on [94, 99]. Furthermore, they need historical data as input, sometimes in large amounts, to achieve satisfactory accuracy [99]. The data additionally has to be *exciting enough*, i.e., to cover the different operating conditions of the building, something not trivial, as discussed in Section 2.1.1. While data imbalance issues can, for example, be tackled through the creation of sub-models [115], this does not scale well with the number of operating points. Moreover, black-box models are sensitive to the features — or the feature extraction method — used as inputs [103].

Remarkably, all these issues are amplified when NNs are involved. Nonetheless, very recently, as a consequence of the growing amount of available data and surfing on the boom of DL applications, many studies leveraged their expressiveness for building thermal modeling, e.g., in [15, 20, 100, 116, 117]. Due to the nonconvexity of classical NNs, which makes them hard to use in optimization procedures, researchers also used specific control-oriented models, such as Input Convex NNs (ICNNs) [118].

2.2.3 Hybrid methods

Hybrid methods combine physics-based knowledge with historical data, striking a trade-off between both worlds. Note that some researchers use the term “hybrid methods” for approaches first building an accurate physics-based model and then generating data with it to train a black-box surrogate to accelerate the inference procedure at run-time, such as in [112, 119], which is out of the scope of this overview and hence not covered here.

Classical gray-box models

When a control-oriented thermal building model is designed, typically for MPC, data is in most cases used to identify the parameters of a simplified physics-based model [95], usually a low-order RC model, such as in [23, 109, 120–124]. Such gray-box approaches require less expert knowledge than pure physics-based models since simplified equations are used. On

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

the other hand, the chosen ODEs incorporate physical knowledge in the models, so that less information has to be learned from data compared to pure black-box models, in turn implying that less historical data is usually required to fit their parameters [95]. Overall, gray-box models are particularly popular due to their ease of implementation, interpretability, close ties to the underlying physics, and because they can be designed to be linear.¹

Despite these advantages, the parameter identification procedure of RC models is generally nontrivial and sensitive to the data quality [23, 126]. Additionally, some nonlinearities might not be well-captured [113], partially explaining why low-order models often perform better than complex ones [24, 127, 128]. Higher-order models furthermore entail more complexity, which can hinder their generalization to unseen data and hence also advocates in favor of low-complexity frameworks [24]. In sum, gray-box approaches allow for a trade-off between the accuracy and the complexity of building models [24]. As a partial solution to this dilemma, a framework to test the flexibility, scalability, and interoperability of gray-box approaches and select the right model architecture was proposed in [23].

Due to the effectiveness of low-complexity architectures, we rely on linear first-order RC modeling techniques inspired by Bünning et al. [129] and simplified versions of Maasoumy et al. [120, 121, 130] to construct the physics-inspired module of PCNNs in this chapter, as detailed in Appendix A.1 and Section 2.3. **This low-complexity physics-inspired module is particularly effective in the case of PCNNs since the black-box module simultaneously captures unmodeled complex nonlinearities in parallel.**

Residual models

Alternatively or additionally, one can leverage historical data to compute the error of a model — often a simplified first principles one — and then fit these residuals with another method to improve performance, such as in [131, 132]. Notably, such approaches are classically separated into two distinct steps, first designing or learning the physics-based model and then fitting its residual error, typically with an NN. We will refer to this type of model as *residual models* in the remainder of this chapter. They are to be contrasted with the proposed PCNNs, which are conceptually close but where both modules are trained simultaneously.

2.2.4 Physics-inspired Neural Network building models

When applying NNs to physical systems, one should always keep their well-known generalization issue discussed in Section 2.1.1 in mind to ensure the underlying physical laws are respected. Despite the recent popularity of the field, to the best of the authors' knowledge, PiNNs were only applied to thermal building modeling in [89, 133–139].

Gokhale et al. and Chen et al. relied on the classical PiNN framework, augmenting the loss

¹This characteristic is particularly desirable in MPC applications since an appropriate choice of objective function then renders the optimization problem to solve at each time step convex and thus tractable [125].

function of their NNs and creating latent states to include some physical intuition in otherwise standard networks [133, 134]. Similarly, Liang et al. trained seven different NNs to predict different quantities of interest and augmented the loss function of each network to steer their outputs towards physically meaningful relations [89]. In another line of work, Nagarathinam et al. designed a specific PiNN architecture for building control [135]. Wang et al. introduced a block lower-triangular NN formulation that retains causality and is well-suited to multi-steps ahead predictions in MPC [136]. However, none of these works can provide guarantees about the physical consistency of their solutions in general.

On the other hand, Drgoňa et al. used NNs to replace the matrices in linear models of building dynamics, which allowed them to enforce the stability and dissipativity of the learned system by constraining the eigenvalues of one of the NNs using the Perron-Frobenius theorem [137]. The state-space matrices were similarly replaced by NNs in [138], with additional nonlinearities and tailored RNN architectures, to predict indoor air quality.

Finally, building on the PCNNs originally proposed in [39], Xiao et al. recently extended them to simultaneously predict the temperature and humidity in a multi-zone building [139]. They additionally modified the physics-inspired module to incorporate nonlinearities without jeopardizing compliance with the underlying physical laws.

2.3 Physically Consistent Neural Networks

With all the considerations about the brittleness of NNs and the ensuing need for them to respect the underlying physical laws outlined in Section 2.1 in mind, this section details the proposed **Physically Consistent Neural Network (PCNN)** architecture. Focusing on a building thermal modeling case study, we show how PCNNs guarantee the required compliance with the laws of thermodynamics while leveraging the expressiveness of NNs to achieve state-of-the-art performance among data-driven methods.

A note on the topology of buildings

Throughout this chapter, two thermal zones are said to be *adjacent* if they share at least one common wall in a building \mathcal{B} , and the collection of zones adjacent to a given zone z form its *neighborhood* $\mathcal{N}(z)$. We consider a zone to be included in its own neighborhood, i.e., $z \in \mathcal{N}(z)$. Similarly, a zone is connected with the outside if it comprises at least one external wall.

To generalize the notion of neighborhood, we define the *n-hop neighborhood* $\mathcal{N}^n(z)$ as the set of zones that can be reached in n steps from zone z , moving to an adjacent zone at each step. Note that, by definition, we have $\mathcal{N}^1(z) = \mathcal{N}(z)$, and $y \in \mathcal{N}^n(z) \iff z \in \mathcal{N}^n(y)$.

Throughout this chapter, we assume the building to be *connected*, i.e., there is no zone (or group of zones) isolated from the rest. This assumption is trivial in practice since one can easily train several separate models if this condition is not met.

2.3.1 Physically consistent building thermal dynamics

While PCNNs can be applied to model a wide spectrum of physical systems, we focus herein on describing building thermal dynamics. We deem the temperature model of a building \mathcal{B} with m thermal zones to be *physically consistent* if the following conditions are met for each zone $z \in \mathcal{B}$:

$$\frac{\partial T_i^z}{\partial v_j^z} > 0, \quad \forall 0 \leq j < i, \quad (2.1)$$

$$\frac{\partial T_i^z}{\partial T_j^{out}} > 0, \quad \forall 0 \leq j < i, \quad (2.2)$$

$$\frac{\partial T_i^z}{\partial T_j^y} > 0, \quad \forall 0 \leq j < i, \quad \forall y \in \mathcal{N}^{i-j}(z), \quad (2.3)$$

where T^z is the temperature in zone z , v^z the heating or cooling power input applied therein, T^{out} the outside temperature, and the subscripts indicate the time step.

In words, (2.1) implies that applying more heating power v_j^z at a given time step j in any zone z leads to higher temperatures T_i^z for all subsequent time steps $i > j$. That is to say, heating a zone has the expected and intuitive impact of increasing its future temperature, following the laws of thermodynamics. Note that cooling powers are defined to be negative by convention in this chapter, ensuring lower zone temperatures when more cooling is applied, as expected. Similarly, (2.2) encodes the fact that higher ambient temperatures induce higher temperatures inside, and (2.3) guarantees that higher temperatures in zone $y \in \mathcal{N}^n(z)$ lead to higher temperatures in zone z after n steps.

Remark 1 (Extensions of physical consistency). *Note that the definition of physical consistency proposed in (2.1)–(2.3) can easily be extended for applications where additional criteria need to be met by the learned model, for example, to enforce physically consistent temperature predictions with respect to solar gains.*

Remark 2 (Generalization of the approach). *Equations (2.1)–(2.3) can be seamlessly adapted to other fields beyond building modeling where simple physical rules can be encoded in a similar fashion. One can then construct a PCNN architecture following the principles presented in the rest of this section to ensure the learned model respects the desired criteria. With such an architecture, the NN running on top of the simplified physics will capture unmodeled phenomena, increasing the representation power of the model.*

2.3.2 Single-zone Physically Consistent Neural Network building models

Conceptually, the key idea of the proposed PCNN architecture is to let a *black-box* and a *physics-inspired* module running in parallel to compute the next output at each step, as depicted on the right of Figure 2.3. The black-box module captures potentially complex nonlinearities while the physics-inspired one ensures that predefined rules are respected,

which are typically representing physical laws and encoded by conditions similar to the ones in (2.1)–(2.3). One possible PCNN architecture modeling the evolution of the temperature in a single thermal zone z while respecting (2.1)–(2.3) is detailed in Figure 2.4. It can be mathematically described as

$$D_{k+1}^z = D_k^z + f^z(x_k^z) \quad (2.4)$$

$$E_{k+1}^z = E_k^z + a_h^z \max\{g^z(u_k^z), 0\} + a_c^z \min\{g^z(u_k^z), 0\} - b^z(T_k^z - T_k^{out}) - \sum_{y \in \mathcal{N}(z)} c_y^z(T_k^z - T_k^y) \quad (2.5)$$

$$\begin{aligned} T_{k+1}^z &= D_{k+1}^z + E_{k+1}^z \\ D_0^z &= T^z(0) \\ E_0^z &= 0, \end{aligned} \quad (2.6)$$

where $D \in \mathbb{R}$ represents the evolution of the black-box module, $E \in \mathbb{R}$ is the *energy accumulator*, i.e., the physics-inspired module, and $T(0)$ is the initial temperature measurement.

The linear physics-inspired module. First, E is influenced by the power input to the zone $v := g(u) \in \mathbb{R}$ at each step, which depends on the control input u , such as the opening pattern of radiator valves, transformed into thermal power by a function g . These inputs are scaled by a constant $a_h > 0$ in the heating and $a_c > 0$ in the cooling case to represent their effect on the air mass in the room. Since cooling power inputs are defined to be negative, cooling the zone lowers the energy accumulated in E , as expected. Second, from the laws of thermodynamics, we know that the modeled zone loses energy through heat transfers to the environment and neighboring zones. We hence subtract these effects, which are proportional to the corresponding temperature gradients with the outside temperature T^{out} and the temperature in neighboring zones T^y , scaled by parameters b and c_y , respectively.

The design of (2.5) is heavily inspired by the classical linear RC building model derived in Appendix A.1.1. The main difference between the generic RC model (A.1) and (2.5) is that PCNNs treat nonlinear solar and additional unknown heat gains using NNs or other nonlinear functions in D instead of relying on engineering-based solutions.

Note that the physics-inspired parameters a_h , a_c , b , and $\{c_y\}_{y \in \mathcal{N}(z)}$ are learned from data simultaneously to the parameters of the black-box module, which is one of the main reasons behind the effectiveness of PCNNs, as discussed in Section 2.5.2.

Remark 3 (Losses to the environment). *For clarity of notation, we assume throughout this chapter that every zone has an external wall. In practice, this assumption is trivially lifted by forcing $b^z = 0$ on all the zones located in the interior of the building.*

Remark 4 (Design of g). *In some buildings, we can directly measure the heating or cooling power input to the zone, i.e., $g(u) = u$. When this is not possible, e.g., when u controls the opening of the valves in radiators, we need to process the controllable inputs into power inputs through some function g . This function might be engineered, for example, as $g(u) = u * \dot{m} * (T^w - T)$*

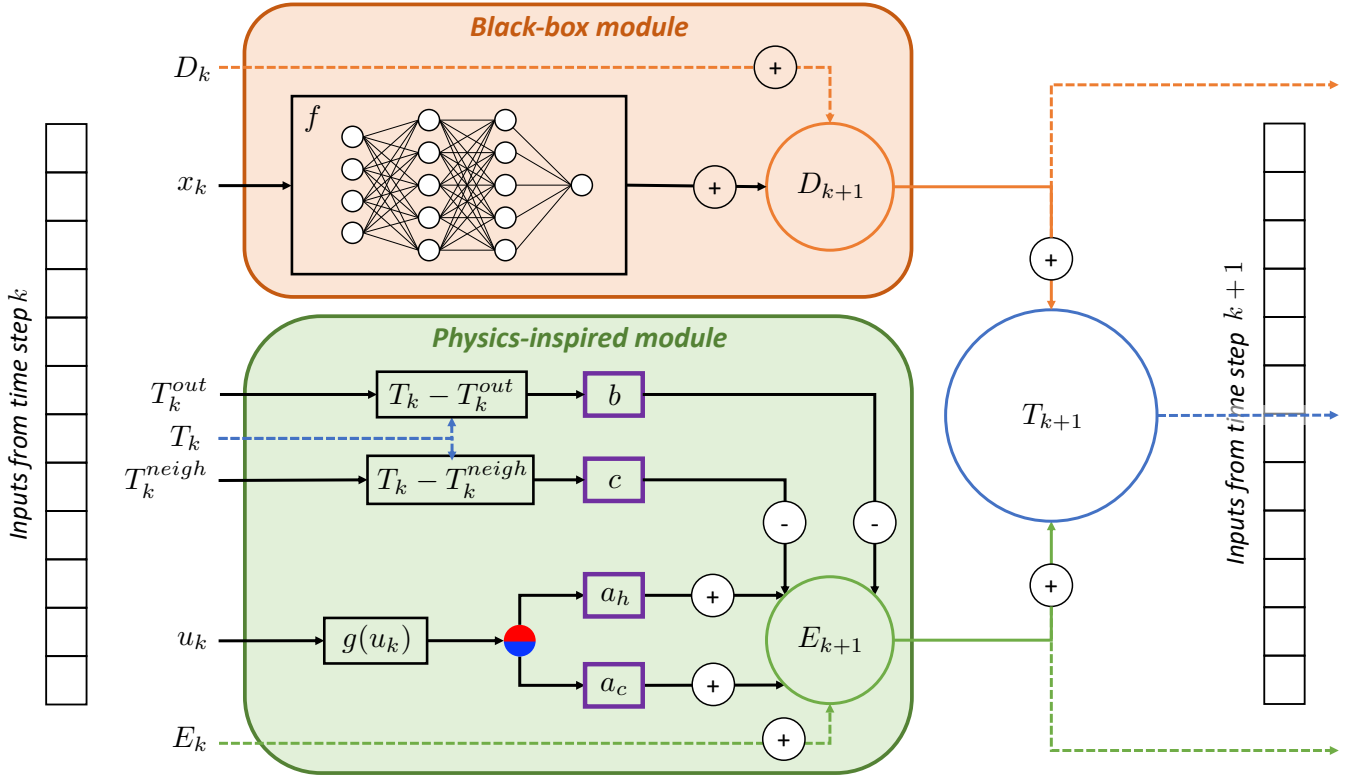


Figure 2.4: The proposed PCNN architecture to model a single zone z , comprised of an orange black-box and a green physics-inspired module, both evolving recursively through the prediction horizon. The dependence on z is dropped and only one neighboring zone $z' \in \mathcal{N}(z)$, denoted *neigh*, is considered for clarity of presentation. The control inputs u , transformed into power inputs by the function g , and the losses to the environment $b(T - T^{out})$ and neighboring zone $c(T - T^{neigh})$ all influence an energy accumulator E , which accumulates or dissipates energy at each time step depending on the received heat gains or losses. Here, the red/blue branching signals a different treatment of the power inputs in the heating and cooling case, respectively, since they are scaled by different constants a_h and a_c . The accumulated energy is then added to the unforced dynamics D , modeled by a residual NN that takes all the features apart from u , T^{out} , and T^{neigh} – gathered in x – as input, to get the final zone temperature prediction T .

for a radiator with mass flow m , water temperature T^w , and u recording the position of the valves in $[0, 1]$. Alternatively, it can be learned from data, e.g., using NNs. However, this learned function should be strictly monotonically increasing, i.e., $\frac{\partial g(u)}{\partial u} > 0$, with $g(0) = 0$: no energy is consumed when there is no control input, $g(u) < 0$ when cooling is on, and $g(u) > 0$ when heating is applied. Importantly, since everything is trained together in an end-to-end fashion in PCNNs, g can seamlessly be learned in parallel to the other parameters.

The black-box module. Running in parallel with the physics-inspired module, the black-box module processes all inputs not treated in E , such as solar gains and time information, gathered in $x \in \mathbb{R}^n$, through a potentially highly nonlinear function f . That is to say, it captures the *unforced* temperature dynamics when no heating or cooling is applied and heat losses are neglected. The independence of f on u , T^{out} , and T will allow us to prove the physical consistency of PCNNs with respect to these inputs in Section 2.3.4.

In practice, the black-box module can typically be designed with residual NNs, choosing f in (2.4) to be any recurrent NN architecture to grasp time dependencies in the data. While f is composed of an encoder-LSTM-decoder structure in our case (see Section 2.4.3), any NN architecture — and even differentiable functions that do not contain NNs — can be used without affecting the physical consistency of the predictions. Nonetheless, due to the sequential nature of temperature dynamics and the expressiveness of NNs, we suspect them to be a good choice in general.

Remark 5 (Coupling between D and E). *Note that since $T_k = D_k + E_k$, the nonlinear black-box module D influences the evolution of the energy accumulator E in Equation (2.5), which is one of the main differences with classical residual techniques, where the physics-based and black-box modules are usually completely separated. This furthermore requires learning the parameters a_h , a_c , b , and $\{c_y\}_{y \in \mathcal{N}(z)}$ simultaneously to f .*

2.3.3 Extensions to the multi-zone setting

While the PCNN architecture described in Section 2.3.2 works well for single-zone modeling, we also propose three possible extensions to simultaneously capture the evolution of the temperature in several interconnected zones exchanging energy, i.e., in a whole building. The only additional information required is the *topology* of the modeled building — which zones are adjacent and which have an external wall —, and multi-zone PCNNs then learn its thermal behavior from data without additional engineering overhead.

Remark 6 (Unknown topology). *If the topology is unknown, one can assume each pair of zones to be adjacent and every zone to have an external wall and then learn to put non-existing connection parameters to zero from data.*

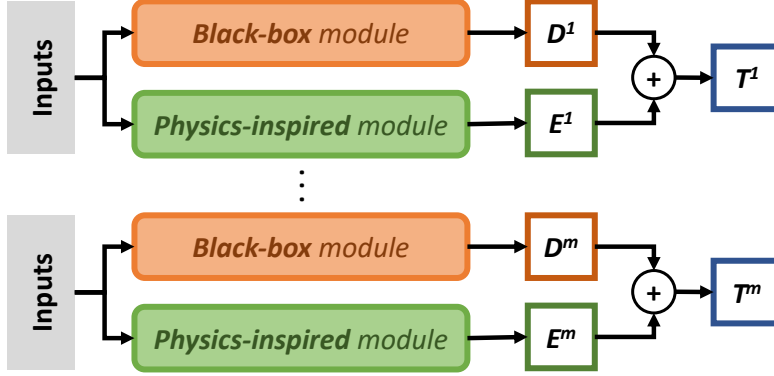


Figure 2.5: X-PCNN: the temperature of each of the m zones is predicted separately.

X-PCNNs: learning several single-zone PCNNs

The most natural and straightforward multi-zone extension is to separately learn one single-zone PCNN for each of the zones to model, as depicted in Figure 2.5. Since this method involves duplicating the original structure for each zone and fitting them independently before combining them, we will refer to the final model as the *X-PCNN* architecture.

For this model to be physically consistent, however, one needs to ensure that $c_y^z = c_z^y$ for each pair of adjacent zones y and z in (2.5). This ensures that the amount of energy flowing from z to y always equals the amount of energy received by y from z , and vice versa. Since each zone is modeled and trained separately in this case, such a condition cannot be imposed during the learning phase, and we thus rely on a heuristic to correct the parameters and enforce this desired property *a posteriori*. Once the models have been trained, for every pair of adjacent zones z and y , we compute the average value identified by both PCNNs and define

$$c^{zy} = c^{yz} := \frac{c_y^z + c_z^y}{2}. \quad (2.7)$$

We then replace c_y^z with c^{zy} in (2.5) for all $y \in \mathcal{N}(z)$ and for all $z \in \mathcal{B}$.

M-PCNNs: sharing the physics-inspired module

To avoid the hand-crafted correction (2.7), which might significantly impact the parameters learned by each PCNN, one can fuse all the physics-inspired modules together, again leveraging our prior knowledge of the underlying physical laws. This gives rise to the so-called *M-PCNN* architecture, pictured in Figure 2.6, where distinct black-box modules are assigned to each zone, but the physics-inspired module is shared. It thus outputs a vector $\mathbf{E} \in \mathbb{R}^m$

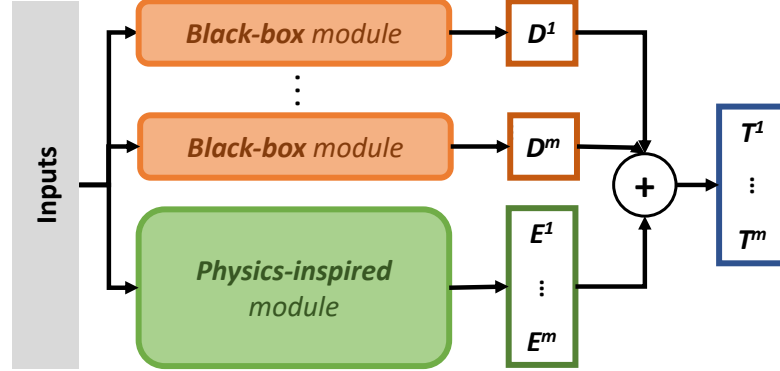


Figure 2.6: **M-PCNN**: the physics-inspired module is shared but multiple black-box modules are learned, one for each zone.

containing the energy accumulated in each zone at each step:

$$\begin{aligned}
 \mathbf{E}_{k+1} &= \mathbf{E}_k + \mathbf{a}_h \odot \max\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} + \mathbf{a}_c \odot \min\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} \\
 &\quad - \mathbf{b} \odot (\mathbf{T}_k - \mathbf{T}_k^{out}) - \Delta \mathbf{T}_k \\
 \mathbf{E}_0 &= \mathbf{0},
 \end{aligned} \tag{2.8}$$

where bold notations correspond to vectorized quantities in \mathbb{R}^m , one dimension for each zone — $\mathbf{a}_h = [a_h^1, \dots, a_h^m]^T$, and similarly for \mathbf{a}_c , \mathbf{b} , and \mathbf{u} —, \odot stands for the element-wise product of two vectors, and $\mathbf{g}(\mathbf{u})_k = [g^1(u_k^1), \dots, g^m(u_k^m)]^T$. Since there is a unique ambient temperature impacting all the zones, we furthermore define $\mathbf{T}^{out} = [T^{out}, \dots, T^{out}]^T \in \mathbb{R}^m$. Finally, $\Delta \mathbf{T}_k \in \mathbb{R}^m$ corresponds to energy transfer between each zone and its neighborhood:

$$\Delta \mathbf{T}_k^z = \sum_{y \in \mathcal{N}(z)} c^{zy} (T_k^z - T_k^y), \quad \forall z \in \mathcal{B}, \tag{2.9}$$

where the superscript z denotes the z -th entry of a vector. By definition, we know $c^{zy} = c^{yz}$ if y and z are adjacent since both represent the same heat transfer coefficient, which is easily enforced during training since all the zones are now modeled simultaneously.

Each dimension of $\mathbf{T} \in \mathbb{R}^m$, i.e., the temperature in each zone z , is then computed as the sum of the physics-inspired and black-box modules:

$$\mathbf{T}_{k+1}^z = D_{k+1}^z + \mathbf{E}_{k+1}^z \tag{2.10}$$

$$D_{k+1}^z = D_k^z + f^z(x_k^z) \tag{2.11}$$

$$D_0^z = T^z(0).$$

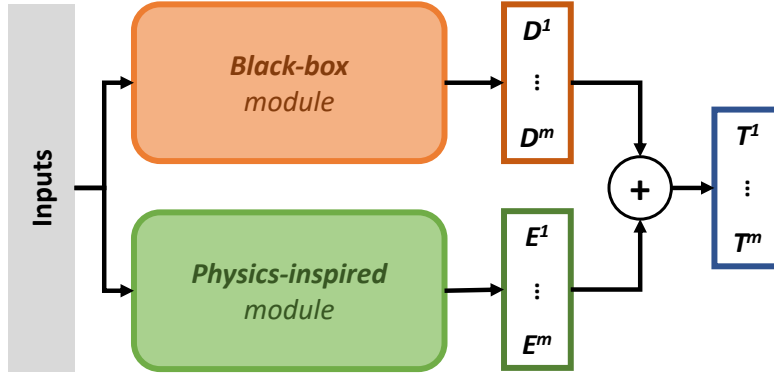


Figure 2.7: **S-PCNN**: both the black-box and physics-inspired modules are shared by all the zones..

S-PCNNs: sharing both modules

To reduce computational complexity and introduce parameter sharing between the zones — which are typically similar in the same building —, we propose a third architecture, dubbed *S-PCNN*, where both the black-box and physics-inspired modules are shared. In other words, the black-box module now has m outputs corresponding to the main dynamics of each of the zones, as pictured in Figure 2.7. Using the vectorized notations as before, with $\mathbf{D} \in \mathbb{R}^m$, we can write the equations of this architecture as follows:

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \tilde{\mathbf{f}}(\tilde{\mathbf{x}}_k) \quad (2.12)$$

$$\begin{aligned} \mathbf{E}_{k+1} = & \mathbf{E}_k + \mathbf{a}_h \odot \max\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} + \mathbf{a}_c \odot \min\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} \\ & - \mathbf{b} \odot (\mathbf{T}_k - \mathbf{T}_k^{out}) - \Delta \mathbf{T}_k \end{aligned} \quad (2.13)$$

$$\mathbf{T}_{k+1} = \mathbf{D}_{k+1} + \mathbf{E}_{k+1} \quad (2.14)$$

$$\mathbf{D}_0 = \mathbf{T}(0)$$

$$\mathbf{E}_0 = \mathbf{0},$$

where the physics-inspired module is the same as for the M-PCNN but we now only have one shared nonlinear function $\tilde{\mathbf{f}}: \mathbb{R}^{d'} \rightarrow \mathbb{R}^m$ transforming the inputs $\tilde{\mathbf{x}} \in \mathbb{R}^{d'}$. Throughout this work, we only consider external inputs that are shared by all the zones — solar irradiation and time information —, leading to $d' = d$ and $\tilde{\mathbf{x}} := \mathbf{x}^z, \forall z \in \mathcal{B}$.

Remark 7 (Zone-dependent inputs). *If some measurements differ zone by zone, one can either stack them in a vector $\tilde{\mathbf{x}} = [(x^1)^\top, \dots, (x^m)^\top]^\top$ and use (2.12) or for example design a shared function $\tilde{\mathbf{f}}: \mathbb{R}^d \rightarrow \mathbb{R}$ and modify (2.12) to $\mathbf{D}_{k+1}^z = \mathbf{D}_k^z + \tilde{\mathbf{f}}(\mathbf{x}_k^z), \forall z \in \mathcal{B}$.*

2.3.4 PCNNs are consistent with the laws of thermodynamics

After applying the transformation (2.7) ensuring that heat transfer coefficients are physically consistent, one can vectorize the X-PCNN physics-inspired module (2.5) to get

$$\begin{aligned} E_{k+1} = & E_k + \mathbf{a}_h \odot \max\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} + \mathbf{a}_c \odot \min\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} \\ & - \mathbf{b} \odot (\mathbf{T}_k - \mathbf{T}_k^{out}) - \Delta \mathbf{T}_k, \end{aligned} \quad (2.15)$$

using the definition of $\Delta \mathbf{T}$ from (2.9). As can be seen directly, this expression is the same as the ones describing M- and S-PCNN physics-inspired modules in (2.8) and (2.13). This means all the proposed multi-zone PCNNs rely on the same physics-inspired module at inference time, with however possibly different parameter values learned during training. This is intuitively expected since they all model the same thermal effects and hence must follow the same physical principles.

In a similar vein, we can rewrite the black-box modules of the X- and M-PCNN architectures in vectorized form as

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \tilde{\mathbf{f}}(\tilde{\mathbf{x}}_k), \quad (2.16)$$

where $\tilde{\mathbf{f}} = [f^1(x^1), \dots, f^m(x^m)]^T$ and $\tilde{\mathbf{x}} = [(x^1)^\top, \dots, (x^m)^\top]^\top$ groups the different inputs.

Leveraging the reformulations in (2.15) and (2.16), we can hence mathematically represent each of the three proposed architectures as

$$\begin{aligned} \mathbf{T}_{k+1} &= \mathbf{D}_{k+1} + \mathbf{E}_{k+1} \\ &= \mathbf{T}_k + \mathbf{f}(\mathbf{x}_k) + \mathbf{a}_h \odot \max\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} + \mathbf{a}_c \odot \min\{\mathbf{g}(\mathbf{u})_k, \mathbf{0}\} \\ &\quad - \mathbf{b} \odot (\mathbf{T}_k - \mathbf{T}_k^{out}) - \Delta \mathbf{T}_k \\ \mathbf{D}_0 &= \mathbf{T}(0) \\ \mathbf{E}_0 &= \mathbf{0}, \end{aligned} \quad (2.17)$$

where $\mathbf{f}(\mathbf{x}_k)$ stands for $\tilde{\mathbf{f}}(\tilde{\mathbf{x}}_k)$ or $\tilde{\mathbf{f}}(\tilde{\mathbf{x}}_k)$ for S-PCNNs, respectively X- and M-PCNNs. The only structural difference between the three proposed models — once the heat transfer coefficients of the X-PCNN have been adjusted by (2.7) — hence comes from the form of $\mathbf{f}(\mathbf{x})$. Remarkably, however, this does not impact their physical consistency, as demonstrated in the following two propositions.

Proposition 1 (Physically consistent heat propagation). *Independently of the structure of \mathbf{f} and \mathbf{x} , any model of the form (2.17) satisfies:*

$$\frac{\partial \mathbf{T}_i^z}{\partial \mathbf{T}_j^y} \geq 0, \quad \forall z, y \in \mathcal{B}, \forall 0 \leq j < i, \quad (2.18)$$

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

with equality if and only if $y \notin \mathcal{N}^{(i-j)}(z)$, as long the following conditions hold:

$$b^z + \sum_{y \in \mathcal{N}(z)} c^{zy} < 1, \quad \forall z \in \mathcal{B}, \quad (2.19)$$

$$c^{zy} > 0, \quad \forall z \in \mathcal{B}, \forall y \in \mathcal{N}(z). \quad (2.20)$$

Proof. See Appendix A.2.1. □

In words, Proposition 1 shows that heat propagates from any zone y to all the other zones z as physically expected, i.e., higher temperatures in zone y will lead to higher temperatures in all the other zones $z \in \mathcal{N}^n(y)$ after n steps.

This proposition can then be used to prove that heating or cooling any zone ultimately increases, respectively decreases, the temperature in the whole building through heat transfers, and that higher ambient temperatures also ultimately heat the entire building. These two facts are formalized in the next proposition.

Proposition 2 (Physically consistent impact of power inputs and ambient temperatures). *Independently of the structure of \mathbf{f} and \mathbf{x} , any model of the form (2.17) satisfies*

$$\frac{\partial \mathbf{T}_i^z}{\partial \mathbf{u}_j^y} \geq 0, \quad \forall z, y \in \mathcal{B}, \forall 0 \leq j < i, \quad (2.21)$$

with equality if and only if $y \notin \mathcal{N}^{(i-j-1)}(z)$, and

$$\frac{\partial \mathbf{T}_i^z}{\partial T_j^{\text{out}}} > 0, \quad \forall z \in \mathcal{B}, \forall 0 \leq j < i, \quad (2.22)$$

as long as (2.19)-(2.20) hold and:

$$a_h^z, a_c^z, b^z > 0, \quad \forall z \in \mathcal{B}, \quad (2.23)$$

$$\frac{\partial g(u)}{\partial u} > 0, \quad (2.24)$$

$$g(0) = 0. \quad (2.25)$$

Proof. See Appendix A.2.2. □

Corollary 1 (Physical consistency of multi-zone thermal PCNNs). *Independently of the structure of \mathbf{f} and \mathbf{x} , any model of the form (2.17) respects the criteria (2.1)-(2.3) if*

$$b^z + \sum_{y \in \mathcal{N}(z)} c^{zy} < 1, \quad \forall z \in \mathcal{B}, \quad (2.26)$$

$$a_h^z, a_c^z, b^z, c^{zy} > 0, \quad \forall z \in \mathcal{B}, \forall y \in \mathcal{N}(z), \quad (2.27)$$

$$\frac{\partial g(u)}{\partial u} > 0, \quad g(0) = 0. \quad (2.28)$$

Proof. Assuming that (2.26)–(2.28) hold, we can apply Propositions 1 and 2. Setting $z = y$ in (2.21) and recalling that any zone is in its own neighborhood — which implies strict positiveness of (2.21) —, PCNNs satisfy (2.1). The satisfaction of (2.2) directly follows from the second part of Proposition 2. Finally, according to Proposition 1, (2.18) is strictly positive if and only if $y \in \mathcal{N}^{(i-j)}(z)$, satisfying (2.3). \square

Remark 8 (Inputs of \mathbf{f}). *While the structure of \mathbf{f} does not impact the validity of Propositions 1 and 2, its inputs do. In particular, \mathbf{f} has to be independent of $\{\mathbf{T}, \mathbf{u}, T^{out}\}$ for the first step of the proofs of both propositions to hold in general. If $\mathbf{f} = \mathbf{f}(\mathbf{x}, \mathbf{E})$ for example, the satisfaction of (2.19) would not be sufficient to guarantee the required nonnegativity of the partial derivatives in (2.18).*

Corollary 1 proves that each of the proposed multi-zone PCNN architectures remains physically consistent as long as all the parameters \mathbf{a}_h , \mathbf{a}_c , \mathbf{b} , and $\{c^{yz}\}_{z \in \mathcal{B}, y \in \mathcal{N}(z)}$ are small positive constants. Note that this makes intuitive sense: all these parameters correspond to inverses of resistances and capacitances, hence small positive numbers, in real buildings since the physics-inspired module is inspired by classical RC modeling techniques (see Appendix A.1.1). Interestingly, (2.26)–(2.28) can easily be enforced during training without modifying the classical BackPropagation Through Time (BPTT) algorithm, hence allowing us to rely on well-established ML tools to train our models, as detailed in Section 2.4.3.

Advantages of PCNNs

The strength of our approach lies in the fact that all the models will remain physically consistent whatever the structure of \mathbf{f} is, being shared or not,² composed of NNs or other nonlinearities. This gives the user complete freedom in the design of the black-box module without jeopardizing the consistency of the model. Similarly, all the parameters of the physics-inspired module might for example be time-varying or computed as nonlinear functions of external inputs without impacting the physical consistency of the model as long as they stay small and positive at all times.

As already mentioned in Remark 2, the very generic structure of PCNNs can also be applied to model complex phenomena beyond thermal modeling, typically where only part of the physics is well understood. Indeed, it is always possible to adapt the structure of the physics-inspired module, which might also include nonlinearities, let the black-box module capture completely unknown dynamics in parallel, and seamlessly learn everything simultaneously in an end-to-end pipeline. This allows one to take advantage of the power of representation of NNs while grounding their solution in existing domain expertise.

Remark 9 (A control perspective). *The PCNNs proposed in (2.17) are power input-affine. This makes such models interesting in control applications, typically for MPC schemes aimed at decreasing the energy consumption of buildings.*

²This is the main difference between the M-PCNN and S-PCNN architectures in our case, for example.

2.3.5 Training from data

Applying (2.17) recursively over the prediction horizon, starting from the measured temperature $\mathbf{T}(0)$, PCNNs can predict the evolution of the temperature in the building while satisfying the criteria in (2.1)-(2.3). One important key to the effectiveness and generality of PCNNs comes from the fact that all the parameters \mathbf{a}_h , \mathbf{a}_c , \mathbf{b} , $\{c^{yz}\}_{z \in \mathcal{B}, y \in \mathcal{N}(z)}$, and \mathbf{f} are learned simultaneously from data using BPTT. This allows us to alleviate the engineering burden associated with classical modeling techniques since PCNNs do not require any prior knowledge about the building structure or parameters beyond topology information.

Throughout this chapter, we assume access to a training data set of time series measurements $\mathcal{D} = \{(\mathbf{x}^{(s)}(0), \mathbf{u}^{(s)}(0), \mathbf{T}^{(s)}(0), T^{out,(s)}(0)), \dots, (\mathbf{x}^{(s)}(l_s), \mathbf{u}^{(s)}(l_s), \mathbf{T}^{(s)}(l_s), T^{out,(s)}(l_s))\}_{s=1}^N$, where l_s is the length and N the number of sequences in the training data, processed as detailed in Section 2.4.1. We then optimize all the parameters of both the physics-inspired and black-box modules together by minimizing the Mean Square Error (MSE) over the prediction horizon:

$$\begin{aligned} \min_{\mathbf{a}_h, \mathbf{a}_c, \mathbf{b}, \{c^{yz}\}_{z \in \mathcal{B}, y \in \mathcal{N}(z)}, \mathbf{f}} \quad & \mathcal{L}_{data} \\ \text{s.t.} \quad & (2.17) \end{aligned}$$

with

$$\mathcal{L}_{data} := \frac{1}{|\mathcal{Z}|} \sum_{s \in \mathcal{Z}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^m \left(\mathbf{T}_{k+1}^{z,(s)} - T^{z,(s)}(k+1) \right)^2 \right] \right], \quad (2.29)$$

where a batch \mathcal{Z} of series is randomly sampled from the training data at each iteration and we leverage PyTorch's BPTT implementation [140].

2.4 Presentation of the case study

To assess the quality of the PCNN architectures detailed in Section 2.3, we carry out extensive performance analyses on a case study, where the objective is to predict the temperature dynamics in a bedroom or the entire building over three days with 15 min time steps. Throughout the rest of this chapter, we assume direct thermal power input measurements to be available, hence setting $\mathbf{g}(\mathbf{u}) = \mathbf{u}$.

2.4.1 The Urban Mining and Recycling unit

We take advantage of NEST, a vertically integrated district located in Duebendorf, Switzerland, and pictured in Figure 2.8 [141]. NEST is composed of several residential and office units, and we focus our attention on the Urban Mining and Recycling (UMAR) unit, circled in white.

UMAR is an apartment composed of two bedrooms, with a living room in between them, and two small bathrooms that are neglected throughout this work. We are thus modeling



Figure 2.8: NEST building, Duebendorf, and the UMAR unit circled in white © Zooey Braun, Stuttgart.

three thermal zones arranged in a line in this chapter, i.e., Zone 2 is connected to Zones 1 and 3, and each zone has at least one external wall. All the rooms are equipped with radiant heating/cooling panels in the ceiling and controlled by opening and closing valves to let hot or cold water flow through them depending on the season.

We rely on three years of data collected between May 2019 and May 2022 and preprocessed as explained in Appendix A.4. This involved downsampling the data to 15 min intervals, smoothing the time series, and disaggregating the thermal power consumption of UMAR into the consumption of each zone.³ Besides these computed thermal power inputs, the data set also contains measurements of the temperature in each zone and outside, the horizontal solar irradiation on-site, time information, and the status of the system, i.e., whether it is in heating or cooling mode. The data has been split into nonoverlapping training and validation datasets, each containing up to 75 h-long time series.⁴

2.4.2 Benchmark models

To analyze the performance of the proposed PCNN architectures, we perform an extensive ablation study and compare them to state-of-the-art gray- and black-box methods. An overview of all the data-driven models used in this work, and whether they are physically consistent, can be found in Table 2.1.

³Since individual room power consumption measurements are not available, we approximated them by disaggregating the total consumption of UMAR using the design mass flows and the amount of time the valves in each room are open.

⁴This corresponds to the prediction horizon of three days plus the 3 h used to warm start NNs (see Section 2.4.3).

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

Category	Model	Physical consistency
Gray-box	<i>Linear</i>	✓
	<i>Res</i>	✗
	<i>Res-cons</i>	✓
Black-box	<i>ARX</i>	✗
	<i>ARX-KF</i>	✗
	<i>LSTM</i>	✗
	<i>PiNN</i>	✗
PCNNs	X-PCNN (Ours)	✓
	<i>M-PCNN (Ours)</i>	✓
	<i>S-PCNN (Ours)</i>	✓

Table 2.1: Physical consistency of the methods investigated in this work.

Gray-box baselines

Linear models. First, it makes intuitive sense to investigate the accuracy of the physics-inspired module of PCNNs on its own, leading to the following linear gray-box model architecture, hereafter referred to as the *Linear* models:

$$\begin{aligned} T_{k+1} = & T_k + \mathbf{a}_h \odot \max\{\mathbf{u}_k, \mathbf{0}\} + \mathbf{a}_c \odot \min\{\mathbf{u}_k, \mathbf{0}\} \\ & - \mathbf{b} \odot (T_k - T_k^{out}) - \Delta T_k + \mathbf{e} \odot \mathbf{Q}_k^{win}, \end{aligned} \quad (2.30)$$

where \mathbf{Q}_k^{win} gathers the solar irradiation on the windows of each zone in a vector, engineered from the measured irradiation on a horizontal surface as detailed in Appendix A.3. Since there is no black-box module taking care of the impact of the sun on building temperatures in this model, we indeed need to include it manually. This can be done efficiently for UMAR but does not generalize to arbitrary buildings when shading effects come into play, for example, limiting the applications of such linear models. As for the other heat gains, \mathbf{e} gathers the trainable scaling parameters reflecting the impact of solar gains on each zone temperature in a vector.

The classical least squares parameter identification procedure presented in Appendix A.1.2 was used to identify single-zone linear models, where \mathbf{a}_h and \mathbf{a}_c were not distinguished. In the single-zone case, the linear model has a sampling time of 1 min, we thus keep the power input fixed over intervals of 15 min when we compare its predictions with the ones of PCNNs.

This identification procedure however gave rise to physically inconsistent parameters in the multi-zone case, prompting us to instead identify \mathbf{a}_h^z , \mathbf{a}_c^z , \mathbf{b}^z , $\{c^{zy}\}_{y \in \mathcal{N}(z)}$, and \mathbf{e}^z for each zone z using Bayesian Optimization (BO), as detailed in Appendix A.5. As for X-PCNNs, the heat transfer coefficients between two adjacent thermal zones were then averaged based on (2.7).

Residual models. A natural extension of the aforementioned linear model is to consider *residual models*, where the idea is to fit the errors of the linear model predictions with a black-box module to improve its performance. Assuming the linear model in (2.30) to provide predictions \hat{T}_{k+1} , a residual model fits a function $f_{Res}: \mathbb{R}^{d'+2m+1} \rightarrow \mathbb{R}^m$, typically modeled with

NNs, to the residual errors. In other words, it minimizes

$$\mathcal{L}_{Res} := \frac{1}{|\mathcal{Z}|} \sum_{s \in \mathcal{Z}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^m \left(f_{Res}^z(T_k^{(s)}, \mathbf{x}_k^{(s)}, \mathbf{u}_k^{(s)}, T_k^{out,(s)}) - (T^{z,(s)}(k+1) - \hat{T}_{k+1}^{z,(s)}) \right)^2 \right] \right], \quad (2.31)$$

instead of \mathcal{L}_{data} . The residual model then predicts temperatures as

$$T_{k+1} = \hat{T}_{k+1} + f_{Res}(T_k, \mathbf{x}_k, \mathbf{u}_k, T_k^{out}). \quad (2.32)$$

This model is dubbed *Res* in the rest of this paper, and f_{Res} has the same structure as the proposed PCNNs' black-box module for fair comparisons.

Remarkably, such residual models cannot be guaranteed to respect the underlying physical laws in general since f_{Res} is not independent of zone temperatures, power inputs, and ambient temperatures. Since they can be seen as being composed of a physics-inspired and a black-box module in (2.32), similarly to PCNNs, we can indeed use similar arguments to prove their physical consistency and hence derive similar necessary conditions on the structure of f_{Res} as the one discussed in Remark 8.

Consequently, we also investigate the performance of a *physically consistent* residual model in this work, dubbed *Res-cons*, where the black-box function learning the residuals only depends on \mathbf{x} , as PCNNs. It fits a function $f_{Res-cons} : \mathbb{R}^{d'} \rightarrow \mathbb{R}^m$ to the residuals, trained similarly to its physically inconsistent counterpart, with the following temperature predictions:

$$T_{k+1} = \hat{T}_{k+1} + f_{Res-cons}(\mathbf{x}_k). \quad (2.33)$$

Note that residual models first fit the base model to the data and then use black-box methods to fit the residual errors while PCNNs learn both modules together. This also implies that the physics-inspired module reflects the main dynamics of residual models, with small corrections from the NN on top, while it only ensures the physical consistency of PCNN architectures, letting more expressive functions like NNs capture the main system dynamics.

Black-box baselines

Autoregressive model with exogenous inputs. As a first black-box method, we analyze the performance of an AutoRegressive model with eXogenous inputs (ARX model), where autoregressive lags of the states and inputs are used to predict the next state:

$$T_{k+1} = \alpha_0 T_k + \alpha_1 T_{k-1} + \dots + \alpha_\delta T_{k-\delta} + \beta_0 \hat{\mathbf{x}}_k + \beta_1 \hat{\mathbf{x}}_{k-1} + \dots + \beta_\delta \hat{\mathbf{x}}_{k-\delta}, \quad (2.34)$$

$$\hat{\mathbf{x}}_k = [\mathbf{u}_k, T_k^{out}, Q_k^{sun}]^T,$$

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

where $Q_k^{sun} \in \mathbb{R}$ is the solar irradiation measurement on a horizontal surface, and the parameters $\alpha_0, \dots, \alpha_\delta \in \mathbb{R}^{m \times m}$, $\beta_0, \dots, \beta_\delta \in \mathbb{R}^{m \times (m+2)}$ are identified through least square regression using the `scikit-learn` library [142]. For a fair comparison, we set $\delta = 11$, i.e., we use information from the last 3 h to define the next temperatures, the same amount of information provided to warm-start NNs (see Section 2.4.3).

We also implemented an advanced ARX model relying on the `statsmodels` package [143], which includes Kalman smoothing and filtering operations out-of-the-box, dubbed *ARX-KF*, to compare PCNNs to an existing toolbox. We set $\beta_1, \dots, \beta_\delta = 0$ so that only current information on external inputs is used. Since the identification procedure was harder in that case, we identified each zone z separately, with:

$$\hat{\mathbf{x}}_k^z = [\mathbf{u}_k^z, T_k^{out}, Q_k^{sun}, T_k^{y_1}, T_k^{y_2}, \dots, T_k^{y_{|\mathcal{N}(z)|}}]^T,$$

where $y_1, \dots, y_{|\mathcal{N}(z)|} \in \mathcal{N}(z)$ are the zones adjacent to z .

Note that ARX models cannot be enforced to be physically consistent for multi-step-ahead predictions in general due to the autoregressive terms; they transform (2.26)-(2.28) into highly nonlinear constraints. Although it is thus hard to assess the physical consistency for ARX models in practice, we could observe that automatic identification of the ARX-KF model assigned a negative scaling parameter for the power input to Zone 2, meaning that heating this zone will lead to lower temperatures. This is sufficient to confirm ARX-KF is *not* physically consistent — and similar issues were found for the classical ARX model.

Remark 10 (Kalman filtering and smoothing). *Kalman filtering and smoothing operations could be included during the data processing phase, potentially impacting the performance of all the models. In this chapter, however, we focus on methods working on unfiltered data, typically involving NNs. Nonetheless, we also provide ARX-KF as an example of what can be achieved with existing toolboxes on a laptop compared to NN-based methods that might require access to Graphical Processing Units (GPUs) for training.*

Neural Network models. As another natural ablation of PCNNs, we also investigate the quality of the black-box module alone. Instead of treating the power inputs and temperatures in a separate module, everything is fed in the black-box function $f_{LSTM}: \mathbb{R}^{d'+2m+1} \rightarrow \mathbb{R}^m$, leading to the *LSTM* model

$$\mathbf{T}_{k+1} = \mathbf{T}_k + f_{LSTM}(\mathbf{T}_k, \mathbf{u}_k, \mathbf{x}_k, T_k^{out}). \quad (2.35)$$

As expected, such classical NN-based methods are naturally physically inconsistent and might fail to capture the underlying physical laws even if they fit the data well (see Section 1.2). For fair comparisons, the *LSTMs* considered in this work have the same encoder-LSTM-decoder architecture as the corresponding PCNNs.

Finally, we also implemented standard physics-informed NNs, hereafter *PiNNs*, again relying on the same architecture as the black-box modules of PCNNs. However, their loss function is

modified to steer the learning toward physically meaningful solutions:

$$\mathcal{L}_{PiNN} = \mathcal{L}_{data} + \lambda \mathcal{L}_{phys}, \quad (2.36)$$

where λ is a tuning hyperparameter. Since the purpose of \mathcal{L}_{phys} is to capture physical inconsistencies and penalize them, we naturally design it to bias the model towards solutions satisfying the desired properties (2.1) and (2.2). Consequently, we penalize negative gradients of the final predicted temperatures, i.e., at time l_s , with respect to control inputs and ambient temperatures observed along the horizon:

$$\mathcal{L}_{phys} = \frac{1}{|\mathcal{Z}|} \sum_{s \in \mathcal{Z}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^m \left(\sum_{y=1}^m \left[r \left(-\frac{\partial T_{l_s}^{z,(s)}}{\partial \mathbf{u}_k^{y,(s)}} \right) \right] + r \left(-\frac{\partial T_{l_s}^{z,(s)}}{\partial T_k^{out,(s)}} \right) \right) \right] \right], \quad (2.37)$$

where $r(x) = \max\{x, 0\}$, also known as the Rectified Linear Unit (ReLU) function. Since we are interested in physically consistent models in this work, we empirically fixed $\lambda = 100$, which ensures the loss term is dominated by the physical inconsistencies, thereby steering the PiNN towards interesting solutions. The nontrivial tuning of this hyperparameter [91] is left for future work.

Note that, in building temperature modeling, one can also augment the outputs of NNs to predict not only zone temperatures but also the temperatures of their respective thermal mass, for example, and then penalize deviations of the latter from the predictions of a physics-based model in \mathcal{L}_{phys} to incorporate prior knowledge in PiNNs [134]. However, this requires access to a physics-based model, introducing engineering overhead. Moreover, it can enforce unwanted biases since the physics-based model might be inaccurate and steer PiNN predictions away from the truth. Consequently, in this work, we penalize the gradients of the temperature predictions instead, according to our definition of physical consistency in Section 2.3.1, which bypasses the need for a physics-based model and only relies on measured quantities while still incorporating knowledge about the underlying laws of physics in PiNNs.

Remark 11 (Computational complexity). *To ensure a model is following the underlying physical laws at all times, one should check the gradients throughout the prediction horizon, and not only for the last predictions, as proposed in (2.37). However, since each gradient computation requires one forward and one backward pass of the data, the computational complexity grows linearly with the number of predictions to analyze. Consequently, we only compute the gradients of the last temperature predictions with respect to all the control inputs and ambient temperatures observed along the horizon to steer PiNNs toward expected solutions, alleviating the associated computational burden.*

2.4.3 Implementation details

Throughout our case study, we assume direct access to the thermal power of each room, hence selecting \mathbf{g} to be the identity mapping, which naturally satisfies Condition (2.28).

Physics-inspired parameters

To ensure the physical consistency of the proposed multi-zone PCNNs, fulfilling the conditions (2.26) and (2.27), we parametrize the log value of each parameter. In other words, we learn $\tilde{a}_h^z, \tilde{a}_c^z, \tilde{b}^z, \tilde{c}^{zy}, \forall z \in \mathcal{B}, \forall y \in \mathcal{N}(z)$ and define:

$$w = w_0 \exp(\tilde{w}), \quad \forall w = \{a_h^z, a_c^z, b^z, c^{zy}\}, \quad (2.38)$$

where w_0 is the initial value of the parameter. Starting from $\tilde{w} = 0$, PCNNs hence learn to scale the initial value w_0 instead of modifying it directly, which is numerically more stable and ensures that w stays small enough. On the other hand, the exponential function keeps all the parameters positive at all times. Note that they are learned simultaneously to the parameters in the black-box module: when backpropagation is used to update the parameters of the NNs, we also leverage the propagated gradients to update the parameters of the physics-inspired module.

As a consequence of this choice, the initialization of the physics-inspired parameters, i.e., the choice of w_0 , is critical. Indeed, as they are inspired by the known physics of buildings, they must correspond to meaningful values. Furthermore, due to the recurrent use of these parameters to modify the state of the energy accumulator along the prediction horizon, wrong values would have a large impact on the quality of the model and the PCNNs might get stuck in local minima. In practice, we saw that using rules of thumb to initialize those parameters to plausible values using our prior knowledge led to good results, as presented in Section 2.5.1. In particular, we define initial values such that, $\forall z \in \mathcal{B}$:

- For a_h^z and a_c^z : The temperature in the zone rises/drops by 1 °C in 2 h when the 1000 W heating/cooling power is applied.⁵
- For b^z and c^{zy} : The temperature drops by 1.5 °C in 6 h when the exogenous temperature is 25 °C lower.

These rules of thumb can be derived from historical data, for example looking at how much time it generally takes for the temperature to rise by 1 °C when the zone is heated with 1000 W for a_h^z , respectively to drop by 1.5 °C when it is 25 °C colder outside and heating is off for b^z . Similar investigations will give plausible initial values for c^{zy} and a_c^z .

Remark 12 (Upperbound on s). *While \tilde{b}^z or \tilde{c}^{zy} can in principle grow uncontrollably and lead to a violation of the necessary condition (2.26), this was not an issue in our experiments. Nonetheless, if required, one can introduce bounds on the learned values \tilde{s} , typically leveraging activation functions like the sigmoid or hyperbolic tangent to control their range.*

⁵In the single-zone case, 1000 W was replaced by the maximum thermal power, around 1.6 kW.

Black-box architecture

In the multi-zone case, each function f has the same encoder-LSTM-decoder architecture, which is repeated when several modules are required for X-PCNNs and M-PCNNs. Both the encoder and decoder are feedforward NNs with 32 hidden units and the LSTM comprises two layers with dimension 64 and is followed by a normalization layer. This architecture was selected over larger ones since we did not observe any significant decrease in performance. However, for single-zone PCNNs — simply referred to as *PCNNs* hereafter —, both the encoder and decoder have two layers of 128 units and the LSTM is composed of two layers of size 512 to ensure the black-box module is expressive enough.⁶

The input of each black-box module, $\mathbf{x} \in \mathbb{R}^6$, gathers the solar irradiation on a horizontal surface and the time information (see Appendix A.4). We let the initial hidden and cell states of the LSTM be learned during training and additionally give the model a warm start of 3 h. In other words, NNs first predict the last 12 time steps in the past, where we feed the true temperatures back to the network to initialize all the internal states, before predicting the temperature over the given horizon. This was empirically shown to improve performance. NNs share a common learning rate of $5e-4$, manually selected small enough to ensure stable convergence, and a batch size of 4'096, to maximize the utility of the GPUs.

Performance assessment

To validate the performance of each model, we rely on the Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE):

$$MAE = \frac{1}{|\mathcal{Z}|} \sum_{s \in \mathcal{Z}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^m |\mathbf{T}_{k+1}^{z,(s)} - T^{z,(s)}(k+1)| \right] \right] \quad (2.39)$$

$$MAPE = \frac{1}{|\mathcal{Z}|} \sum_{s \in \mathcal{Z}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s-1} \left[\frac{1}{m} \sum_{z=1}^m \frac{|\mathbf{T}_{k+1}^{z,(s)} - T^{z,(s)}(k+1)|}{T^{z,(s)}(k+1)} \right] \right]. \quad (2.40)$$

The PCNNs are implemented in PyTorch [140] and were trained on NVIDIA P100 GPUs.

Code status

An up-to-date version of PCNNs can be found on <https://github.com/Cemempamoi/pcnn>, while the versions used in [39] and [40] are referenced in the respective papers.

⁶We later carried out a small experiment to assess the impact of the NN architecture on multi-zone PCNNs and could decrease its size for subsequent experiments, leading to the smaller architecture of dimensions 32–64.

2.5 Performance analysis and comparison to other methods

In this section, we present various results obtained from the previously detailed case study. We first analyze the performance of single-zone PCNNs trained on data from Zone 3, detailing their behavior, before providing an in-depth comparison of multi-zone PCNNs with other state-of-the-art data-driven methods in modeling the thermal behavior of the entire UMAR unit.

2.5.1 Investigating the behavior of single-zone PCNNs

To understand the behavior of PCNNs, we start by investigating their performance on a single-zone modeling task, focusing on Zone 3 of UMAR. All the results discussed hereafter were obtained by comparing the multi-step prediction performance of the different models on almost 2'000 three-day-long time series from the validation set. Each model is recursively applied to predict the temperature for 288 steps (three days) assuming knowledge of all the inputs, and compared to the true measured temperatures.

While we only discuss one PCNN in depth throughout this section — the one that achieved the lowest validation loss in Table 2.2 —, Appendix A.6 provides additional insights, analyzing the impact of the random seed and choice of thermal zone to model. The results are consistent in all cases, hinting at the robustness, respectively the flexibility, of the proposed approach.

Improving the generalization issue of NNs

As presented in Table 2.2, while PCNNs could not attain the performance of classical LSTMs on the training data due to their constrained structure to follow the underlying physical laws, they obtained lower errors on the validation set. This **confirms that PCNNs solve part of the generalization issue of classical NNs**, having a smaller tendency to overfit the training data but retaining enough expressiveness to perform well on previously unseen data. In other words, the physics-informed module inside PCNNs seems to give them useful information, allowing them to beat the performance of classical unconstrained LSTMs.

Since physical consistency is required for control-oriented thermal models, as discussed in Section 2.1, and LSTMs fail to satisfy this criterion, as pictured in Figure 2.1, we focus on comparing the best-found PCNN from Table 2.2 to the linear baseline model hereafter. This analysis is aimed at providing insights into the significance of the black-box module running in parallel to the physics-inspired base to capture nonlinear effects without jeopardizing the needed physical consistency of the model.

2.5 Performance analysis and comparison to other methods

Seed	<i>LSTMs</i>		<i>PCNNs (ours)</i>	
	Training loss	Validation loss	Training loss	Validation loss
0	0.57	2.28	1.83	1.93
1	0.57	1.92	1.85	1.65
2	1.14	2.30	2.06	1.75
3	0.97	2.22	2.28	1.73
4	1.00	1.77	1.90	1.97
Mean	0.85	2.10	1.99	1.81

Table 2.2: Comparison of the training and validation loss for five LSTMs and PCNNs, scaled by 10^3 .

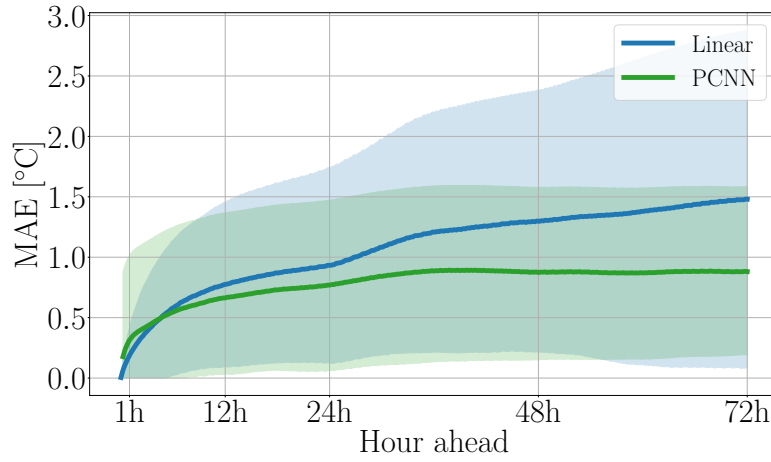


Figure 2.9: Mean and standard deviation of the error at each time step of the prediction horizon for both the RC model in blue and the PCNN in green, where the statistics were computed from almost 2000 predictions from the validation set.

Performance improvement compared to the linear model

Since predicting the evolution of the temperature for several time steps ahead entails a recursive use of the architecture in Figure 2.4, we leverage the ability of LSTMs in the black-box module to handle long sequences of data to minimize the error over long horizons. On the other hand, the linear baseline was fitted to optimize the single-step accuracy, as commonly done in the literature. However, this leads to error propagation over time, as pictured in Figure 2.9, where we plotted the MAE and one standard deviation for both models over the validation set, i.e., unseen data. Note that while the physics-based baseline used here is not optimal, it was nonetheless tuned to obtain good accuracy, with an average error below 1°C after 24 h.

One can observe the PCNN providing better predictions than the baseline in general, which is supported by the MAEs reported at key points along the horizon in Table 2.3. In particular, the PCNN can keep a good accuracy even on long horizons, with an error more than 40% lower

Hours ahead	Linear model	PCNN (ours)
1 h	0.19 °C	0.31 °C
6 h	0.58 °C	0.55 °C
12 h	0.78 °C	0.66 °C
24 h	0.93 °C	0.77 °C
48 h	1.30 °C	0.88 °C
72 h	1.48 °C	0.88 °C

Table 2.3: Comparison of the MAE of the two models over the prediction horizon.

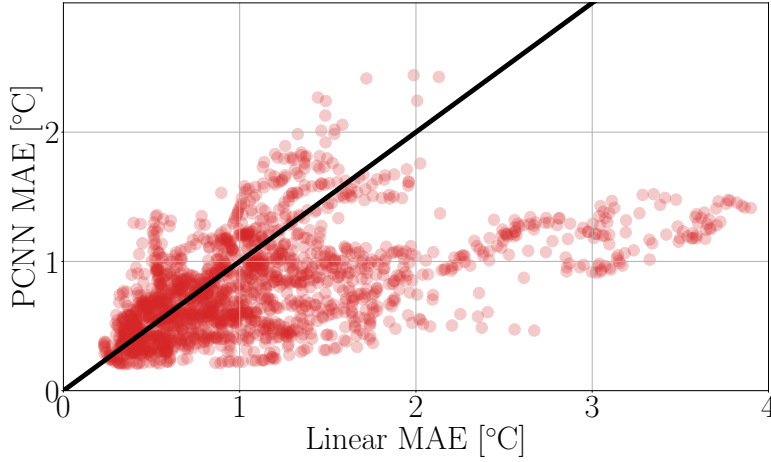


Figure 2.10: Scatter plot of the MAEs of both models on each test sequence, with the black diagonal line representing equal performance.

than its counterpart after three days. On the other hand, it presents slightly higher errors at the beginning because of the implemented warm-start (see Section 2.4.3): since they firstly predict past data — the last 3 h — PCNNs might indeed start predicting the three-day-long validation sequences from a temperature different from the true one. Nonetheless, since we observed that the warm start benefited the overall performance of PCNNs during our experiments, we kept it in the final implementations.

To investigate the MAEs obtained by both models on each validation sequence, we also provide the corresponding scatter plot in Figure 2.10. In general, the PCNN dominates the baseline, with its error rarely being significantly larger than the one of the physics-based model, which would be represented by points over the black diagonal line. On the other hand, towards the lower right side of this figure, we find sequences where the PCNN presents a significantly better accuracy than the linear model.

This superiority of the PCNN over the linear RC baseline is confirmed by the error distributions of both models in Figure 2.11, with the errors of the PCNN (green) clustered below 1 °C and almost always below 2 °C while the errors of the baseline in blue are much more spread out. This indicates that the PCNN is robust with respect to different inputs, even on unseen data.

2.5 Performance analysis and comparison to other methods

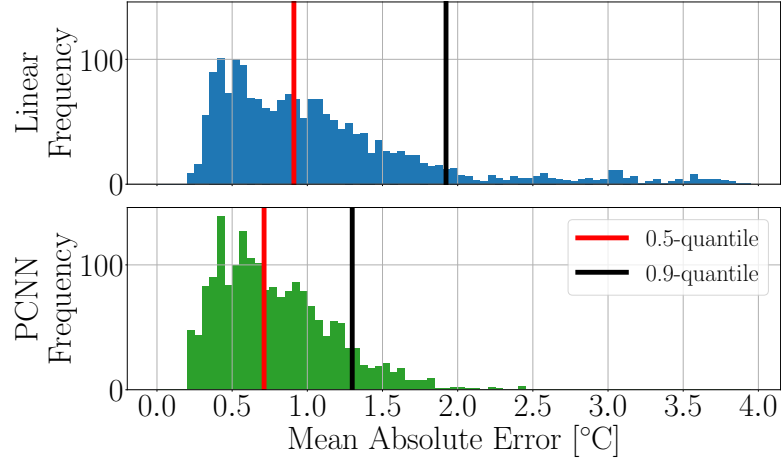


Figure 2.11: Distribution of the MAE of both models over the test sequences, with the 50% and 90% quantiles marked in red, respectively black.

Altogether, we can conclude that this PCNN is less prone to extreme errors and keeps the majority of errors lower than the linear physics-based baseline, proving its robustness and effectiveness. Remarkably, all the results were obtained over three years of data, hence under various weather conditions and during all the seasons, which also hints that exogenous variables do not impact the quality of the model much. Overall, it shows the effectiveness of training both modules simultaneously over long horizons leveraging BPTT to improve upon classical physics-based techniques. This would be especially valuable when engineering the solar gains of a thermal zone from the global solar irradiation measurements — as done for the linear model in Appendix A.3 — would become cumbersome and lead to even higher errors for classical engineering-based models.

Remark 13 (Link to Chapter 4). *In the case of UMAR, as discussed in Appendix A.3, we can accurately model solar gains, such that a linear model of the form (2.30) might achieve an accuracy similar to the one of PCNNs treating solar gains as black-box inputs. Consequently, we suspect the training procedure to have a significant impact on the observed performance gap between both models, hinting at the potential of BPTT to identify structured (here gray-box) models. This hypothesis will be analyzed in-depth in Chapter 4, confirming the ability of BPTT to improve upon traditional SI techniques in such settings where prior knowledge is available.*

Empirical analysis of the physical consistency of single-zone PCNNs

After the detailed analysis of PCNN's performance, let us now provide an empirical and visual examination of the physical consistency of PCNNs, formally proven in Section 2.3.4, to showcase how PCNNs retain physical consistency even on *unseen* data, avoiding the classical generalization issue of NNs discussed in Section 2.1. To that end, we randomly sampled a sequence from the validation data set and compared the temperature predictions of both the

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

linear baseline and the PCNN analyzed above in Figure 2.12 when:

- the original and true thermal power inputs are applied (blue),
- no power is used (black), hereafter named *uncontrolled*,
- only the first half of the power inputs are used (red),
- only the second half of the input is applied (orange),

where we separate the power inputs in half with respect to their magnitudes, i.e., so that both the red and orange control sequences apply roughly the same total power. For reference, we also added the ground truth in dashed blue.

Firstly, comparing the blue predictions with the dashed ground truth, we see both models performing well, exemplifying the results discussed above. Remarkably, the proposed PCNN is able to accurately match the ground truth despite the large amount of heating power applied and the temperature rising to more than 30 °C, something unusual in real settings and hence not well covered by the training data.

Furthermore, looking at the three other cases, for which there is no ground truth anymore, both models again show similar behaviors, the visual consequence of both of them being physically consistent. The red predictions deviate from the blue ones at the same point in time for both models since, as expected, we should get lower temperatures as soon as we stop heating the zone. Similar conclusions can be drawn by comparing the orange and black temperature predictions. Finally, looking at the uncontrolled predictions, one can observe smoother patterns for the PCNN due to the unforced base dynamics being captured by LSTMs instead of the more aggressive linear regression at the core of the linear model.

To clarify the differences between both models' physical behaviors, we can subtract the uncontrolled predictions from the other curves for both models. The result is pictured in Figure 2.13 and allows us to assess the impact of the three different control sequences on the final predictions. As expected, both models still exhibit similar behaviors, with predictions diverging from the uncontrolled dynamics — from zero in Figure 2.13 — as soon as the heating is turned on. On the other hand, when it is turned off, the gap slowly closes (second half of the red curve) because of the higher inside temperature leading to higher energy losses to the environment and the neighboring zone. Note that the impact of the latter is hard to distinguish here since it is an order of magnitude smaller than the losses to the outside.

One can empirically assess the difference in the parameters a_h , a_c , b , and c learned by both models leveraging plots such as Figure 2.13. For example, a_h is smaller for the PCNN since the differences with the uncontrolled dynamics are generally smaller. In other words, heating has a smaller impact on the PCNN zone temperatures. Despite not being visible in that plot, a similar conclusion can be drawn about the cooling parameter a_c . Concerning heat losses, the PCNN learned parameters b and c entailing roughly the same amount of energy transfer as the baseline, which is particularly observable in the red curves having a similar slope in both cases after one day. However, we cannot separate the effect of b and c in these plots since

2.5 Performance analysis and comparison to other methods

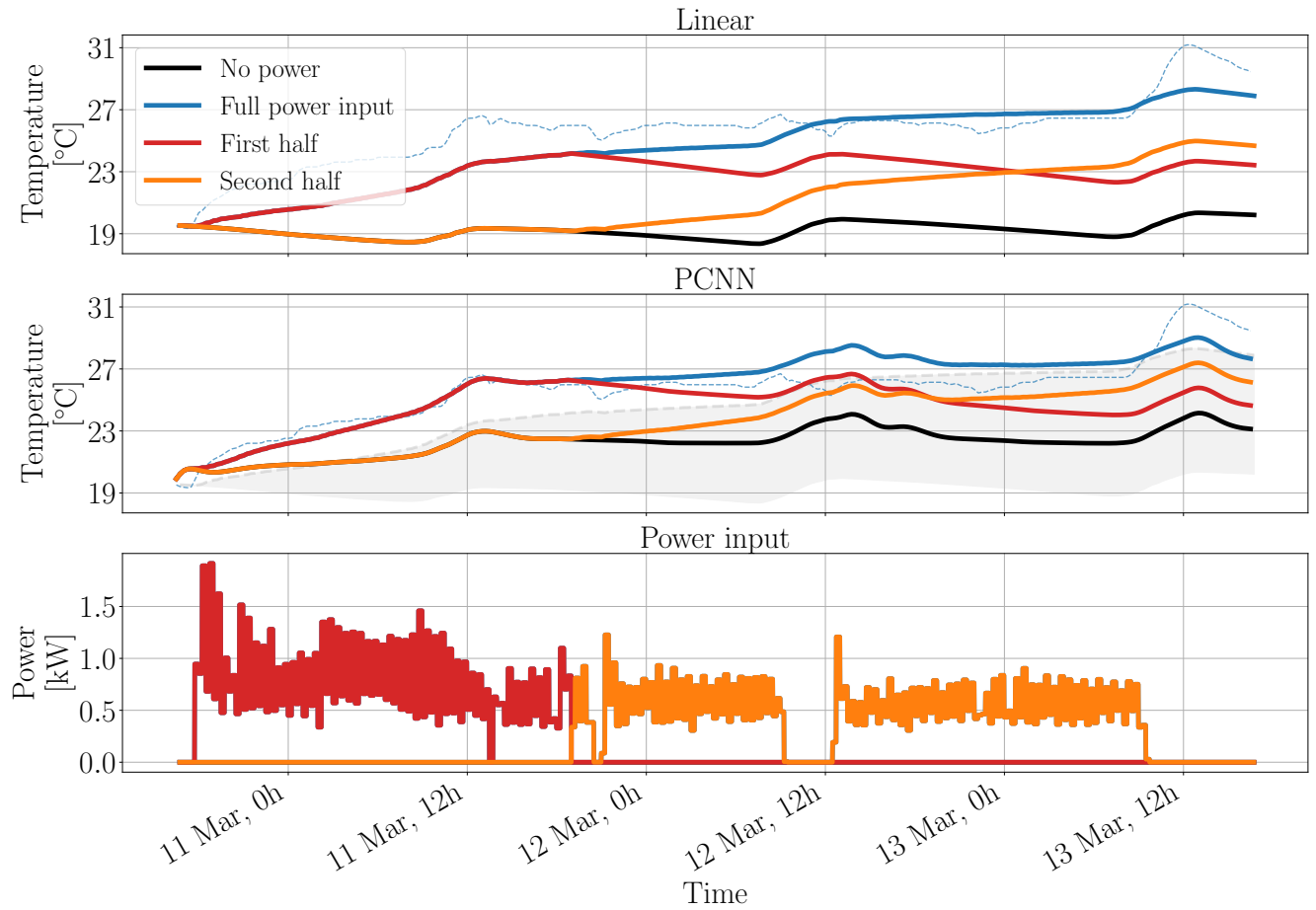


Figure 2.12: Comparison between the linear model (top) and the PCNN (middle) temperature predictions given the bottom heating power inputs, over three days. In blue, one can assess the precision of both models compared to the ground truth (dashed) when the original total power inputs is used. Then, the red and orange curves show the result when heating is only turned on roughly during the first day, respectively the second and third one. Finally, the black uncontrolled dynamics reflect the case when no heating is applied. For comparison purposes, we shaded the span of the linear model predictions in the middle plot.

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

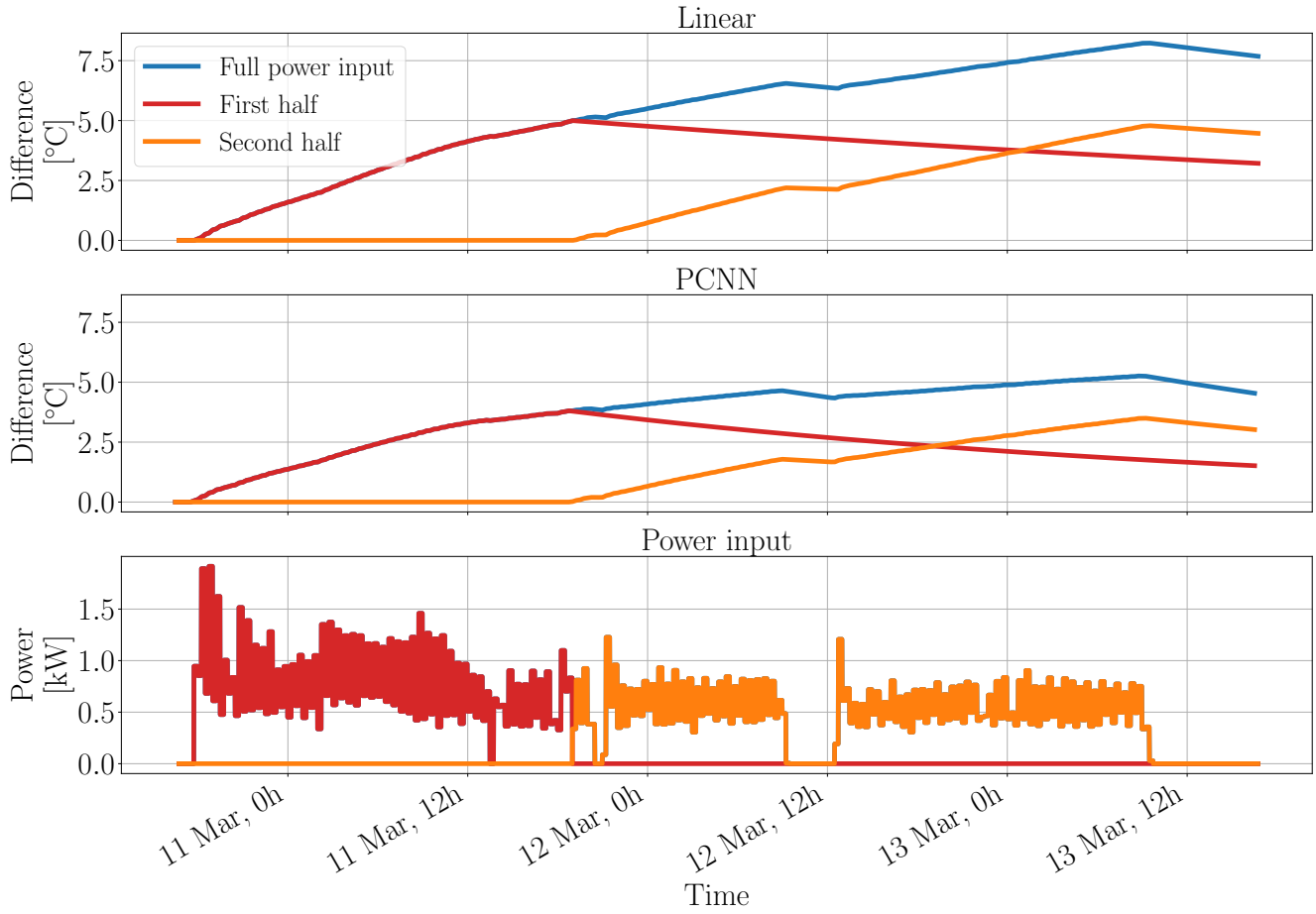


Figure 2.13: Difference between each temperature prediction and the black baseline (no power) in Figure 2.12, for the linear model (top) and the proposed black-box structure (middle), given the bottom power inputs. Note that the blue curve corresponds to the use of the full power input sequences, i.e., both the red and orange ones.

losses to the outside and the neighboring zone are lumped together. Each model could indeed assign more importance to one or the other energy transfer, and further analyses would be required to investigate and clarify this relation between b and c .

2.5.2 Benchmarking multi-zone PCNNs

While accurate zone temperature models are important, in practice, we are often interested in modeling the thermal dynamics of a whole building. Consequently, this section provides a comprehensive analysis of the methods presented in Section 2.4.2 and Table 2.4 to model the three zones in UMAR simultaneously. Similarly to the single-zone case, all the results discussed hereafter were obtained by comparing the multi-step prediction performance of the different models on more than 750 three-day-long time series from the validation set. To incorporate the fact that UMAR consists of three thermal zones, the MAE of a model is then defined as its average performance over these zones. Altogether, this will allow us to understand the trade-offs between the physical consistency, accuracy, and computational complexity of the various examined data-driven building modeling methods examined.

Robustness to randomness. For consistency, all the NN-based models⁷ were run with several random seeds. Remarkably, it does not impact performance significantly, with standard deviations in the range of 0.01 – 0.04 and 0 – 0.2% for the MAE and MAPE, respectively, as detailed in Table 2.4. This corresponds to a variation in performance of maximum 3% and is often smaller than the observed discrepancy in accuracy between different models, hinting that the observed performance gaps are significant.

As for the single-zone case in Section 2.5.1, this exemplifies the robustness of PCNNs, which do not seem significantly impacted by the random seed in general, or at least similarly to classical NNs. In this specific case study, the X-PCNN and M-PCNN architectures seem slightly more robust than the S-PCNN one, but the latter sometimes outperforms M-PCNNs. Overall, X-PCNNs seem to have the upper hand, attaining consistently high performance even under different random seeds, but more analyses with different data sets and case studies have to be conducted before drawing any definitive conclusion.

Overall performance comparison

Unless stated otherwise, the results discussed hereafter were obtained by the best-performing seed for each model, achieving the errors reported in Table 2.4. As can be observed, all the proposed PCNN architectures attain state-of-the-art accuracy, both in terms of MAE and MAPE. They are followed by physically inconsistent black-box methods, especially the ones

⁷The LSTM, PiNN, and PCNN architectures. Despite also being composed of an NN, residual models are not considered as *NN-based* models in this work since their main dynamics are still captured by the underlying linear model, and not the NN.

⁸While the LSTM and S-PCNN models were run on five seeds due to their slightly higher sensitivity, the other results were obtained over three seeds leading to very consistent performance.

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

Category	Model	Physical consistency	Best MAE	Best MAPE	MAE distribution	MAPE distribution
Gray-box	<i>Linear</i>	✓	1.79	7.5%	-	-
	<i>Res-cons</i>	✓	1.50	6.4%	-	-
PCNNs	<i>X-PCNN (Ours)</i>	✓	1.17	4.9%	1.18 ± 0.01	$5.0\% \pm 0.0\%$
	<i>M-PCNN (Ours)</i>	✓	1.25	5.3%	1.26 ± 0.01	$5.4\% \pm 0.0\%$
	<i>S-PCNN (Ours)</i>	✓	1.22	5.1%	1.27 ± 0.04	$5.4\% \pm 0.2\%$
Gray-box	<i>Res</i>	✗	1.79	7.7%	-	-
Black-box	<i>ARX</i>	✗	1.68	7.1%	-	-
	<i>ARX-KF</i>	✗	1.35	5.6%	-	-
	<i>LSTM</i>	✗	1.27	5.5%	1.33 ± 0.04	$5.7\% \pm 0.2\%$
	<i>PiNN</i>	✗	1.37	5.8%	1.38 ± 0.01	$5.9\% \pm 0.1\%$

Table 2.4: Physical consistency, best MAE, and best MAPE of the methods investigated in this work. The mean performance of the five NN-based model architectures, as well as the corresponding standard deviation, over three to five runs,⁸ is also reported. Note that the linear and LSTM models correspond to learning only the physics-inspired module of S-PCNNs, respectively the black-box one. Furthermore, *Res-cons* is equivalent to fitting both modules of S-PCNNs sequentially, showcasing the importance of learning all the parameters of PCNNs simultaneously to attain state-of-the-art accuracy.

relying on very expressive NNs. As expected, the least expressive class of methods, the gray-box one, performs the worst. Combining these results with the physical consistency of each method, we can conclude that the proposed PCNN architectures indeed take the best out of both gray- and black-box methods in this case study, **attaining the best performance while respecting the underlying physical laws, without trade-off, hinting at their potential to become state-of-the-art thermal building models.**

While X-PCNNs achieve the lowest error here, we suspect this performance to be influenced by the analyzed case study. Indeed, the temperature dynamics in UMAR are strongly impacted by solar gains, which reduces the importance of energy exchanges between the zones. This might explain why it is possible to fit the overall building dynamics well even when independently training one model for each zone and why the *post hoc* correction (2.7) does not seem to have any significant impact on the final model performance. We suspect this required correction might have a stronger influence on multi-zone buildings where temperature dynamics are less impacted by weather conditions and more governed by energy exchanges between the zones, which may, in turn, decrease the quality of X-PCNNs.

Remarkably, both residual models (*Res* and *Res-cons*), while conceptually close to PCNNs,⁹ are unable to achieve similar performance. This hints towards the **benefits of learning all the parameters together in an end-to-end fashion instead of first identifying the linear part and then fitting the residual errors.**

⁹Especially *Res-cons*, where the only difference in architecture comes from the solar irradiation processing.

2.5 Performance analysis and comparison to other methods

Finally, these results suggest that the black-box modules of PCNNs can process raw solar irradiation data and infer its impact on zone temperatures. Indeed, PCNNs outperform gray-box models, which have access to engineered solar gains (Appendix A.3). Interestingly, since the impact of the sun is implicitly computed by NNs, PCNNs could easily be applied to any building, even when shading comes into play, making the engineered processing of solar data required for gray-box models much more complex.

Physical consistency can be helpful for Neural Networks. Remarkably, in this case study, enforcing the physical consistency of LSTMs, as in PCNNs, seems to improve their accuracy despite the introduced constraints, confirming the benefits of the ongoing trend in ML research to include prior knowledge in NN architectures. Even if it might intuitively seem that introducing structural constraints should hinder the expressiveness of NNs, our results add to the literature suggesting that it can on the contrary be helpful. Moreover, one can draw similar conclusions with the two residual models investigated in this work, with *Res-cons* clearly outperforming its physically inconsistent counterpart despite both models relying on the same linear basis. Altogether, this points towards the **advantages of grounding NN architectures in the underlying physics to ensure that they learn meaningful solutions**.

Comparing physically consistent methods. PCNNs are on average 30 – 35% and 17 – 22% more accurate than the other physically consistent methods, namely the *Linear* and *Res-cons* model, respectively. The MAE of the physically consistent models over three days is plotted in Figure 2.14, showing that the proposed PCNNs not only perform better *on average* but along the entire prediction horizon, except during the first few hours, where all models attain similar performance. The main reason behind this behavior is the warm start of PCNNs, which often gives erroneous first predictions, but they offer much stronger performance in the long run. **At the end of the horizon, PCNNs indeed present an error 34 – 41% and 10 – 18% lower than *Linear* and *Res-cons*, respectively**, with the best performance again achieved by the X-PCNN.

The necessity of physical consistency

Knowing that physical consistency was beneficial in our case study, PCNNs even outperforming black-box methods, let us now visualize the thermal behavior of one S-PCNN and one PiNN in Figure 2.15. For this analysis, the thermal power is turned off in Zone 1 and 2 and we examine the impact of heating (red), cooling (blue), or providing no power input (black) in Zone 3. Note that the heating pattern corresponds to the true power inputs measured in Zone 3 in March 2021, which we mirrored to create the cooling pattern. This is similar to what was performed in Figures 2.1 and 2.12 in the single-zone case, but we now also expect physically meaningful energy exchanges between the connected thermal zones.

As one can immediately realize, following the laws of thermodynamics, heating or cooling Zone 3 increases or decreases its temperature, respectively, in the S-PCNN model. This effect is then propagated to the adjacent Zone 2, and later to Zone 1, impacting their temperatures even though they are neither heated nor cooled, as anticipated. Note that while only the

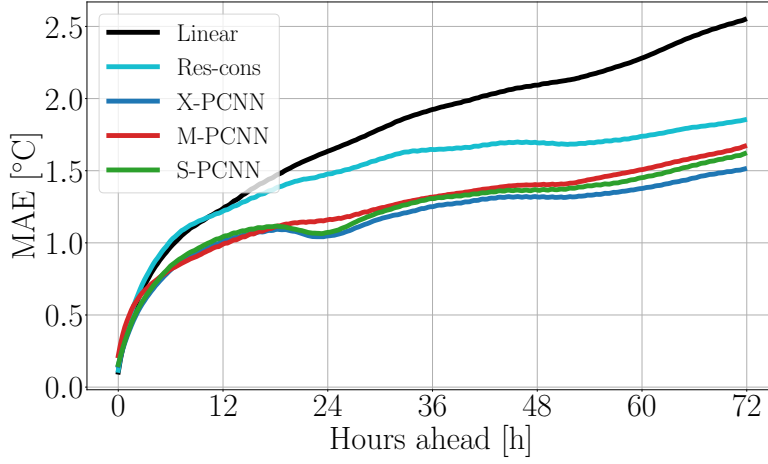


Figure 2.14: MAE of all the physically consistent methods over the prediction horizon averaged over the three zones and the time series of three days in the validation data set.

temperature predictions of one S-PCNN are pictured in Figure 2.15, similar effects were observed for all the PCNNs we trained. This is expected since all of them share the same physics-inspired module to ensure they follow criteria (2.1)-(2.3).

On the other hand, all power inputs lead to almost indistinguishable temperature predictions for the PiNN and the LSTM plotted in Appendix A.7. Despite fitting the data well (see Table 2.2), these models are hence obviously flawed and can be misleading in practical applications. We can sometimes even observe lower temperature predictions when heating is turned on than when the zones are cooled, a clear sign of physical inconsistency (see Appendix A.7 for zoomed-in results). This exemplifies the issue of **shortcut learning in the case of thermal modeling, where NNs manage to fit the data well without respecting the underlying physics**.

Interestingly, *Res* did capture a much more significant impact of heating and cooling but remains completely oblivious to the underlying physics, with cooling often resulting in higher temperatures than heating. This illustrates the need to consider physical consistency when designing residual models as well, such as in the proposed *Res-cons* architecture. In general, all these observations suggest that **physical consistency should always be considered when dealing with NNs for physical systems**.

PiNNs cannot guarantee physical consistency. This analysis illustrates how PiNNs only *steer* the learning towards interesting solutions without providing any guarantees concerning the actual behavior of the model. In fact, trained PiNNs always gave very similar predictions to LSTMs in our experiments, as can also be seen by comparing Figures A.4 and A.5. This hints that the additional physics-inspired loss term in \mathcal{L}_{PiNN} did not have much impact on the final solution found despite the large λ used. While tuning this hyperparameter might lead to better results, it is a cumbersome task and would still never guarantee the physical consistency of the final model, as discussed in Section 2.1, and was thus not considered in this section.

2.5 Performance analysis and comparison to other methods

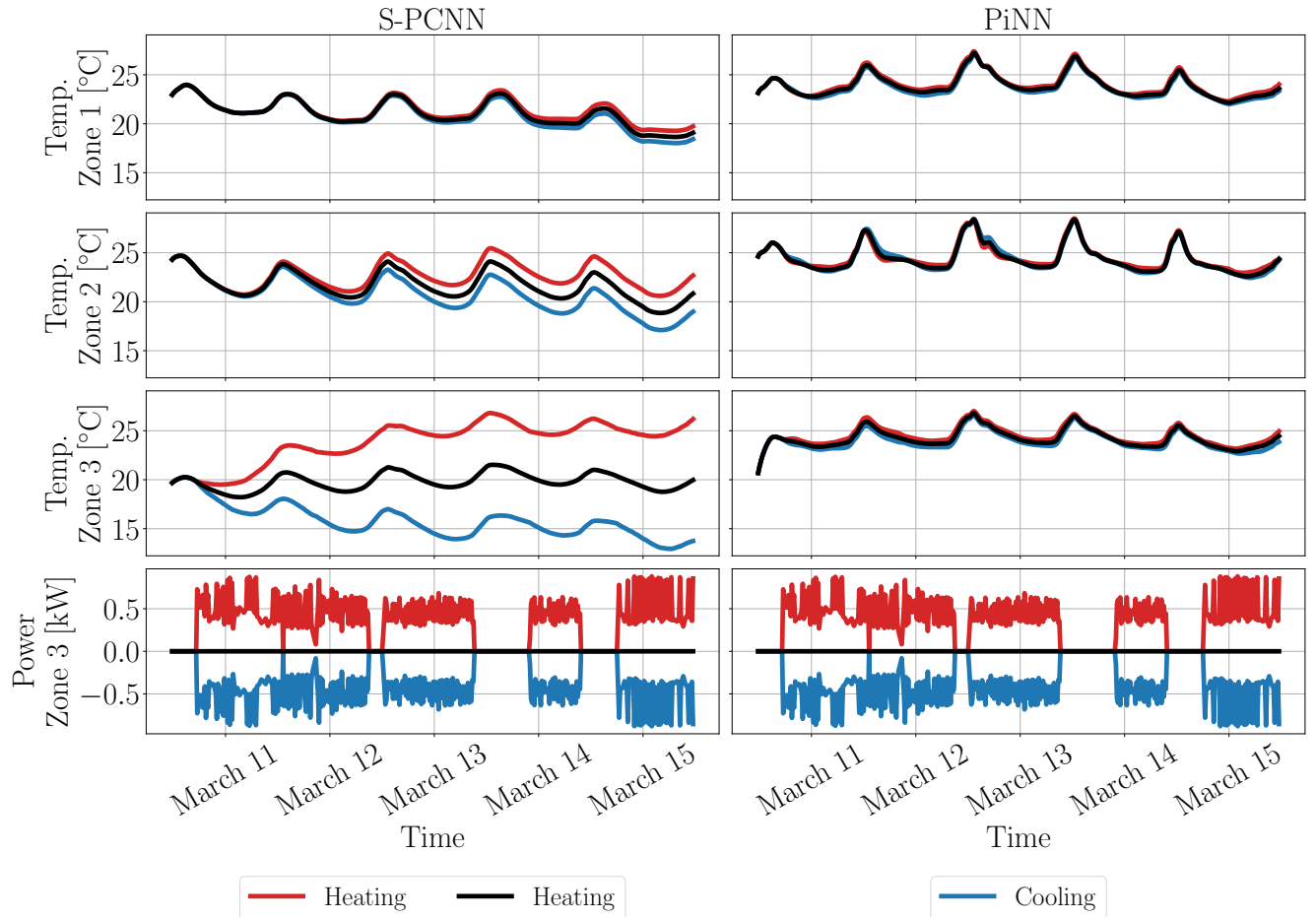


Figure 2.15: Visualization of heat propagation for an S-PCNN and a PiNN. The bottom plots show the heating (red) and cooling (blue) power inputs applied to Zone 3 while heating and cooling are turned off in Zone 1 and 2, compared to the situation when no power is applied (black). The other plots depict the corresponding temperature predictions of each model in the three zones.

Low errors are not always correlated with good models

Very importantly, our results point out a somewhat counter-intuitive and often overlooked characteristic of NNs: contrary to physics-based models, **a good fit to the data does not necessarily imply that the quality of the model is good**. In our case, the PiNNs and LSTMs were indeed able to fit the data well while completely discarding the impact of heating and cooling, i.e., solely mapping external conditions to building temperatures. When modeling physical systems, one should hence always make sure NNs do not simply find shortcuts to fit the data without respecting the underlying physical laws. This calls for physically grounded architectures, such as PCNNs, for applications where physical consistency is critical.

We suspect that LSTMs and PiNNs were able to achieve high accuracy without considering the impact of heating and cooling because of the specific data used in this case study. First, windows cover the entire East facade of UMAR, rendering the building especially sensitive to solar gains and external weather conditions. Second, while different controllers were applied during the data collection, all of them had the same objective of maintaining the building temperature in a comfortable range and hence reacted similarly to external conditions. Coupling these facts, it seems indeed plausible to accurately predict building temperatures solely based on external conditions and without considering heating and cooling inputs. In other words, we suspect the very expressive LSTMs and PiNNs to have learned the *closed-loop* response of the system instead of the expected *open-loop* one, hence implicitly accounting for the influence of power inputs instead of explicitly modeling their effect. This might explain how they found non-physical shortcuts modeling the evolution of inside temperatures well. Interestingly, the linear model also failed to capture any significant impact of heating and cooling (see Appendix A.7), showing that it is also possible to fit this data well while almost discarding these inputs without jeopardizing physical consistency.

Numerical analysis of physical consistency

To strengthen the theoretical and visual claims in Table 2.4 and Figure 2.15 about the physical consistency of NN-based models, we can carry out a numerical investigation of the gradients of their temperature predictions. Since gradients can be retrieved automatically through the `torch.autograd` module [140], it allows us to empirically assess if criteria (2.1) and (2.2) are respected, a necessary condition to ensure physical consistency. Following Remark 11, we investigate the gradients of the temperature predictions at the end of the three-day-long horizon with respect to the power inputs and ambient temperatures observed at each time step. This corresponds to the gradients used to steer the learning of PiNNs in (2.37), except for the X-PCNN, for which fewer gradients can be automatically recovered, as detailed in Appendix A.8. Note that we are only interested in the sign of each gradient, which should be positive according to (2.1) and (2.2) — their magnitude cannot be compared since they do not have any physical meaning.¹⁰

¹⁰This stems from the fact that NN-based models are fitted to normalized, hence dimensionless, data.

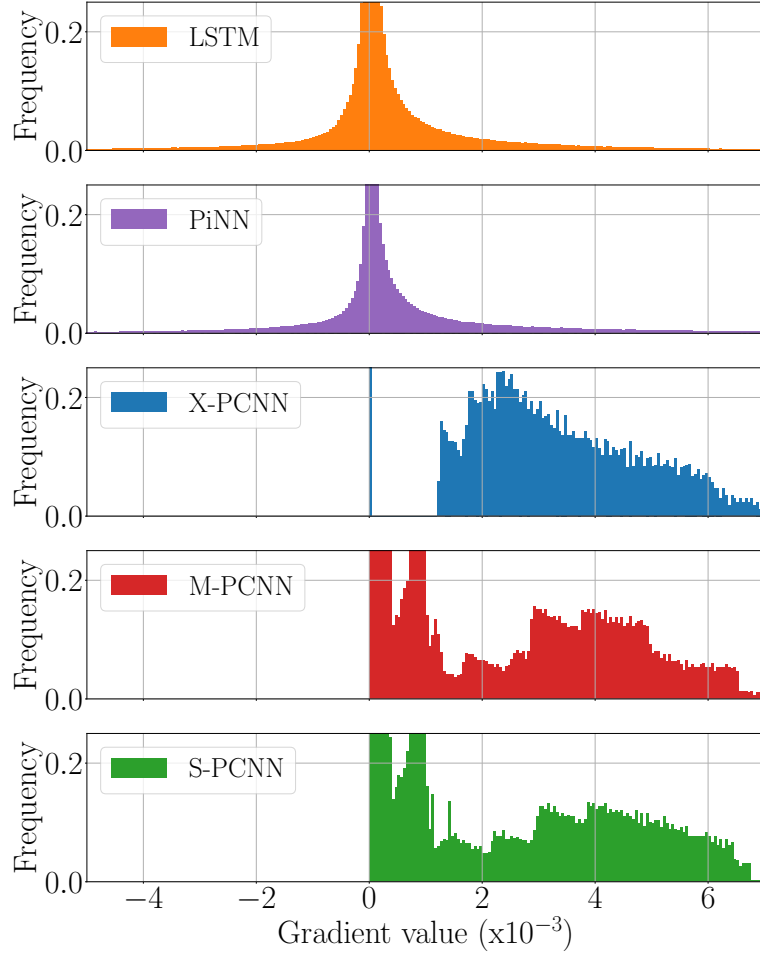


Figure 2.16: Distribution of the gradients of the temperatures at the end of the prediction horizon with respect to power inputs and external temperatures observed along the horizon for the NN-based models.

Over the entire validation data set, we have access to more than two million gradient values for each model, except the X-PCNN, with slightly over one million values, as computed in Appendix A.9. The resulting density histograms are shown in Figure 2.16, where one can directly observe negative gradients only for the two black-box models not grounded in the underlying physics. In fact, penalizing negative gradients in \mathcal{L}_{PiNN} decreased the magnitude of the PiNN gradients, steering them to zero, but did not significantly change the proportion of negative ones. In other words, it did not improve the physical consistency of PiNNs since they still violate conditions (2.1) and (2.2) almost as often as classical LSTMs.

Interestingly, the small magnitude of the PiNN and LSTM gradients corroborate what can be seen in Figure 2.15, with very little impact of heating and cooling for these models. On the other hand, thanks to their physics-inspired module, the proposed PCNN architectures

keep all the gradients that require positivity in \mathbb{R}_+ and with larger magnitudes, as desired and observed in Figure 2.15 for the S-PCNN, providing a numerical argument supporting their physical consistency.

Computational complexity to train the different models

As final comparison metric between the models analyzed throughout this case study, Figure 2.17 presents the time required by each model per training iteration. Importantly, these numbers are subject to implementation considerations and have to be taken with a grain of salt. Nonetheless, all of the NN-based models used the same backbone architecture, which allows relative comparisons, for example between the three proposed PCNNs, between the two residuals models, or between LSTMs and PiNNs. Note that the linear and ARX models are not considered here since their “training” procedure is very different: it neither requires access to GPUs nor relies on gradient descent.

First, as expected, PiNNs take more time to run than classical LSTMs since each batch has to be forwarded and backwarded through the networks twice, once to compute the predictions used in \mathcal{L}_{data} and another time to calculate the gradients in \mathcal{L}_{phys} . Second, residual models need access to the predictions of the underlying linear model at each step to compute the residual errors before fitting them, which also entails a clear computational overhead compared to LSTMs. Finally, PCNNs need to compute both the black-box module output \mathbf{D}_k and the physics-inspired module predictions \mathbf{E}_k at each step k along the horizon, which also entails additional overhead on top of classical black-box models. Interestingly, this is comparable to what happens in residual models, explaining to some extent why the latter and S-PCNNs require similar amounts of resources.

Compared to S-PCNNs, M-PCNNs and X-PCNNs are significantly more computationally intensive. This intuitively follows from the shared black-box module of S-PCNNs reducing the number of parameters to fit compared to M-PCNNs. On the other hand, X-PCNNs require learning several models separately — one for each zone — instead of everything together, which multiplies the computational overhead needed to create and move data to the GPU at each iteration and leads to an increased computational burden compared to M-PCNNs. Stemming from these remarks, we would expect these differences between the various PCNN architectures to grow if we were to apply them to larger buildings with more thermal zones.

Remark 14 (Parallelizing X-PCNNs). *The training times reported here correspond to the total time required to train each model for one iteration, i.e., the sum of the training times of each single-zone PCNNs in the case of X-PCNNs, to represent the total amount of computations needed. In practice, however, the different single-zone PCNNs could easily be trained in parallel since they are independent, which can significantly decrease the effective training time of X-PCNNs (dividing it approximately by three in our setting with three zones). This would make them the fastest multi-zone PCNNs to deploy but at the cost of additional computational complexity.*

2.5 Performance analysis and comparison to other methods

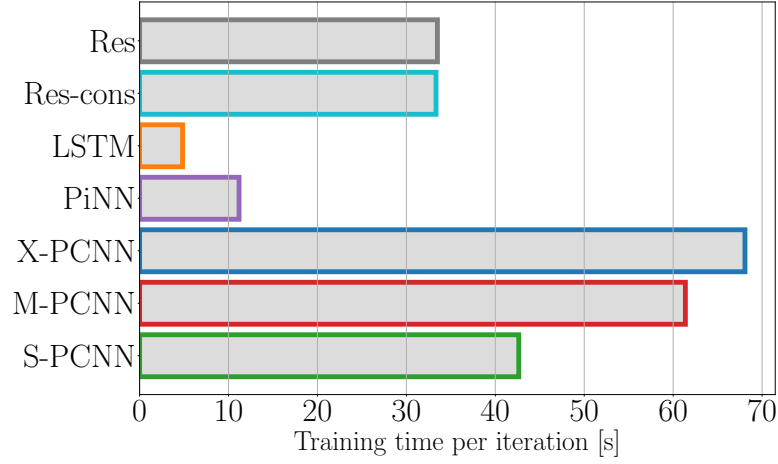


Figure 2.17: Average training time per iteration of the methods relying on a GPU.

2.5.3 Summary

The proposed multi-zone PCNNs respect the underlying physics by design and at all times despite requiring little engineering, contrary to classical physically consistent methods. On the other hand, they outperformed state-of-the-art black-box methods in terms of accuracy on a case study, hinting that the constrained structure introduced to ensure they follow some ground rules does not hinder their expressiveness. Our analyses showed little difference between S-, M-, and X-PCNNs in general, with S-PCNNs entailing the least computational complexity and X-PCNNs attaining the best accuracy on the analyzed case study.

Remarkably, PCNNs showed significantly better performance than classical physically consistent data-driven methods, with accuracy improvements of 30 – 35% and 17 – 22% compared to a linear and a residual model, respectively. While these results were obtained on a specific building, the significant performance gaps suggest that this trend would be observed for other applications, making PCNNs state-of-the-art building thermal models.

Our investigations also illustrated a well-known pitfall of classical PiNNs and LSTMs, which can find shortcuts to fit the data well without respecting the underlying physical laws. This exemplifies the need to not solely consider the fit to the data as a measure of the quality of NNs but also ensure that their predictions make sense from a physical point of view. Our findings thus support the current trend to incorporate inductive biases, i.e., prior knowledge, in NNs to alleviate their infamous generalization issues, leading to more principled architectures — like the proposed PCNNs.

2.6 Conclusion and outlook

In this chapter, we proposed a novel physically consistent NN architecture, providing fully data-driven control-oriented multi-zone building thermal models with little engineering overhead. The main idea of PCNNs is to let a physics-inspired and a black-box module run in parallel, the former guaranteeing the compliance of the output with the underlying physical laws — the laws of thermodynamics in the case of building temperature modeling — and the latter capturing unknown nonlinear dynamics, typically relying on NNs. We formally proved their physical consistency and extensively benchmarked them against other state-of-the-art data-driven methods.

To conclude the chapter, this section proposes an overview of opportunities, challenges, and potentially interesting future investigations regarding PCNNs.

2.6.1 The potential of Physically Consistent Neural Networks

PCNNs provide an alternative to engineering-heavy physics-based approaches, leveraging NNs to capture nonlinear and other hard-to-model behaviors without jeopardizing the desired physical consistency. Furthermore, they are highly flexible since the same architecture can be used for different buildings, again reducing the engineering overhead compared to white-box methods.

Extending the current architecture. While the physics-inspired module used throughout this work ensures physically consistent temperature predictions with respect to power inputs and ambient temperatures, this could be modified or extended. If other physical principles shall be respected, one can easily modify the form of the energy accumulator to account for the new desired properties. For example, ensuring a consistent impact of solar gains might be critical for some applications. Similarly, while we only considered trainable constant parameters in the physics-inspired module herein, it could be extended to capture more complex phenomena, like time-varying parameters or parameters that depend on other factors. Note that the physics-inspired module does not have to be linear and might incorporate nonlinearities for more expressiveness without jeopardizing physical consistency, similar to what was proposed in [139]. Thanks to the BPTT training procedure of PCNNs, the parameters of the physics-inspired module can be seamlessly learned from data¹¹ in parallel with the parameters of the black-box one.

Furthermore, while the only nonlinear gains considered in this chapter come from solar irradiation, the flexibility of the black-box module could also allow it to capture other unknown or hard-to-model heat gains, such as the ones stemming from occupants. If data on these new heat gains is available, one could indeed easily feed it to the black-box module together with solar irradiation measurements and time information to let the NNs learn their impact on zone temperatures.

¹¹Note that the same training procedure will be used for SIMBa in Chapter 4.

Going beyond temperature predictions. Given the importance and complexity of the comfort of the occupants in buildings, many different parameters need to be controlled to ensure adequate Indoor Air Quality (IAQ), and not only the temperature [144]. Since more and more factors need to be accounted for, it will further increase the engineering burden associated with traditional physics-based methods to model all the required variables. On the other hand, assuming data to be available, PCNNs can easily be extended to model more complex IAQ criteria. Such an extension was proposed in [139], where the physics-inspired module was augmented to accommodate humidity predictions on top of temperature ones.

Accelerating white-box models. Due to their performance, PCNNs would also be potential candidates for accelerating ODE-based models, which typically incur a considerable computational burden at run-time, similarly to what was proposed in [112, 119]. Indeed, one could first use such a high-performance model to generate a large data set, train a PCNN to approximate the corresponding dynamics, and then leverage the latter for faster inference. This would shift the computational load from inference to training, which is preferable for some applications, at the cost of a yet-to-be-quantified performance loss. Compared to classical pipelines relying on vanilla NNs, however, PCNNs would guarantee compliance with the underlying physical laws at run-time.

Tackling other applications. Finally, while we only discuss how to apply PCNNs to building thermal modeling in this work, they could be leveraged to model many other dynamical systems, provided historical data is available. Indeed, engineers usually have at least an approximate understanding of the physics driving any system of interest — and expect models to follow these principles. This prior understanding can be formulated in the physics-inspired module, similar to what we did in this work for buildings, before letting an NN simultaneously adapt to the data to grasp any unmodeled or not well-understood phenomena. Interestingly, this could alleviate the computational burden of calibrating accurate physics-based models: a simplified physics-inspired module might be enough to fit the data well when an NN captures the residual physics in parallel.

2.6.2 Limitations of Physically Consistent Neural Networks

If PCNNs achieved consistent and state-of-the-art performance throughout our investigations, we cannot draw any definitive conclusions since we only benchmarked them on a single case study. Indeed, PCNNs might fail to model other buildings with different thermal dynamics accurately, and all the results presented herein hence have to be taken with a grain of salt. On the other hand, however, UMAR is not an easy case study, with temperatures routinely rising above 26 °C even in winter. Due to its east-facing glass facade, solar gains indeed play a crucial role, and their nonlinear nature is challenging to capture. Our results are thus a good indication of PCNNs' ability to model nontrivial effects.

Apart from the specificity of the case study, another potential source of quantitative errors in the results comes from the many hyperparameters of NNs — and hence PCNNs. They

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

were not tuned to optimality in all the cases but rather empirically set to achieve consistent performance. While we would not expect drastic changes for better hyperparameters, and none that would modify our conclusions, it could still have a quantifiable impact. However, as long as PCNNs, PiNNs, and LSTMs use the same hyperparameters, we expect the results presented in this chapter to hold, with PCNNs achieving the best performance overall.

Beyond these considerations specific to our numerical experiments, the widespread deployment of PCNNs for building thermal modeling still faces several challenges.

Building topology. First, while only requiring access to the topology of the building significantly reduces the engineering burden compared to traditional physics-based methods, it might not be trivial in practice. For example, the layout of the zones is usually easy to recover from design plans, but the link with the data is often missing. In other words, knowing which data point corresponds to which thermostat and in which zone that thermostat is located can be complicated. While *automatic topology discovery* — recovering the topology of the building from data — could solve that issue, it remains an open problem in the building sector. Alternatively, one could borrow ideas from the automatic gray-box identification in [145] and greedily add nodes to the topological graph to discover the topology, for example. However, this implies training several PCNNs at each step and would hence incur a significant computational burden that would not be feasible in practice, at least for large-scale buildings.

Behavior of the physics-inspired module. The initialization and convergence of the physics-inspired module parameters are additional significant limitations necessitating further care before any full-scale deployment of PCNNs. Since these physics-inspired parameters are learned in parallel with very expressive NNs, their initial value plays a key role. Indeed, the NNs might otherwise try to capture all the trends in the data and discard the physics-inspired module — i.e., setting its parameters to values close to zero —, for example, similarly to what could be observed for LSTMs and PiNNs in our experiments in Section 2.5.2. Although this would never violate the physical consistency of PCNNs, it could lead to meaningless solutions if the physics-inspired parameters take unrealistic values.

While the hand-crafted initialization rules described in Section 2.4.3 and used throughout this chapter perform well for such a small-scale case study, an automated framework detecting plausible values from data would be required to tackle large-scale buildings with several hundreds of zones. Worryingly, we observed that the quality of the solution can vary significantly if these parameters are initialized to unrealistic values. In other words, PCNNs do not always recover physically consistent parameters from data without meaningful initial values.

To make matters worse, even after being initialized to plausible values, these physics-inspired parameters only represent a tiny fraction of the total number of parameters while significantly influencing PCNN predictions. The impact of backpropagation on the parameters of the physics-inspired module requires further analysis, especially since we tune the hundreds of parameters of the black-box module in parallel. On the one hand, we do not want the physics-inspired parameters to change too much from the initial guess since they need to be

consistent with the physical world; on the other, if they move too little, they will not adapt to the data to improve accuracy. The optimal trade-off between these two paradigms to achieve robust learning and performance remains an open question.

Data availability. Finally and critically, before considering the deployment of PCNNs, be it for building modeling or any other application, the appropriate data has to be available. This entails having access to all the necessary data points in sufficient quantity and high quality. As discussed, NNs are indeed notoriously data-hungry and suffer from poor generalization. In other words, they will try to fit undesired trends if some are present in the data.

In the case of building modeling, for example, we assumed throughout our work to have access to individual zone power inputs, which are usually unavailable. While it can be possible to disaggregate the total power measurement into zonal ones based on system design information, as was done for UMAR, it is not always possible to access the required data in practice. This disaggregation function, instead of being engineered, could be learned from data simultaneously to the other parameters of the physics-inspired and black-box modules, as mentioned in Remark 4, but the efficacy of such a scheme to capture the true individual powers remains an open question. Note, however, that an approximate solution might be sufficient in practice: as long as the right total amount of thermal power is provided to the building, it will then be distributed among the zones through heat transfers, following the laws of physics encoded in PCNNs.

2.6.3 Outlook

Beyond training PCNNs on other data sets and buildings to assess their robustness, some of the aforementioned limitations must be addressed before widespread adoption and large-scale experiments. In particular, understanding how to automatically initialize the physics-inspired module parameters and the impact the training procedure has on them requires further analyses. Setting different learning rates for both modules and investigating what happens to the gradient of these sensible parameters could be helpful in that regard.

Improving initialization. Instead of the data-based solution proposed in this work, one could alternatively use SI techniques to first fit the physics-inspired module to the data. Setting $\mathbf{D} \equiv 0$ and $\mathbf{E}_0 = \mathbf{T}(0)$, this procedure would find a solution where the physical parameters approximately fit the data. One could then initialize all the NN weights to zero — instead of random values — so that the PCNN learns to improve upon the performance of the physics-inspired module, similarly to what was discussed in [146].

Note that this resembles residual models, but we would expect the physics-inspired parameters to be updated in parallel with the NNs in the second step to adjust to the fact that the black-box module now captures some nonlinearities in the data. In other words, in this scheme, both modules would still be learned together, as proposed in this work and contrary to residual models. However, the physics-inspired module would be initialized through SI techniques,

Chapter 2. Physically Consistent Neural Networks for thermal building modeling

and the black-box one would start at zero, replacing the hand-crafted-rule-based and random initialization of both modules, respectively, used throughout this work. Interestingly, such an approach could be seamlessly integrated with the SIMBa toolbox for SI proposed in Chapter 4, which also relies on BPTT, as the PCNNs proposed in this chapter.

Improving learning. In a similar vein, instead of fitting PCNNs from scratch for each new modeling task, one could investigate how to transfer (parts of) an existing PCNN trained with another data set or on a different building. This is a popular research topic, with several open questions, such as: What amount of data is needed for a successful transfer? What part of PCNNs can be transferred? How can we handle different building topologies? These questions recently gave rise to Transfer Learning (TL) [147], a field gaining in popularity in building applications [148] that might provide some answers. However, PCNNs have their peculiarities rendering them more challenging to transfer than vanilly NNs. In particular, the physics-inspired module needs to comply with the desired physical laws on the new data set.

Alternatively, training a PCNN with a large number of parameters to capture the dynamics of a wide variety of buildings, only requiring little fine-tuning towards good performance on any new building, could be a promising approach, similar to what was done in [114].

To ease the training of PCNNs and alleviate some of the associated computational burden further, one could leverage *curriculum learning*, where the prediction horizon is gradually increased during training, starting from one-step-ahead predictions [149]. This would allow PCNNs to start by solving easier problems and converge to meaningful solutions before gradually increasing the difficulty. Additionally, to simplify the learning task, the temperature measurements data set — or the time series of any quantity of interest — could first be decomposed into sub-components with less complex dynamics, for example, leveraging the Fourier or wavelet decompositions. This improved the final modeling performance in [150], for example.

Capturing more complex behaviors. These training improvements would incidentally help to apply PCNNs to larger-scale case studies and increase in importance if more complex indoor air quality models — not only considering the inside temperature — are desired. As we move towards more occupant-centric building control frameworks, ever-more complex building models will indeed be required. It would hence be interesting to investigate how to extend the simple physics-inspired module used throughout this work to capture more complex IAQ phenomena, in line with the humidity predictions introduced in [139].

Analyzing data requirements. In general, it would be intriguing to analyze how different hyperparameters and — more importantly — the quantity and quality of the data influence PCNNs. While we leveraged three years of data throughout this work, which covers all seasons and various operating conditions, such a data set is rarely available in practice. Consequently, it is important to understand what data PCNNs require to achieve given performance criteria. For example, it might be possible to learn a good model for each season separately, using (much) less data in each case than required to train a PCNN performing well all year round.

Related to these investigations, it would be interesting to assess the potential of *continuous learning*, where recently collected data points could be used to retrain PCNNs periodically after a shorter offline training phase. This could allow PCNNs to adapt online to the slowly changing environmental conditions, among others. If the PCNN is only expected to be accurate for a short period of time, it might indeed be possible to maintain a well-performing model from limited amounts of recent data, bypassing the expensive offline training phase covering all possible operational conditions.

On the other end of the data spectrum, how to deal with large data sets — for example, consisting of hundreds or thousands of buildings — at a reasonable computational cost remains an open question. Leveraging techniques from federated learning and doing spatial and/or temporal downsampling to cluster similar buildings together, similar to what was done in [114], could speed up the learning in these cases.

Impact of other research fields. Apart from improvements to PCNNs, several other domains could help the widespread deployment of PCNNs. As mentioned in the limitations above, these include advances in automatic topology discovery and automatic power disaggregation algorithms. Going one step further, it could be interesting to intervene *before* the data collection starts, introducing systematic procedures to link data points and the physical building when the system is set up, following the concept of *linked data*, for example [151]. Depending on the implementation of such a scheme, the building topology could also be stored with the data so that PCNNs could have direct access to all the required information and be seamlessly trained without any engineering effort.

Applications. Whenever PCNNs achieve state-of-the-art accuracy, as in our case for UMAR in this work, they could be subsequently deployed in many different applications, especially around control. First, since they are physically consistent, they can be leveraged as trustworthy simulation environments for DRL agents, as proposed in Chapter 3. Maybe more importantly for practical applications nowadays — given the early stage of DRL research —, it could be interesting to use PCNNs as thermal models in MPC controllers. In particular, since the PCNN architecture analyzed in this chapter is power input-affine, it would result in convex optimization procedures inside the MPC for an appropriate choice of cost function [125]. Consequently, it would provide a low-complexity MPC formulation — despite relying on NNs in the black-box module — while bypassing the need for engineering-heavy models.

Going beyond buildings. Finally, to investigate the general potential of the proposed architecture with a physics-inspired module being complemented by a black-box one, other applications beyond building thermal modeling would be required. In practice, engineers often have access to simplified physical models of the system and these could be leveraged as physics-inspired modules. With highly expressive NNs simultaneously grasping unmodeled effects from data, we would expect PCNNs to generally achieve high accuracy, but this theory remains to be tested. It showed promising results for gas piston systems, for example (see Chapter 4), but more case studies are required before drawing any definitive conclusion.

2.6.4 Concluding remarks

This chapter introduced a novel NN architecture, dubbed PCNN, introducing expert knowledge in a physics-inspired module to ground the predictions in the underlying physics. Despite being limited to a single case study, empirical analyses showed PCNNs can achieve impressive performance, surpassing other data-driven methods. They allow us to use NNs while avoiding some of their pitfalls, mainly linked to their lack of generalization. Indeed, PCNNs achieved better performance on the validation data than classical LSTMs despite being outperformed on the training one. One should keep in mind, however, that this issue is not fully solved: there is still an NN in the pipeline that can behave undesirably in some situations. PCNNs are indeed only physically consistent with respect to the inputs treated accordingly in the physics-inspired module.

Nonetheless, overall, **we hope that PCNNs can pave the way for potentially large-scale NN-based models able to simultaneously provide state-of-the-art performance *and* physical guarantees for building thermal modeling and beyond.**

3 Prospects and hurdles of Deep Reinforcement Learning for building control

Following the discussion on the importance of advanced building control methods in Section 1.1, this chapter investigates the potential of DRL policies in that context. We start by defining the characteristics of an *ideal* building controller and discussing the advantages and disadvantages of existing control methods. Along this analysis, we argue in favor of computationally lightweight, constrained, and efficient model-free DRL solutions. We then report two case studies supporting the feasibility of such controllers in Sections 3.3–3.4, paving the way towards model-free DRL agents that could be deployed from scratch in buildings and automatically learn to optimize operations at large-scale.

3.1 The potential of model-free Deep Reinforcement Learning

Buildings are characterized by slow thermal dynamics, i.e., it takes several minutes for heating to have a noticeable impact on the temperature, for example. Practitioners thus often choose a time step of 10–15 min for high-level controllers focused on energy minimization [28]. This implies that only a few control decisions and data samples can be taken daily, rendering real-world experiments challenging [14, 144]. Furthermore, occupants play a major role in terms of disturbances — they provide additional heat gains and can open doors and windows, for example — and comfort preferences [144, 152]. This adds to the fact that every building presents distinct thermal dynamics and Heating, Ventilation, and Air Conditioning (HVAC) facilities and reacts differently to various *disturbances*. These disturbances — mainly stemming from weather conditions and the behavior of the occupants — can also vary widely from one building to another depending on its location or the occupants' habits, among others. **Overall, we hence require adaptive control solutions able to adjust to each building** [28].

Throughout this chapter, we differentiate between *offline* methods, which use a fixed historical data set to train a model or a control policy before deploying it on the system, and *online* approaches, which learn a model or a control policy while controlling the system and collecting data. Note that both ideas can be combined, and we may then refer to the offline and online phases as the *pre-training* and *fine-tuning* procedures, respectively.

3.1.1 Requirements of an ideal building controller

To ensure its widespread adoption, we **argue that an *ideal* building controller should meet the following requirements**, inspired and adapted from the points raised in [15, 28, 153, 154]:

- R1. **Optimality:** First, an ideal controller should naturally achieve optimal system-wide performance, i.e., use the least possible energy while satisfying the comfort of the occupants.
- R2. **Robustness to disturbances:** Building thermal dynamics are heavily impacted by disturbances, and an ideal controller should be robust and react accordingly to perform optimally in any condition — during a sunny summer day or a snowy winter one and whether occupants are present or not, for example.
- R3. **Constraint satisfaction:** Although maintaining a comfortable indoor environment is a soft constraint in our setting — violations will not break the system and might not even affect the occupants —, any controller deployed in a real-world application needs to respect the preferences of the occupants to avoid complaints to achieve widespread adoption. This calls for control solutions that can handle (soft) constraints, i.e., limit the total time and amount of constraint violations.
- R4. **Lifetime adaptability:** An ideal controller should detect changes in building dynamics due to different occupant preferences, retrofitting operations, or aging of the components. It should continuously adapt its behavior and provide adequate responses. This is also known as continuous commissioning [155].
- R5. **Scalability:** Deploying an advanced thermal controller in a modern *smart* building might entail coupling it with solar Photovoltaic (PV) electric energy production, batteries, Battery Electric Vehicles (BEVs), and appliance scheduling, among others [41]. Furthermore, occupants might be sensitive not only to the indoor temperature but also to the relative humidity and air movements, for example, and react differently to these conditions depending on their personalized comfort preferences [144]. These issues naturally amplify with the size of the building to optimize, and an ideal control solution should thus be able to deal with large numbers of variables and control actions.
- R6. **Transferability:** Since every building is different, an ideal control method should be system-agnostic and out-of-the-box to undergo widespread adoption. This is required to alleviate the engineering effort associated with manual interventions during deployment and avoid prohibitively expensive solutions.¹
- R7. **Fast convergence** (online): While online methods naturally need time to adapt to a building when deployed, contrary to offline ones, an ideal controller should not take months or years before reaching satisfactory performance. It could otherwise consume significant amounts of energy and incur unacceptable discomfort for the occupants.

¹In this chapter, *transferability* refers to system-agnostic methods that can seamlessly be deployed in any building once they have been developed. This is slightly different from its classical definition in the TL literature, where transferring a policy generally refers to taking information from one building, transferring (part of) the corresponding data, model [147], or control policy [156] to another one, and then fine-tuning the controller on the new task. TL could benefit any data-driven controller, alleviating R6 and R7 to some extent. However, whether it can fully solve the transferability issue in practice — according to our definition — remains an open question.

3.1 The potential of model-free Deep Reinforcement Learning

Method	Requirement						
	R1 Optimality	R2 Robustness	R3 Constraint satisfaction	R4 Adaptability ^a	R5 Scalability	R6 Transfer- ability	R7 Fast convergence
RBC	--	-	o	--	--	-	/
BO-RBC	o	+	+	++	--	+	-
MPC	+	++	++	-	-- ^b	-- ^b	/
DPC	+	o	o	-	o	-	/
DeePC	o	+	++	+ ^c	--	-	o
Online MPC	+	+	+	+	- ^b	- ^b	+
Vanilla DRL	++ ^d	--	--	++	++	++	--
Safe DRL	o	-	++	+	o	++	--
Backup-DRL	o	-	+	+	++	++	--
Online MBDRL	+	-	--	++	- ^b	+ ^b	o
MPC-DRL	+	+	+	++	-- ^b	- ^b	o
IL-DRL	+	-	o	++	-	-	+
ReL-DRL	+	o	-	++	-	-	+
Offline DRL	++ ^d	+	+	++	- ^b	- ^b	/
Potential of CEO-DRL	++ ^{d,e}	+ ^f	++ ^f	++	++	++	+ ^f

Glossary: **RBC**: Rule-based control, **BO-RBC**: RBC tuned with Bayesian Optimization, **MPC**: Model Predictive Control with a fixed model learned before deployment, **DPC**: Differentiable Predictive Control, **DeePC**: Data-enabled Predictive Control, **Online MPC**: MPC with online model update, **Vanilla DRL**: Online model-free Deep Reinforcement Learning, **Safe DRL**: Constrained policy optimization methods, **Backup-DRL**: DRL with safety backup controllers, **Online MBDRL**: Model-based DRL with online model update, **MPC-DRL**: Methods merging MPC and DRL, **IL-DRL**: DRL with Imitation Learning (IL), i.e., the policy first learns to imitate an existing controller offline, **ReL-DRL**: DRL with Residual Learning (ReL), i.e., the DRL agent interacts with another controller for assistance, **Offline DRL**: DRL agents leveraging a building simulator to train offline before deployment, **Potential of CEO-DRL**: The potential of Constrained and Efficient Online model-free DRL algorithms discussed in this chapter.

Keys: "--": particularly challenging, "-": not met in general, "o": neutral or unknown,

"+": partially met (i.e., nontrivial or not always met), "++": met in general, "/": not applicable.

^aWhile any offline method can be extended to be updated online and satisfy R4, we consider them separately for clarity here.

^bThese challenges might be alleviated through the use of data-driven modeling techniques like PCNNs (Chapter 2) or SIMBa (Chapter 4).

^cWhile the vanilla version of DeePC does not adapt to new data online, recent indications hint that it can indeed be done [157].

^dSince they do not rely on the accuracy of a model online, these methods might outperform model-based approaches.

^eAnalyzed in Section 3.3.

^fAnalyzed in Section 3.4.

Table 3.1: Qualitative assessment of the potential of different building control methods to satisfy the seven identified requirements of an ideal controller, summarized from the discussion in Sections 3.1.2 and 3.1.3.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

In light of the aforementioned seven requirements of an ideal building control method, let us now discuss the advantages and disadvantages of existing approaches. A qualitative summary of the following two sections can be found in Table 3.1.

3.1.2 From classical to data-driven adaptive control methods

RBC: Rule-based controllers

While manually tuned RBCs still dominate current building industry practices, they usually fail to achieve system-wide optimal performance, even for simple problems, and hence do not meet R1 and R5 in general [15, 28]. Furthermore, an RBC will only satisfy R2–R3 if it has been expertly tuned to react to every disturbance and constraint, a challenging task even for a single control variable [158]. This also means that R4–R6 are generally not met by RBCs: adapting to changes online would indeed require retuning the parameters every time the dynamics or the occupants change, and scalability and transferability cannot be achieved with a controller that requires heavy manual tuning for each application [16].

BO-RBC: Automatic tuning of rule-based controllers

Since the main issues with RBCs stem from the tedious manual tuning procedure required to achieve good performance, automatic tuning algorithms, typically based on Bayesian Optimization (BO) [159], might lead to significant improvement. This allows RBCs to satisfy R4 and R6 since they can be optimized from scratch on any building and keep updating the parameters throughout their lifetime. BO-RBC combinations were already successfully demonstrated for building control in [160, 161]. Remarkably, vanilla BO can be extended to handle contextual inputs and constraints, hence naturally meeting R2–R3 [162–164]. However, R1 remains an open question in general, as the performance of RBCs is limited by the chosen rules.

Worse yet, R5 is not met by these autotuning methods since BO is infamously known for its computational complexity in high dimensions [165]. To mitigate this scalability issue, DRL could also be used for RBC tuning (as in [166], for example), but the efficacy of such a scheme has yet to be tested for building control and might fail to incorporate constraints (R3). Furthermore, it would most likely be hindered by the slow dynamics of buildings since DRL is infamous for its poor sample complexity [29]. Although this might be mitigated to some extent through entropy maximization to ensure sufficient exploration of the state-space, as proposed by [167], it still took several hundreds of episodes to converge on a straightforward proportional-integral-derivative controller tuning problem. If the building RBC parameters are only updated daily, like in [160, 161], for example, it might take months or years to find well-performing parameters, jeopardizing R7. Note that, despite being more sample efficient than DRL, tuning three parameters with BO — and only for the heating season — still took several months to converge in [161].

3.1 The potential of model-free Deep Reinforcement Learning

MPC: Model Predictive Control

MPC is arguably the most popular advanced building control method, but it has not yet been widely accepted by the industry, mainly because of the high associated engineering and installation costs [7, 28, 168, 169]. Since it relies on an explicit model of the system to predict future trajectories and then solves a constrained optimization problem at each step, MPC achieves near-optimality, is robust to disturbances, and satisfies constraints by design (R1–R3), provided detailed forecasts of the disturbances are available and the model is accurate. These are however strong assumptions in practice.

On the other hand, in its offline form, i.e., when the model is not adapted online, R4 is not satisfied [13]. Furthermore, the model of the building’s thermal dynamics — or any variable of interest — at the core of any MPC controller plays a crucial role; imprecise models can indeed lead to poor control performance [170, 171]. This shifts the burden back to finding accurate building models, a nontrivial task, as extensively discussed in Section 2.1.2. MPC thus satisfies neither scalability nor transferability in general (R5–R6), especially if it relies on cumbersome white-box models.

Leveraging data-driven models. To alleviate the burden of engineering models, one can instead use gray- or black-box models in MPC, such as linear [172], random forest [17], Gaussian process [173], or NN [174] models, for example. Note that there is usually a trade-off here between model complexity — often positively correlated to the final performance of MPC in building applications [175] — and the associated computational complexity of the optimization problem at inference time. This influences the ability of MPC controllers to simultaneously meet R1 and R5: more complex models will be required to solve large-scale problems with satisfactory performance (R1), leading to more complicated optimization procedures and thus hindering R5.

Interestingly, the PCNNs proposed in Chapter 2 could be ideal candidates to solve this trade-off since they can be input-affine (Remark 9) — leading to convex optimization procedures inside the MPC for adequate choices of cost function [125] — while reaching state-of-the-art accuracy without engineering overhead. Consequently, a PCNN-MPC framework could be expected to simultaneously achieve near-optimality and scalability (R1 and R5), provided accurate disturbance forecasts are available.

As a predictive control method, MPC indeed relies on disturbance forecasts to optimize the control inputs, and it is known to be sensitive to their precision [171, 176]. This limits its ability to scale to more complex problems in general (R5) — even if a well-performing model of the dynamics were accessible for any building — since each disturbance would need an accurate forecast model (the PV production or the BEV schedule, for example). In fact, solely grasping the impact of occupants is a complex task with a research field of its own [177–179]. Coupled with the fact that possibly extensive data sets would be required for each building to develop such data-driven MPCs, transferability is still out of reach of standard MPC controllers (R6).

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

Remark 15. *Combining ideas from PCNNs in Chapter 2 and SIMBa in Chapter 4 may enable the creation of a pipeline capable of simultaneously identifying the behavior of all the variables and disturbances from historical data. This could be a significant step towards MPC controllers satisfying all the requirements except R4, but significant effort is still required to create building-agnostic identification methods capable of capturing all the complexities of the building control problem to achieve scalability and transferability (R5–R6). Furthermore, such a controller could only be deployed after an initial data collection phase.*

DPC: Differentiable Predictive Control

As another means to reconcile performance and scalability (R1 and R5) for MPCs, Drgoňa et al. recently introduced the Differentiable Predictive Control (DPC) framework [180]. DPC only requires an offline data set to learn a neural state-space model and a control policy able to handle state and input constraints. In other words, it acts as a proxy for an MPC, bypassing the optimization procedure, similar in spirit to approximate MPC frameworks like [181, 182]. DPC was applied to building control in [183] and presents the potential to meet both R1 and R5, as PCNN-MPC.

On the other hand, it has the same drawbacks as other data-driven MPCs concerning data and forecast requirements, which hinder scalability and transferability (R5–R6). Since it relies on NNs, DPC might also suffer from their generalization issue (Section 2.1.1) and perform poorly for unexpected disturbances (R2). Additionally, it might fail to respect constraints (R3) since the latter are only incorporated through penalties in the loss function and not hard-coded like in MPC [183].

DeePC: Behavioral approaches

To avoid pitfalls associated with finding a suitable model for MPC, researchers have recently leveraged Willem’s *fundamental lemma* [184] for direct controller design. For Linear Time-Invariant (LTI) systems, this lemma gives sufficient conditions under which *any* trajectory of a system can be represented as a linear combination of the input/output Hankel matrix’ columns. In other words, all future trajectories of an LTI system can be predicted from a past one, provided the latter is *sufficiently excited*. This recently led to the rise of Data-enabled Predictive Control (DeePC) approaches, where past data is used instead of a model to predict future trajectories [185]. It has been applied to building control in [186, 187], for example.

As a predictive method, DeePC naturally satisfies constraints R3 as long as the building dynamics are linear and the historical data set is sufficiently excited. However, the vanilla DeePC version assumes no measurement noise and is not robust to disturbances in practices [188]. To satisfy R2, one can instead leverage one of the several schemes introduced to robustify DeePC against noise, such as [188, 189]. On top of these improvements, one can consider online time-varying Hankel matrices to adapt to the changing building dynamics online and

3.1 The potential of model-free Deep Reinforcement Learning

satisfy R4. However, this raises questions about which data to retain and discard at each step to attain the best performance in general [190]. A very promising direction to solve this dilemma was recently provided in [157], where an efficient update algorithm was proposed. It allows for fast online computations even with a growing amount of data by transforming the original DeePC problem into a lower-dimensional one of fixed complexity.

On the other hand, for the lemma to hold, one needs to ensure the persistency of excitation of the inputs, which requires an additional mechanism for real-world experiments, as discussed in [188], complicating the deployment of DeePC. To make matters worse, the data requirements for DeePC to perform adequately likely differ from one building to another, with two to thirty days being selected in [188, 190] depending on the setting, for example. Note that less data might be required if the Hankel matrix is periodically updated online to capture new conditions. Combined with the notorious difficulty of tuning DeePC controllers [188, 191], this approach does not scale or transfer well yet (R5–R6).

Finally and more critically, DeePC relies on the linearity of the underlying system to leverage Willem’s fundamental lemma. While the temperature dynamics of a thermal zone are approximately linear, this characteristic will not persist for more complex case studies (R5), and we cannot expect DeePC to achieve global near-optimal performance in general (R1). Finally, as MPC and DPC, DeePC still relies on forecasts of all the disturbances, complicating its deployment in large-scale applications and transferability (R5–R6), as discussed above.

Online MPC: Model Predictive Control with online model update

To alleviate the issues stemming from finding accurate building models — or a substantial amount of high-quality data for data-driven techniques —, one can learn or refine the model online while controlling the system [192]. Traditional approaches usually start from a model learned offline and periodically refit its parameters given the newly collected data to improve the performance of MPC controllers [33, 193–196].

In a similar vein, any of the data-driven MPC or DPC approaches discussed above could be extended to satisfy R4 by periodically updating their corresponding model online. Each method would keep its advantages (R1–R3) and inconveniences (R5–R6), mainly stemming from the need for accurate disturbance forecasts if the models can be fully learned online, like in [197]. However, this is the exception rather than the rule since most approaches first initialize the models offline [193–196] and hence suffer from the same pitfalls as classical methods during that phase. On the other hand, for data-driven modeling techniques, less data might be required than in the purely offline case since the model is then fine-tuned online to improve its accuracy under new circumstances [33].

Even if a model were to be fully identified online, similarly to the discussion on adaptive DeePC, the optimal updating procedure and frequency remain open questions, with models being updated weekly using the past month of data in [195] or being updated daily using the

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

entire data set in [196], for example. Additionally, it is likely building- and problem-dependent, further complicating scalability and transferability (R5–R6). Finally, since the model is updated online in that case, the performance could suffer during the learning phase (R7). Nonetheless, online MPC seems to quickly provide acceptable results in practice, mainly thanks to its ability to incorporate constraints. This allows it to usually not behave catastrophically online, especially after some offline model pre-training, as exemplified by the results in [193–197].

A promising pipeline. One promising approach was proposed in [33], where an online NN-MPC was coupled with an outlier detector and a fallback controller. The main idea of the former is to detect whether the NN model can be trusted before applying the MPC control inputs to the building and fall back to the safe controller otherwise. Remarkably, this avoids the pitfalls associated with the generalization issue of NNs (Section 2.1.1). Since the NN is periodically re-trained, it safely learns to accurately capture new operating points, in turn decreasing the need for the backup controller throughout the deployment time.

This scheme provides a generic solution applicable to many buildings and enjoys promising scalability and transferability properties, alleviating R5–R6 to some extent. However, disturbance forecasts are still required, and using NNs as models might lead to complex optimization procedures for the MPC at inference time, complicating large-scale applications (R5).² Furthermore, more complex problems would require NNs with more parameters to be accurately modeled. This, in turn, would necessitate larger data sets to initially fit the model offline and achieve satisfactory performance online in a reasonable amount of time, i.e., avoid falling back to the backup controller too often during the learning phase (R7). Nonetheless, we suspect that leveraging PCNNs (Chapter 2) in such a pipeline might lead to controllers satisfying R1–R6.

3.1.3 Deep Reinforcement Learning

Instead of anticipating the impact of various actions in order to choose the best one, as in the predictive methods discussed above, RL agents traditionally directly interact with the system and do not require access to a model. At each time step, they choose an action depending on the observed state of the environment,³ get rewarded for it, and the environment moves to the next state. All RL *algorithms* aim at maximizing long-term rewards via trial and error [19].

DRL — RL with NN-based control policies — rose to prominence after achieving groundbreaking and even superhuman performance on Atari games in 2013 [198]. However, its main successes are often limited to simulated environments or systems that can be accurately modeled (for example, [199–201]). State-of-the-art real-world applications of DRL are indeed usually only possible after first training in simulation in fields like robotics [202], plasma control [64], or drone racing [203], among others. This stems from the challenges arising from real-world DRL applications, including the fact that agents should be able to learn from limited

²This can be alleviated in practice by using input-affine NNs, such as the PCNNs proposed in Chapter 2.

³The system is often referred to as the *environment* in the RL literature; both words will be used interchangeably throughout this chapter.

3.1 The potential of model-free Deep Reinforcement Learning

samples and respect system constraints at all times [204]. Indeed, robots or drones can only fail in simulation to avoid breaking the physical system, for example — and failures are generally required to teach DRL agents how to avoid them due to their trial-and-error paradigm. Even if the right choice of DRL algorithm and hyperparameters can lead to successful learning on real robots, efficiency and stability remain significant challenges in practice [205].

Note that there is some confusion around the meaning of *model-free* and *model-based* DRL in the literature. The latter traditionally encompasses *algorithms* leveraging a model of the environment (possibly learned online) to improve the learning efficiency by generating artificial trajectories, going beyond the standard trial-and-error nature of DRL. However, it can also refer to DRL *pipelines* requiring access to a model — typically for offline pre-training — even if the training algorithm itself does not use the model and still relies purely on trial and error. Throughout this work, we follow the latter definition, arguing that *any* need for a model, be it for pre-training or as part of the learning algorithm, inherently renders a method model-based.

Applying DRL to building control

Following its success on various tasks, DRL has recently also gained popularity in the context of building control, replacing its old tabular RL counterpart [14, 31, 41, 206]. However, it remains very *data-inefficient* in practice [29, 207], and DRL agents struggle to converge to meaningful solutions in a reasonable amount of time — mainly because of the slow thermal building dynamics. A vanilla *model-free* DRL agent can indeed take months or years to converge to satisfactory performance, all the while not guaranteeing the comfort of the occupants [14, 30, 31, 208]. Even after achieving satisfactory training performance, DRL agents might still react unexpectedly to new environmental conditions the NN-based policy has never seen before, i.e., fail to generalize (Section 2.1.1). Thus, neither R2–R3 nor R7 are generally met [13].

Final performance potential. On the other hand, if the agents continuously update their policy while interacting with the system, they naturally adapt to changing building dynamics and hence meet R4. Furthermore, although the final performance of DRL agents remains an open question in practice, there are indications that they can indeed find well-performing control policies [209–212], or at least achieve a performance close to MPC [213], hence addressing R1. This contradicts the findings in [13] to some extent, where a tuned MPC significantly outperformed many DRL agents with different hyperparameters. However, the MPC had perfect knowledge of all disturbances, and some hyperparameter choices still led to DRL agents achieving near-optimal performance. This confirms that finding the right hyperparameters and reward function to get the best performance out of DRL agents is generally challenging — and probably building-dependent [43].

Scalability and transferability potential. Since it learns solely from interactions with the environment, DRL is *system-agnostic* and hence presents great transferability potential (R6). Additionally, there are indications that DRL agents can scale to complex building control prob-

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

lems and hence satisfy R5 [14, 41, 206]. Scalability is also usually less challenging compared to optimization-based predictive methods since DRL policies only require a forward pass through an NN at inference time, a relatively computationally lightweight operation [213]. Furthermore, DRL agents only require access to the current state of the system online in principle, implicitly learning to anticipate future disturbances and bypassing the need for accurate forecasts of predictive methods, which also helps meet R5–R6.⁴

Meeting constraints (R3)

Safe DRL. First, one could borrow tools from the vast safe RL literature [216, 217] based on constrained policy optimization [218, 219], for example, to handle constraints and satisfy R3. Alternatively, one could learn neural Lyapunov or barrier functions in parallel with the control policy to guarantee the stability or safety of DRL agents [220]. However, these approaches would not improve the robustness to new disturbances (R2) while adding further computational complexity, hence impacting the scalability and convergence speed and jeopardizing R5 and R7 further.

Backup-DRL. To ensure the satisfaction of the occupants at deployment time and avoid catastrophic failures of the agents, many practical implementations of DRL in buildings leverage backup controllers [20, 214, 221, 222]. Usually, people either fall back on some known safe policy [223–225] or correct the actions of the agent [226] as soon as they are deemed *unsafe*.⁵ Classifying actions as safe or unsafe can be achieved by defining ad-hoc rules [222–225], constructing a shield from temporal logic specifications [227], learning when to switch [228, 229], or using a model (see below), among others. One key limitation here is that agents are *saturated* and usually cannot learn from their mistakes [226]; frequent interventions of the backup controller might hence hinder learning, leading to slower convergence R7 and potentially inducing sub-optimality R1.

Addressing robustness, constraint satisfaction, and convergence speed (R2, R3, and R7)

Despite the pitfalls associated with finding accurate building models discussed in Chapter 2, researchers often turn back to model-based methods to help DRL agents to satisfy R2, R3, and R7 in practice. Rather than solely relying on interactions with the system, as in the model-free case, Model-Based DRL (MBDRL) agents indeed leverage a model of the environment to help with robustness, constraint satisfaction, or data efficiency, i.e., reduce the number of interactions with the physical system. This section provides a non-exhaustive overview of possible approaches.

⁴Note that if disturbance forecasts are available, they can also be included in DRL applications, either extending the state space [43] or biasing Q-values towards forecasts for better estimation of future returns [214], for example. This might improve performance and should hence be considered whenever possible [212, 215].

⁵We follow the traditional characterization of *safe* and *unsafe* actions even if we are dealing with soft constraints that do not pose safety challenges per se. In this chapter, an *unsafe* action could be an action incurring discomfort to the occupants or leading to unreasonable energy consumption, for example.

3.1 The potential of model-free Deep Reinforcement Learning

Online MBDRL. To alleviate the issues linked to R7 to some extent, traditional MBDRL methods leverage models for planning — also known as *hallucinating* trajectories — during learning and artificially augment the data collection frequency, decreasing the number of interactions required with the environment [20, 230–234]. While this alleviates the sample complexity problem of model-free DRL agents to some extent (R7), it raises several new issues concerning the construction of an accurate model online to achieve scalability and transferability (R5–R6), as in the online MPC case discussed above. For example, enough data must first be collected before a well-performing and hence trustworthy model of the environment can be built and leveraged [235]. Furthermore, if the model is learned online, one has to ensure the DRL agent explores the environment sufficiently — but safely — to build a representative model [230]. Worryingly, a small bias in the model can significantly impact the final performance of DRL agents [236]. To make matters worse, optimizing the model accuracy can be uncorrelated with maximizing the rewards, leading to suboptimal control policies that maximize the rewards for the learned model but achieve suboptimal performance on the real system [237, 238].

All these problems are amplified if NNs are used as models, as is often the case in MBDRL, since their generalization issues could generate misleading trajectories and bias the control policy towards wrong behaviors, limiting the ability of such controllers to be robust to new disturbances and handle constraints to satisfy R2–R3 [20, 32, 210, 232, 234, 239, 240]. Note that using NNs is not necessary, and it might be possible to accelerate convergence (R7) by leveraging low-complexity models instead, as demonstrated in [241], for example.

Leveraging SIMBa. Alternatively, one could apply automatic linear or almost linear SI tools like SIMBa from Chapter 4 to design promising online MBDRL applications. A framework like SIMBa could indeed retain good model accuracy but low complexity, decreasing the amount of data required to find accurate models and thus potentially allowing for online-only MBDRL. As for other online methods, however, the model update procedure and frequency to achieve good performance over the lifetime of the building (R1, R4) remains an open question, hindering its widespread adoption and application to large-scale problems (R5–R6).

MPC-DRL: Merging MPC and DRL. To deal with R2–R3, several works proposed merging DRL's ability to learn online and MPC's robustness capability to handle constraints. For example, DRL has been used to modify the cost function parameters [242, 243] or learn the system model [238] of MPC frameworks online. Given the ability of DRL to account for long-term rewards, it can also be merged as the *final cost* in an MPC, shortening the horizon of the optimization problem to decrease the associated computational burden [239]. Alternatively, DRL has also been leveraged to learn lifting functions in Koopman-based approaches, inducing convex optimization procedures that are easier to solve online [244]. Conversely, inspired by MPC's predictive nature, a model can be leveraged to predict which actions are safe. One can then modify unsafe actions of DRL agents at each time step leveraging differentiable projections [245], solving an optimization problem [226], or using heuristic corrections [209, 246], for example.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

While it does alleviate issues of DRL related to robustness and constraint satisfaction (R2–R3), merging DRL and MPC however introduces additional computational complexity and reliance on models⁶ and disturbances forecasts,⁷ hence jeopardizing the inherent scalability and transferability of vanilla DRL methods (R5–R6), as discussed for MPC.

IL-DRL: Imitation learning. To augment the data efficiency of DRL agents, one can also leverage behavioral cloning [247] or learn from expert demonstrations [248, 249], such as applied for building control in [30, 42, 238]. We collectively refer to these approaches as *Imitation Learning* (IL) herein. The main idea is to *pre-train* DRL agents by first imitating another controller offline on a fixed data set, i.e., learning to replicate the behavior of the controller used for the data collection. This can potentially increase the robustness, the ability to handle constraints, and the convergence speed of DRL policies (R2–R3, R7), but it is naturally heavily influenced by the quality of the controller to imitate [153]. Furthermore, as a supervised learning task, it falls under the same generalization issues to unseen data as classical NNs (see Section 2.1.1): it might require access to a large historical data set and control policies solely learned offline cannot be expected to perform well in states that are not represented in the data [250].

The dependence on the data collection controller introduces new challenges on top of data availability in terms of scalability and transferability for IL. Indeed, since implementing well-performing building-agnostic controllers for large-scale problems is nontrivial, as discussed, having access to a good baseline to imitate in any large-scale case study is a strong assumption that is often not met in practice (R5–R6). Furthermore, even a pre-trained policy can still require one year to converge to the performance of a baseline controller when deployed in the building [251] — even if it converges faster than when learning from scratch, it might not be sufficient to satisfy R7. Nonetheless, other works reported promising results, with IL-based DRL policies achieving performance similar to the baseline after a few days only and rapidly learning to improve upon it in [30], for example. In general, IL allows one to warm start control policies and can be very useful in practice, but it should then be combined with other methods to ensure satisfactory performance on physical systems.

ReL-DRL: Residual learning. Rather than directly imitating a controller from data, if the baseline or expert policy is known, DRL agents can be trained to improve the performance of this base controller instead of learning everything from scratch, leading to various forms of Residual Learning (ReL) [212, 252, 253]. For example, one can separate the learning problem into different sub-tasks, letting DRL agents optimize the most complex ones while taking care of the other cases with RBCs [213]. Alternatively, both controllers can be merged, computing the final control input as a weighted sum of the prior and DRL controllers [246, 254]. In other lines of work, RBCs were used as guidance for DRL agents in the initial stage of learning in [255] or to restrict exploration by restraining the agent’s outputs to lie close to the RBC ones [212].

⁶Even if the model is learned, such as in Gnu-RL, finding the right structure is still not trivial [238].

⁷Except when the MPC is run with a single-step horizon [239].

3.1 The potential of model-free Deep Reinforcement Learning

ReL can significantly accelerate convergence [213] to help meet R7 but is naturally limited by the quality of the baseline controller, unless a fading mechanism is leveraged to decrease its importance over time, such as in [255]. ReL hence requires access to a well-performing prior controller, limiting its widespread adoption for large-scale applications, as discussed for IL (R5–R6). Finally, it does not solve robustness and constraint satisfaction issues in general since the DRL policy can usually overrule the baseline controller and behave similarly to a vanilla DRL agent (R2–R3).

Offline DRL. Instead of imitating or complementing a prior controller, one can also leverage a simulation environment to pre-train agents offline [64, 202, 203], as applied for building control in [15, 208], for example.⁸ We can expect these agents to satisfy R7, i.e., they should perform well since the start of the deployment in the physical building as long as the simulator is accurate.

Indeed, the main challenge stemming from offline DRL training is the *Sim2Real* gap, i.e., the performance drop between simulations and real-world deployments [257, 258]. Because of the generalization issue of NN-based policies (Section 2.1.1), adequate performance can not be ensured in situations not represented in simulation (R2) [259]. Soft data augmentation [260] or the more classical *domain randomization* [261] approach can alleviate this problem to some extent. The main idea behind domain randomization is to run extensive simulations with different parameters and disturbances to train DRL agents to react to any situation and constraints, which can help in satisfying R2–R3. One can go further and pre-train agents to react not only to various external conditions but also to building characteristics, temperature setpoints, or electricity prices, for example, similarly to what was done in [42], as a step towards satisfying R6. However, such a scheme drastically increases the training time required to incorporate all possible scenarios in DRL agents.

Note that there are no guarantees that such a pre-train agent will satisfy the comfort of the occupants after deployment (R3), as the constraints are not hard-coded in the control policy [123]. Additionally, with or without domain randomization, the quality of the learned policy naturally depends on the accuracy of the chosen simulator, which has to accurately capture the main building dynamics for DRL policies to perform well [20, 32, 239, 240]. While offline pre-training does help mitigate issues related to R2–R3, it might thus not always satisfy these requirements. In sum, it shifts the burden back to modeling [262] and hence suffers from the same pitfalls as the other offline model-based methods in terms of scalability and transferability (R5–R6).

Leveraging PCNNs. Remarkably, merging offline DRL with the state-of-the-art data-driven PCNNs provided in Chapter 2 to pre-train agents could potentially achieve impressive performance without any heavy engineering, as hinted in [154]. At the cost of significant com-

⁸Note that, contrary to MPC applications, the simulator does not require any specific (low-complexity) structure since no optimization needs to be run at each step [43]. Nonetheless, previous works hinted that low-complexity simulators could already allow DRL agents to learn desired behaviors, such as preheating, provided the model is physically consistent [256].

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

putational complexity to introduce extensive domain randomization analyses with PCNNs — to improve the robustness and constraint handling of DRL agents, going towards the satisfaction of R2–R3 — and then letting DRL agents learn over the entire deployment period to adapt to new building dynamics (R4), this could be a step towards controllers meeting all the requirements. However, this pipeline depends on the availability of large amounts of historical data to first fit PCNNs. This is typically unavailable in new buildings or buildings where one would want to install such new technology, limiting the scalability and transferability of such a pipeline (R5–R6). Furthermore, its final performance remains to be carefully analyzed (R1).

3.1.4 Towards computationally efficient constrained near-optimal online DRL

In light of the discussions above, the rest of this chapter aims to provide tools paving the way toward DRL agents able to fulfill all the seven requirements R1–R7. Although several interesting research directions exist, we **postulate that model-free DRL algorithms make ideal candidates to avoid the scalability and transferability issues linked to models (R5–R6).**

Additionally, vanilla model-free DRL agents performed close to MBDRL on two out of three case studies in [233], especially on the most complex one. This hints at their ability to compete with more computationally intensive model-based DRL solutions. Nagy et al. even reported results where the model-free version outperformed the model-based one [210]. On top of that, online model-free DRL can achieve performance close to offline pre-trained agents after a few weeks in [30, 43], suggesting that model-free DRL has the potential to rapidly perform comparably to its model-based pre-trained counterpart. Despite these promises, however, **there is still a need for efficient DRL agents converging in a reasonable time for any building (R7) and that can react to any disturbance (R2) and satisfy the comfort of the occupants at all times (R3), all the while achieving near-optimal performance (R1).**

Since we cannot afford to wait weeks for each DRL agent to converge on a physical building and assess its performance, we naturally turn back to simulations for our analyses, leveraging the PCNNs proposed in Chapter 2 as simulation environments in the rest of this chapter.⁹ This allows us to test various hypotheses and investigate solutions paving the way for system-agnostic model-free DRL building controllers satisfying R1–R7.

Summarized contributions

Despite adaptability, scalability, and transferability (R4–R6) being important requirements for real-world deployments, as discussed above, they are generally satisfied by model-free online DRL algorithms and are not considered in detail in the rest of this chapter. Instead, we focus on a relatively simple case study where DRL agents minimize the thermal energy consumption of a single zone while maintaining the temperature inside in a comfortable

⁹We also subsequently deployed one of the trained agents in the corresponding physical building for a qualitative assessment of the Sim2Real gap, confirming the validity of our results (Section 3.3.4).

range for the occupants. Our investigations can thus be seen as a proof of concept or feasibility study of model-free DRL controllers achieving near-optimality under any condition while guaranteeing the comfort of the occupant and converging in a reasonable amount of time, hence satisfying R1–R3 and R7. Since our implementations do not add any computational complexity, we argue they should not significantly hinder the ability of DRL controllers to be scaled and transferred to other buildings (R5–R6).

Near-optimality of model-free DRL. After some preliminaries in Section 3.2, Section 3.3 first provides an extensive analysis of the final performance achieved by model-free DRL agents in this case study, delivering additional evidence of their ability to reach near-optimal performance and thus meet R1. These investigations complement the results found in [209–211], but in several different settings and using a more accurate simulation environment.

Safe and computationally efficient DRL. As a first step towards the satisfaction of R2–R3, and R7, Section 3.4 then analyzes one way to transfer expert knowledge to DRL agents. This will allow us to simultaneously guarantee adequate indoor thermal comfort for the occupants in any conditions (going towards the satisfaction of R2–R3) and avoid exploring sub-optimal state-action pairs for faster convergence (R7). Remarkably, the proposed interventions are computationally lightweight to avoid jeopardizing scalability and transferability (R5–R6).

Altogether, while limited to a single case study and the low-complexity framework of only controlling one zone temperature, our investigations point towards the ability of model-free DRL controllers to meet R1–R7. In practice, this would mean **such controllers could be deployed from scratch in any building and learn to optimize their operations, bypassing the need for historical data or accurate models while being computationally inexpensive.**

3.2 Preliminaries

3.2.1 Basics of Reinforcement Learning

RL problems are usually formulated as Markov Decision Processes (MDPs), which are represented by tuples $\langle S, \mathcal{A}, \mathcal{P}, \rho_0, r, \gamma \rangle$, where S is the state space, \mathcal{A} the action space, $\mathcal{P} = \mathcal{P}(s'|s, a)$ the probability of transitioning from state $s \in S$ to $s' \in S$ when action $a \in \mathcal{A}$ is taken, ρ_0 the initial state distribution, $r = r(s, a)$ the reward function, and $0 < \gamma < 1$ the discount factor [19].

At each time step t , given an observation s_t of the state of the environment, the RL agent chooses a_t . The environment then transitions to s_{t+1} according to \mathcal{P} and sends the new state and $r(s_t, a_t)$ to the agent. The objective of any RL algorithm is to find a *policy* $\pi(a|s_t)$ that maximizes the expected discounted cumulative returns:

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{P}(s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \quad (3.1)$$

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

To that end, many algorithms rely on learning an approximation of this objective, namely the *Q-function*, which estimates the expected return the agent will receive if it takes action a in state s and then follows the current policy π :

$$Q^\pi(s, a) = \mathbb{E}_{a_t \sim \pi(\cdot | s_t), s_{t+1} \sim \mathcal{P}(s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right]. \quad (3.2)$$

The Q-function and policy are often dubbed the *critic* and *actor*, respectively.

In our experiments, we rely on the ϵ -greedy strategy for exploration, i.e., we apply the following action to the environment at each step during training:

$$a(s) = \text{clip}(\pi(s) + \epsilon, a^{low}, a^{high}), \quad \epsilon \sim \mathcal{N}(0, \sigma), \quad (3.3)$$

where the noisy actions are clipped element-wise between a^{low} and a^{high} , the predefined action bounds from the environment, and ϵ is the Gaussian exploration noise with a standard deviation of σ . All the transition tuples (s_t, a_t, r_t, s_{t+1}) observed by the agent are stored in a *replay buffer*.

3.2.2 Actor-critic algorithms

In practice, policies and Q-functions are often parametrized with NNs as π_θ and Q_ϕ , respectively, leading to DRL. Numerous algorithms have been developed to maximize (3.1) [263]. Among the countless improvements and techniques presented in various contributions, we can point out the influence of target networks, which we use in this work [264]. The idea is to keep a copy of the actor and critic in memory, only updating them slowly to decrease the usual overestimation bias of Q-values and stabilize the learning process.

In this work, we are interested in deterministic actor-critic methods stemming from the Deep Deterministic Policy Gradient (DDPG) algorithm [265], where both π_θ and Q_ϕ are optimized in parallel leveraging gradient descent. While different flavors exist, most actor-critic algorithms compute the gradient of the critic using a variant of the Temporal Difference (TD) loss [265]

$$\hat{\nabla}_\phi Q_\phi = \nabla_\phi \left[\frac{1}{|\mathcal{Z}|} \sum_{(s, a, r, s') \in \mathcal{Z}} (Q_\phi(s, a) - y(a, r, s'))^2 \right] \quad (3.4)$$

$$y(a, r, s') = (r + \gamma Q_{\phi, targ}(s', \pi_{\theta, targ}(s'))) , \quad (3.5)$$

where a batch \mathcal{Z} of past transitions is sampled from the replay buffer and used to estimate expectations. Leveraging the policy gradient theorem [266], one can similarly use the critic to estimate the actor gradient as

$$\hat{\nabla}_\theta \pi_\theta = -\nabla_\theta \left[\frac{1}{|\mathcal{Z}|} \sum_{s \in \mathcal{Z}} Q_\phi(s, \pi_\theta(s)) \right]. \quad (3.6)$$

Note that these gradients are easily computed using automatic differentiation when the actor and the critic are parametrized with NNs.

In this chapter, we rely on the Twin Delayed Deep Deterministic (TD3) policy gradient algorithm, which introduces a few modifications over DDPG to limit the well-known overestimation bias of Q-functions plaguing vanilla actor-critic algorithms [267]:

- Inspired from the success of Double Q-learning [268], two critic networks are learned in parallel, and the smallest of the two approximated Q-values is used as the target in (3.5) to limit overestimation.
- To avoid the policy exploiting overestimated Q-values, noise is added to the action a' before it is evaluated by the critics in (3.5).
- To avoid instability arising from fast-changing Q-functions, the actor and target networks are updated less frequently than the critics.

Remarkably, however, these adjustments do not impact the actor gradient in (3.6), allowing us to seamlessly integrate the proposed gradient modifications detailed in Section 3.4 into TD3.

3.2.3 Problem setting

Action space. Throughout this chapter, the task of the agents is to control the heating or cooling power of one bedroom in UMAR (i.e., Zone 1 or 3 from Section 2.4.1) at each time step of 15 min. To be specific, we defined $\mathcal{A} = [-1, 1]$, i.e., $a^{low} = -1$ and $a^{high} = 1$, which is then linearly transformed to power inputs being applied to the zone as

$$P_t = \begin{cases} \frac{a_t+1}{2} P_{heat}^{max} =: V_t P_{heat}^{max}, & \text{in the heating case,} \\ \frac{1-a_t}{2} P_{cool}^{max} =: V_t P_{cool}^{max}, & \text{in the cooling case,} \end{cases} \quad (3.7)$$

where P_{heat}^{max} and P_{cool}^{max} stand for the maximal heating and cooling power, respectively.

We also use the valve opening percentage $V_t \in [0, 1]$ — assumed to be proportional to the thermal power input to the zone throughout this work — when controlling the physical building. Indeed, we cannot directly control individual zone power inputs during real-world experiments, only the valves, and we can only fully open or close them.¹⁰ We hence turn to pulse-width modulations: if the agent decides to use the full power, we open the valves for 15 min, and if it wants to use a fraction of the power instead, we open the valves for the corresponding fraction of time.

Remark 16 (Valve openings as proxy for thermal power). *In practice, the amount of energy transferred to the room also depends on the temperature gradient between the water in the pipes*

¹⁰In fact, we only have access to the temperature setpoint in each zone of the physical building, with an internal mechanism triggering the valves. However, writing a very high or very low setpoint, e.g., 28 °C or 16 °C, invariably fully opens the valves or closes them, respectively, in the heating case, and conversely during the cooling season. Although the valves might sometimes take time to open and close, we assume throughout this work to be able to either fully open or close the valves at any given time.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

and the air in the zone. We do not have control over the water temperature but it can be assumed to stay roughly constant in the case of UMAR. Since heating/cooling are always required at similar zone temperatures — to maintain the comfort of the occupants —, the latter can also be assumed to be approximately constant during experiments. This leads to a roughly constant temperature gradient, which makes the valve opening percentage approximately proportional to the maximum available thermal power, as assumed herein.

State space. Throughout our analyses, we will use the PCNNs developed in Chapter 2 as simulation environments. Consequently, the state s observed by the agents at each time step is similar to the PCNN inputs: it is composed of zone temperatures (of the controlled and neighboring zone), ambient conditions (the ambient temperature and solar irradiation measured on a flat surface), and time information (sine and cosine functions of the month of the year and time of the day, and the day of the week, as discussed in Appendix A.4). Additionally, agents know the current temperature comfort bounds and the *case* they are in.¹¹

To have more expressive policies aware of the evolution of the environment in time, we also add 12 autoregressive terms of the zone temperatures and ambient conditions so that agents know the state of these variables during the last three hours when making decisions.

Reward function design. All agents aim to minimize energy consumption while respecting predefined dynamic temperature comfort bounds for the occupants. While we require the temperature to stay between 23 and 24 °C during the night — from 20 h to 8 h —, it can be relaxed during the day, when bedrooms are generally unoccupied. Consequently, the lower bound is relaxed by two degrees, from 23 °C to 21 °C, during the day in the heating case. Conversely, the upper bound is increased by two degrees throughout the day in the cooling case. Comfort violations over a given period are then expressed in Kelvin Hours, summing the difference between the zone temperature and the bounds at each time step.

Mathematically, throughout our experiments, DRL agents optimize a trade-off between energy consumption and comfort violations, maximizing the following reward function:

$$r(s_t, a_t) = -\max\{L_t - T_t, T_t - U_t, 0\} - \alpha P_t, \quad (3.8)$$

where L_t and U_t represent the lower and upper comfort bounds on the temperature T_t at time t , respectively, and α is a weighting factor. As a rule of thumb, we designed the nominal value of α such that agents receive the same penalty for using a power of 1 kW and for being 0.5 °C outside of the comfort bounds.

Remark 17 (High temperature bounds). *The comfort bounds used throughout this chapter correspond to relatively high indoor temperatures during the heating season; the lower bounds*

¹¹We refer to the *case* as whether the system is in heating or cooling mode. This is required since UMAR is heated/cooled by letting hot/cold water flow through the ceiling panels. On the other hand, agents can only open or close the corresponding valves; they cannot modify the water temperature and decide whether to heat or cool. In other words, if the zone temperature is too high and the system is in heating mode, DRL agents cannot cool the room, and the optimal decision is to do nothing.

could be decreased in practice to avoid wasting energy. However, we choose 23°C here to reflect the measured temperatures in UMAR in winter and create a challenging control scheme: due to the sensitivity of UMAR to solar gains, the temperature inside is often maintained above 21°C without heating. Consequently, setting artificially high comfort bounds ensures the controllers can be compared on a nontrivial task.

Hyperparameters. In our implementations, we arbitrarily chose three hidden layers of 512 neurons for all the NNs — the critic, the actor, and their target networks — to ensure policies could be expressive enough. We use a slightly modified version of the Adam optimizer [269] with a learning rate of 10^{-4} , and we rely on the TD3 implementation from the tianshou library [270]. Finally, we set $\gamma = 0.95$.

Remark 18 (Discount factor). *Note that, for building control applications, the discount factor gamma has to be selected close to one. Indeed, an optimal controller should present preheating and precooling behaviors, i.e., it should start heating or cooling the room at the end of the afternoon to meet the comfort bounds tightening at 20 h every day. This costs more energy in the early evening but avoids later penalties due to comfort violations. Such desired behaviors can only be captured by controllers that are not too myopic, i.e., take long-term rewards into account. This is not the case when $\gamma \ll 1$, as also noted in [271], for example.*

Training procedure. To train and evaluate the agents, we create 15 h to 75 h-long sequences of data with no missing values,¹² hereafter referred to as *trajectories*. They were processed from three years of operational data as explained in Section 2.4.1 and separated into a training and a validation set. The agents are trained on trajectories randomly sampled from the training set and regularly evaluated on the validation set to assess their performance. To learn policies robust to measurement noise, we add independent Gaussian noise to the zone temperature measurement during training.

3.2.4 Baselines

To analyze the performance of the DRL agents, we compare them to two classical RBC baselines. *Baseline 1* is tracking a reference 0.5°C away from the bound, turning the heating on and off as soon as the target temperature is met during the heating season. *Baseline 2* is a classical rule-based controller with a one-degree hysteresis, i.e., it starts heating at full power when the temperature reaches the lower bound and until one degree has been gained. During the cooling season, both baselines use similar strategies, starting to cool once the temperature reaches the upper bound or the reference half a degree from it.

Since we randomly sample a trajectory from historical data at the beginning of each episode, the initial zone temperature might be out of the comfort bounds, leading to *unavoidable*

¹²Each sequence is composed of 3 h of past data — required to warm-start the PCNN simulator from Chapter 2 and build the autoregressive terms of the agents' observations — and a 12 h to 72 h prediction horizon under the control of the agents.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

penalties for any controller. We thus implemented an additional agent fully opening or closing the valves¹³ until the zone temperature reaches the bounds for the first time in each episode.¹⁴ This allows us to keep track of these unavoidable comfort penalties throughout our experiments.

Remark 19 (The difference between both bedrooms). *The two bedrooms in UMAR have a similar architecture. We took advantage of that situation during real-world experiments, deploying a rule-based controller in the bedroom not controlled by the DRL agent for qualitative comparison purposes. However, since one bedroom is adjacent to another unit while the other has two external walls, they present slightly different thermal dynamics. Nonetheless, it still allows for analyses of the behavior of both controllers under similar external conditions.*

3.3 Deep Reinforcement Learning can achieve near-optimal performance

Before any DRL controller can achieve widespread adoption, guarantees of near-optimal performance in general are required (R1). In particular, this must be independent of the desired trade-off between energy consumption and comfort violations — α in (3.8) — or choice of random seed, to which DRL is known to be sensitive [272]. However, while DRL has been compared to rule-based controllers in the literature, for example, in [15, 20, 32, 116, 256, 273], it was rarely benchmarked against advanced control methods, apart from limited comparisons to MPC in [13, 43, 44, 213, 233], for example.

In particular, the *optimality gap* of DRL agents has rarely been addressed in building control applications [274]. Notable exceptions include the analyses in [209–212], but these studies are limited to the investigation of a single agent in simplified first-principles-based simulators. Thus, they shed only little light on the ability of DRL agents to find well-performing control policies in general. On the other hand, the often-used complex building models based on EnergyPlus [106] or NNs [15]) are often highly nonlinear, making it difficult to compute the optimal control inputs for benchmarking purposes and explaining the lack of studies on the optimality gap.

Leveraging PCNNs. In this chapter, we proposed to use the PCNNs developed throughout Chapter 2 as simulators. As discussed in Chapter 2, they achieve state-of-the-art modeling performance without jeopardizing physical consistency, making them suitable simulators for DRL applications while incorporating nonlinear behaviors typically not captured by first principles models. Remarkably, the PCNNs proposed in Chapter 2 are input-affine (Remark 9), allowing us to set up tractable Linear Programs (LPs) for the optimization of control inputs.

¹³Depending on whether the initial temperature is too high or too low and whether we are in the heating or cooling season.

¹⁴An example is pictured in Figure 3.2, where the temperature is too high at the trajectory during the heating season. All controllers thus first have to wait for the temperature to drop to the comfortable range and receive unavoidable comfort violation penalties.

3.3 Deep Reinforcement Learning can achieve near-optimal performance

In the rest of this section, we analyze the impact of different trade-offs in the reward function and random seeds on DRL agents and compare them to the theoretical performance upper bound that could be achieved with perfect knowledge of all the disturbances. With these investigations, we provide more indications toward the ability of model-free DRL agents to achieve near-optimal performance in various settings (R1). However, this in-depth analysis is limited to a single case study — one of the two bedrooms in UMAR —, and whether such conclusions could be drawn for any building in general remains an open question.

3.3.1 Computing optimal control inputs with PCNNs

To compute the optimal power inputs on a validation trajectory s of length l_s , we assume perfect knowledge of all the disturbances T^{out} , T^{neigh} , and D over the horizon. Given a PCNN (2.4)–(2.6), we need to solve the following optimization problem, where the objective function is designed to match the reward of the agents and zone indices were dropped for clarity since agents only control a single zone:

$$\min_{P_0, \dots, P_{l_s-1}} \sum_{k=0}^{l_s-1} \tilde{\alpha} P_k + \epsilon_{k+1}^L + \epsilon_{k+1}^U \quad (3.9)$$

$$s.t. \quad E_{k+1} = E_k + qP_k - b(T_k - T_k^{out}) - c(T_k - T_k^{neigh}) \quad (3.10)$$

$$T_{k+1} = E_{k+1} + D_{k+1} \quad (3.11)$$

$$E_0 = 0$$

$$L_{k+1} - \epsilon_{k+1}^L \leq T_{k+1} \leq U_{k+1} + \epsilon_{k+1}^U, \quad \forall k = 0, \dots, l_s - 1, \quad (3.12)$$

$$P_{cool}^{max} \leq P_k \leq P_{heat}^{max}, \quad \forall k = 0, \dots, l_s - 1.$$

Here, we solve for the optimal thermal power inputs, which appear linearly in the objective function (3.9) and constraint (3.10), showing it is indeed an LP. Note that knowing the optimal thermal power inputs directly allows one to recover the optimal control inputs from (3.7).

We use $\tilde{\alpha}$ in the LP, which equals the weighting factor α in heating cases and $-\alpha$ during the cooling season, so that it always penalizes the absolute value of the power used, as in the reward function (3.8). Similarly, ϵ_{k+1}^L and ϵ_{k+1}^U capture comfort violations in (3.12) and are penalized in the objective function to reflect the reward function of the agents. Since we know each bedroom only has a single neighboring zone and an external wall, (2.5) simplifies to (3.10), where q stands for either a_h or a_c in (2.5) depending on the case.¹⁵

The key property of PCNNs rendering this optimization feasible is the fact that the highly nonlinear unforced dynamics D are *independent* of the power inputs P and can hence be computed *a priori* for the entire horizon (knowing all external conditions). They are then used as external variables to compute the temperature evolution in (3.11). This LP can be solved very efficiently with common tools, which allows us to compute the optimal solution

¹⁵We assume the case to be fixed for any given trajectory, i.e., the system is either in heating or cooling mode and does not switch over the horizon, which allows us to set the value of $\tilde{\alpha}$ and q before the optimization.

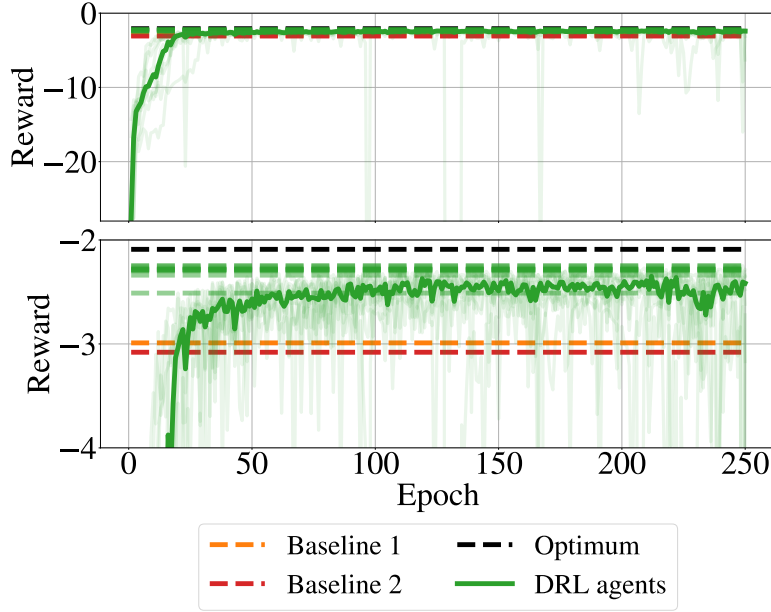


Figure 3.1: Convergence rate of DRL agents with different random seeds in green, with the median in bold and the maximal reward attained by each agent in dashed lines. For reference, the baseline and optimal performance are also shown in dashed lines.

for thousands of trajectories in a feasible time.¹⁶

3.3.2 Performance analysis

The impact of randomness

Since DRL policies are notoriously sensitive to randomness [272], we first trained 10 agents with different random seeds to compare their performance. Figure 3.1 reports the average performance of each agent on a fixed set of 50 sequences from the validation set after each training epoch, with the median performance of the 10 agents in bold. The bottom plot is a zoomed-in version of the top one for clarity. Here, one epoch represents 5'000 training time steps in the environment, i.e., slightly over 50 days of data from the training set. To complete these results, the best performance attained by each agent over the training horizon is reported in dashed lines in Figure 3.1 and in Table 3.2, along with the rewards that the two rule-based baselines achieve on the 50 validation sequences and the optimal ones computed from (3.9).

Overall, one can see the performance of each agent fluctuating significantly along training, but most agents could find a control policy achieving rewards between -2.25 and -2.3 on the 50 validation sequences. All DRL agents present comparable learning patterns: they converge to policies achieving similar performance to the baselines after approximately 20 epochs and

¹⁶The code and data can be found on <https://gitlab.nccr-automation.ch/loris.dinatale/NoDRL>.

3.3 Deep Reinforcement Learning can achieve near-optimal performance

Agent	Rewards	Agent	Rewards	Agent	Rewards
Baseline 1	-2.99	DRL agent 1	-2.29	DRL agent 2	-2.25
Baseline 2	-3.08	DRL agent 3	-2.27	DRL agent 4	-2.51
Optimum	-2.09	DRL agent 5	-2.25	DRL agent 6	-2.34
		DRL agent 7	-2.28	DRL agent 8	-2.29
		DRL agent 9	-2.30	DRL agent 10	-2.28

Table 3.2: Performance of all the controllers on the 50 fixed trajectories from the validation set compared to the optimal one, reported from Figure 3.1. For DRL agents, this corresponds to the best-attained rewards on this set over the training period.

consistently outperform them afterwards. Nonetheless, we can see a few exceptions along the training pattern, with some agents’ performance plummeting for a few epochs but recovering quickly. It is also noteworthy that one of the agents performed significantly worse than the others, only reaching a best reward of -2.51 . This proves that DRL agents are also sensitive to the choice of random seed in our setting, one should always train several agents to rule out the possibility of the random seed significantly impacting the quality of the results.

Deep Reinforcement Learning agents learn expected behaviors

Interestingly, all agents captured the expected and desired preheating and precooling behaviors. Indeed, they learned to take action earlier than the other controllers, especially the rule-based ones, to anticipate constraint tightenings, as pictured in green Figure 3.2, for example. They usually heat or cool the zone until the temperature is slightly above or below the comfort bound in the late afternoons in the heating or cooling case, respectively, and then stop to avoid wasting energy. This often results in the temperature reaching the bound again just before the constraints are relaxed at 8 h. Overall, this strategy is not far from the optimal one in black, which consists of waiting until the last moment to preheat or precool the room at full power to meet the constraint tightening exactly. It then leverages its full knowledge of the environment to input just enough energy in the system for the temperature to stay exactly at the desired limit and avoid comfort penalties unless, for example, solar heat gains are expected to create more violations later. Finally, due to their reactive nature, both reactive baselines in red and orange cannot anticipate constraint tightenings and relaxations, leading to comfort violations in the early evening hours. Furthermore, they do not account for the impact of future disturbances — typically solar gains —, leading to overheating and overcooling behaviors, for example, towards the end of the horizon in Figure 3.2.

In the particular case of Figure 3.2, the DRL agent consumed slightly more energy than the optimal solution, 5.47 kWh against 5.32 kWh, and incurred slightly more comfort violations, 2.9 Kh against 2.1 Kh.¹⁷ This difference mainly comes from the DRL agent slightly heating the room after midnight on March 20, contrary to the optimal solution, which also incurs slightly

¹⁷After subtraction of the unavoidable comfort violations of 11.9 Kh stemming from the very high temperature at the beginning of the trajectory.

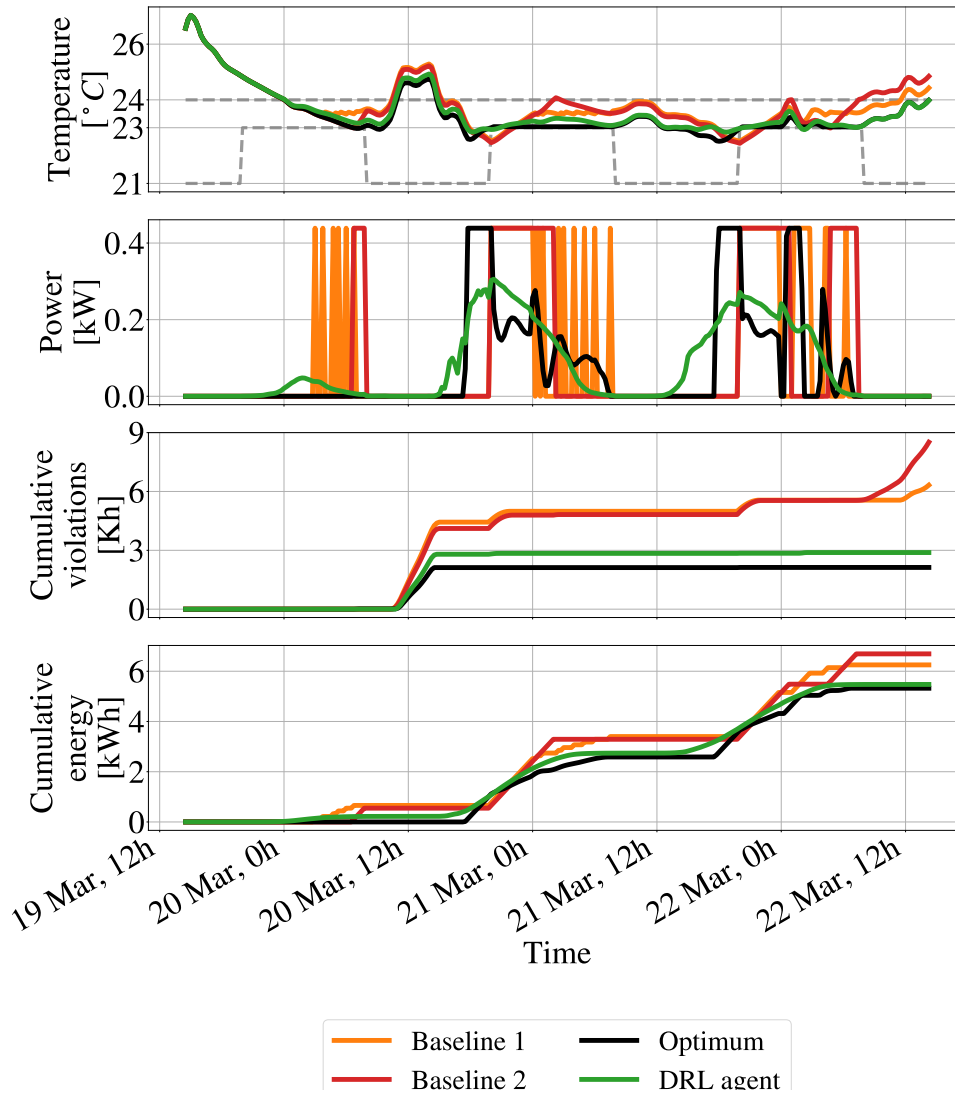


Figure 3.2: Example of the behavior of a DRL agent over 3 days, compared to the baselines and the optimal solution. The zone temperature (with the comfort bounds in dashed gray) and corresponding thermal power input are respectively reported in the first two plots. The bottom two gather the cumulative comfort violations and energy consumption of each controller.

3.3 Deep Reinforcement Learning can achieve near-optimal performance

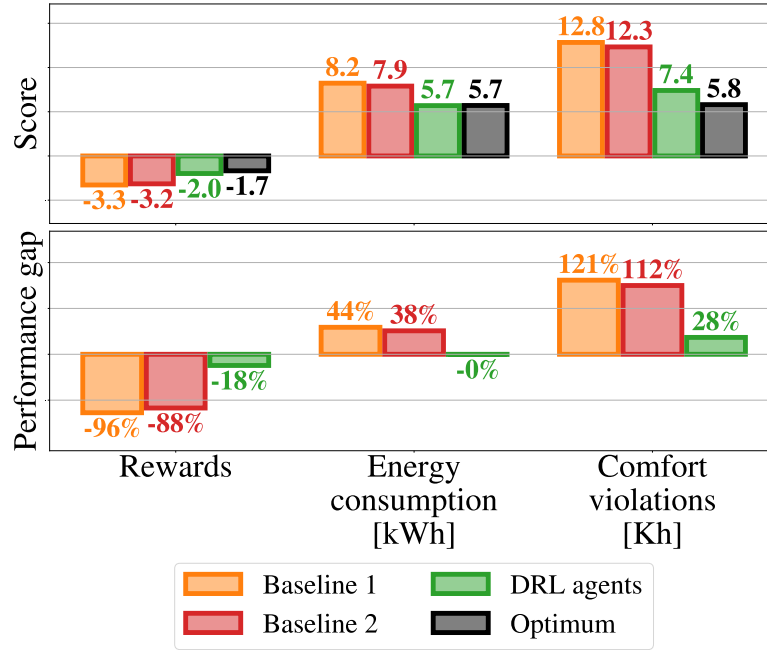


Figure 3.3: Average rewards, energy consumption, and comfort violations over three days achieved by the baselines and DRL agents compared to the optimal solution and after the subtraction of the unavoidable penalties, and the corresponding performance gaps.

higher temperatures over the day and hence more comfort violations.

In general, DRL agents tend to converge to risk-averse policies in our setting because of the noisy observations returned by the environment during the training phase. Indeed, they usually slightly overheat or overcool the zone when the constraints are tightened, ending up further away from the bounds than strictly necessary. This allows them to avoid comfort penalties stemming from the noisy temperature measurement jumping outside the comfort zone they observed during training.

Generalization to the entire validation data set

To estimate the best performance of DRL agents, this section focuses on the policies achieving the maximum reward on the 50 validation trajectories for each agent. We analyzed their average performance on almost 2'000 three-day-long trajectories from the validation set in terms of rewards, energy consumption, and comfort violations. In the top plot of Figure 3.3, one can observe the mean performance achieved by the different random seeds on each metric, compared to the baseline and the optimal ones, after subtraction of the unavoidable penalties. The bottom plot then shows the corresponding performance gap of each controller compared to the theoretical optimal one.

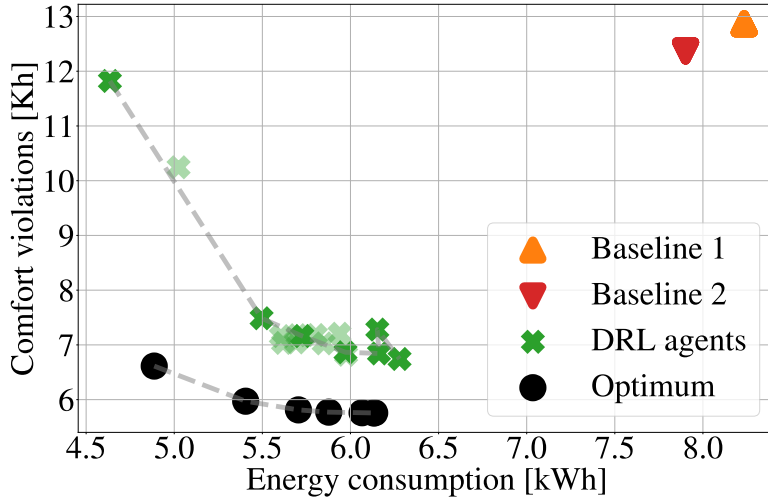


Figure 3.4: Sensitivity analysis of DRL agents and the optimal solution to different weighting factors in (3.8), decreasing it from left to right along the dashed lines. The trade-off obtained by both baselines is also plotted for reference, as well as the one struck by the agents ran with different seeds in Section 3.3.2 in shaded green crosses.

Notably, with the chosen parameters, DRL agents could converge to a near-optimal solution. They found a relatively similar trade-off between energy consumption and comfort violations as the optimal solution, consuming the same amount of energy at the cost of slightly increased comfort violations of a little over 25%. This is, however, still significantly better than what the baselines can do, both in terms of energy savings and comfort improvements, where their performance drop is over 35% and 110%, respectively. Altogether, these results confirm that **DRL agents can converge to policies that not only significantly outperform classical controllers but simultaneously attain near-optimal performance.**

3.3.3 Sensitivity to the weighting factor

To assess the impact of the weighting factor in (3.8), this section investigates the trade-offs between energy consumption and comfort violations reached by all the controllers for different choices of α . To that end, we trained agents in the same environment, with a fixed random seed, but multiplying or dividing the nominal α by increasing powers of two to reflect situations where more and more importance is put on decreasing the energy consumption or the amount of comfort violations, respectively. The resulting trade-offs between both objectives for the DRL policies and the optimal solutions are plotted in Figure 3.4, where the weighting factor was decreased from 4α to $\frac{1}{16}\alpha$ from the left to the right along the dashed Pareto frontiers. The performance of the baselines is also reported for reference.

As can be observed, the agent could generally strike a trade-off similar to the optimal one, usually consuming approximately the same amount of energy at the cost of slightly more

3.3 Deep Reinforcement Learning can achieve near-optimal performance

comfort violations, as long as the weighting factor is not too big. Once α is multiplied by 4, the agent indeed struggles to find an interesting solution: it uses very little energy but does not improve the comfort of the occupants much compared to the baselines (top left green cross). When we tried to increase α by a factor of 8, the agent quickly converged to a very poor policy that never used energy at all, and this result is hence discarded here. On the other hand, decreasing the weighting factor impacts the quality of the solution less: policies consume slightly more energy each time, slowly reducing the amount of comfort violations.

It is important to remember here that the choices of random seed again impact the results, which could explain why the trade-off obtained by the agent with a weighting factor of $\frac{1}{8}\lambda$ is slightly higher than the Pareto front. This is confirmed by the shaded green markers showing the trade-offs achieved by the agents discussed in Section 3.3.2, where we see the impact of different random seeds for the same weighting factor. All random seeds usually lead to similar solutions, but we can also observe the outlier pointed out previously (top left shaded green marker). Interestingly, despite obtaining worse rewards than all the other agents, it still lies near the Pareto front. In other words, it seems this agent converged to a solution that was not optimal in this situation but might be the expected behavior under different circumstances.

To summarize, one should be careful with the design of the weighting factor, as it might impact the quality of the solution. However, a wide range of values — from 2α to $\frac{1}{16}\alpha$ — could be selected and still lead to near-optimal behaviors in this case study. Remarkably, **this parameter can thus be adjusted to reflect the preferences of the building occupants without significant performance drop**. In general, choosing a value that is too large seems to be more problematic than the contrary. Finally, while the random seed seems to impact which region of the Pareto frontier the solution converges to, our experiments hint that DRL agents usually strike a trade-off near the Pareto frontier of all DRL policies. However, they diverge from it when they try to use too little energy, confirming that small values for α may be safer

3.3.4 Real-world deployment

To confirm that PCNNs provide a meaningful simulation environment to train DRL control policies, we deployed one of the above-analyzed agents in the physical building to assess the Sim2Real gap. Taking advantage of UMAR having two almost identical bedrooms, we simultaneously deployed Baseline 2 in the other one, and the resulting temperatures and control inputs are reported in Figure 3.5, adapted from [154]. As mentioned in Section 3.2, since we do not have access to the thermal power input to the rooms in the physical building, we used the valve opening as a proxy for energy consumption (see Remark 16).

Remarkably, the agent and the baseline both show similar temperature behaviors to what was observed in simulation, in Figure 3.2, for example. Despite the short span of the experiment, the DRL agent maintained the temperature close to the lower bound, as expected, thereby saving energy compared to the hysteresis controller — except when the default controller took over when the connection was lost for a few hours (shaded gray).

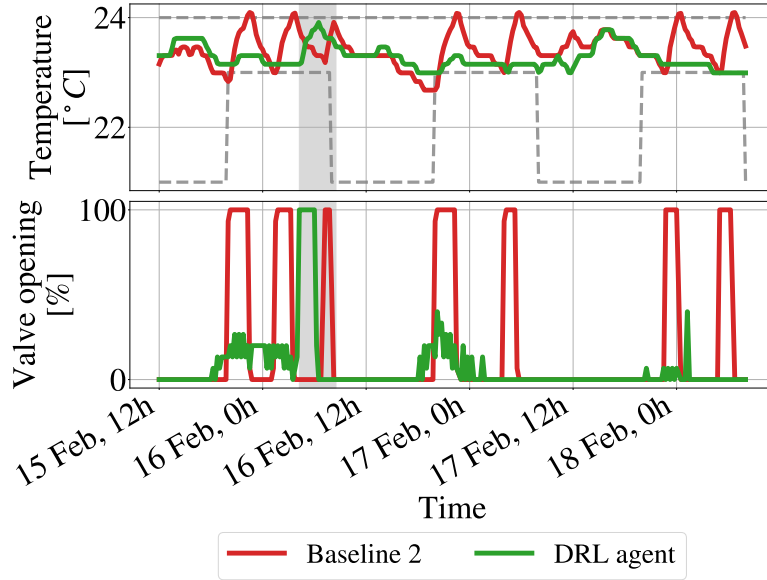


Figure 3.5: One bedroom of UMAR controlled by a classical rule-based controller (red), the other by the proposed DRL agent (green), adapted from [154]. The connection was interrupted in the shaded area and default controllers took over.

This brief investigation **hints that PCNNs provide accurate simulation environments to train meaningful DRL policies and that the results obtained throughout this chapter are likely to extrapolate to the physical building despite the Sim2Real transfer.**

3.3.5 Achieving near-optimal performance has a cost

Overall, these investigations on the impact of the random seed and weighting factor balancing energy consumption and comfort violations indicate that DRL control policies generally not only outperformed rule-based baselines but could also achieve near-optimality. Specifically, although different random seeds sometimes lead to lower-quality solutions in terms of rewards, they all seem to lie near the Pareto frontier of energy consumption and comfort violations. Remarkably, various choices of reward functions led to near-optimal solutions — until α is chosen too large. Altogether, this **provides nonnegligible evidence of the ability of DRL to achieve near-optimal performance in diverse settings and hence satisfying R1.**

On the other hand, one should keep in mind that these results are limited to a single-zone temperature control case study and might not generalize to different buildings or more complex case studies. Additionally, in line with previous works discussed in Section 3.1.3, these DRL agents **strongly suffer from data inefficiency, taking 20 episodes of more than 50 days of data — almost three years (!) — to converge to a performance comparable to RBCs.** Consequently, the next section introduces a few modifications to vanilla DRL agents to improve their convergence speed, towards the satisfaction of R7.

3.4 Constraining agents and accelerating convergence

As detailed in Section 3.1.3 and confirmed by our investigations in Section 3.3, vanilla model-free DRL agents are severely impacted by their data inefficiency, i.e., they typically require a significant number of interactions with the environment to converge. This mainly stems from the trial-and-error paradigm at the core of RL algorithms, which relies on an extensive exploration of the state-action space to find optimal policies. This not only leads to significant computational costs but also limits the deployment of DRL methods on physical systems without pre-training in simulation [64, 202, 203].

To speed up the training of DRL agents, researchers have investigated how to leverage expert demonstrations [248, 249], but this requires access to an expert policy that is not always available in practice. Instead, we postulate in this section that **prior knowledge of physical systems often allows us to design simple rules that RL agents should follow *a priori***, such as “Do not heat the room if it is already 26°C”; we indeed know this action will *always* be suboptimal in that state, there is no need for agents to explore its consequences. Critically, this rule applies to *any building* and does not jeopardize R7. Remarkably, incorporating such expert knowledge in control policies has already been identified as a promising step towards more efficient physics-informed RL algorithms [275].

Constraining and accelerating DRL. In this section, we propose Efficient Agents (EAs), which incorporate computationally lightweight modifications of actor-critic algorithms to encode simple rules in DRL agents. Specifically, we introduce *artificial state-dependent constraints* on the agents’ actions to restrict exploration to interesting regions of the state-action space. In other words, the key idea is to avoid visiting state-action pairs that are known to be suboptimal by the expert to accelerate the convergence towards meaningful solutions and thus increase the efficiency of DRL (R7). In the case of building control, we leverage this framework to concurrently ensure DRL agents do not violate the occupants’ comfort too often and react as expected to external disturbances, towards the satisfaction of R2–R3.

Note that while state-dependent bounds were concurrently introduced in [276], they are enforced *a posteriori* in the environment instead of directly modifying the agent’s behavior and do not necessarily improve convergence speed (R7). On the other hand, prior knowledge successfully accelerated learning in [277]. However, rules were only used to *guide* DRL agents, there is hence no guarantee that the desired prior knowledge will be respected (R2–R3).

3.4.1 Constraining Reinforcement Learning agents

To bound the decisions taken by RL agents, one typically defines some constrained set of actions and either projects the actions of the agents on this set at each time step or switches to a fallback controller when needed [223–225].¹⁸ The main challenge with these operations is that

¹⁸In the case of discrete action spaces, one can also *mask* unsafe or undesired actions, avoiding the need for backup controllers, such as in [212, 278].

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

they are generally *not differentiable* and hence cannot be learned by RL agents, i.e., they will never correct their mistakes if no additional mechanism is used to let them know when they have been saturated. Three notable exceptions were provided in [245, 279, 280], who leveraged differentiable optimization layers [245], modified the policy updates to account for projections [279], and used linear constraints to derive a closed-form solution of the projection step [280]. This linear assumption was lifted in [281], but only to correct the agents' actions online, as it cannot be used in training since no closed-form solution exists anymore. However, these methods either entail additional computational burden and require access to a model of the system [245, 279] or rely on a learned linearization of the constraints [280]. Alternatively, one could apply tools from the safe RL literature relying on constrained policy optimization [216, 218, 219]. However, this would again introduce engineering and computational overhead, defeating the main purpose of this section, which aims at reducing the computational burden of actor-critic algorithms by leveraging simple engineering intuition.

State and action constraints. The complexity of the methods discussed above often stems from the fact that they are designed to impose *state constraints* on DRL agents. This is a more challenging problem since it generally leads to complex safe action sets for the agent at each step. Here, however, we argue that prior knowledge can straightforwardly be used to accelerate the training of DRL agents through simple state-dependent *box constraints on their actions*, which allows us to leverage less computationally intensive tools.

To alleviate the issue of non-differentiability of the projection without increasing either the engineering or the computational burden, one can let agents learn when to switch to the fallback controller [229]. However, the satisfaction of the constraints could not be guaranteed any more. Alternatively, Reward Shaping (RS) heuristics might be used in various forms to penalize agents when constraints are violated or let them know when they were saturated [225, 227, 282]. While such methods might accelerate the learning process to some extent, they are indirect, i.e., they only influence the learned policies through the reward function that the agent will learn to optimize over time. Moreover, shaping the reward function simultaneously impacts the learning process of both the actor *and* the critic.

A computationally inexpensive solution for constrained and efficient DRL

In this section, we propose to **constrain the actions of DRL agents by clipping them according to expert-designed state-dependent bounds and subsequently modify the gradient update step of the actor to let agents learn from their mistakes and accelerate their convergence to expected actions**. Importantly, contrary to RS, these interventions only affect the actor, allowing the critic to learn the *true* Q-values.

Notably, our method bypasses the need for complex projection steps and does not require access to a fallback controller or an expert policy. Moreover and critically, the proposed modifications do not impact the computational complexity of the learning algorithm, are straightforward to design and implement, and can be coupled with any actor-critic algorithms.

3.4 Constraining agents and accelerating convergence

Note that, in stark contrast with [277], where the expert knowledge is potentially overridden by the policy, our method enforces the wanted behaviors on agents *at all times*.

In a case study relying on a PCNN simulator, the proposed EAs converge up to six to seven times faster than classical agents and two to three times faster than RS-based ones while retaining good final performance. This hints that the proposed computationally inexpensive modifications can efficiently leverage expert knowledge to accelerate DRL algorithms.

3.4.2 Efficient actor-critic agents

This section details how to encode prior knowledge as simple rules in deterministic actor-critic algorithms¹⁹ to limit the exploration of known suboptimal state-action pairs; first saturating actions taken by the control policy accordingly and then modifying the gradient update of the actor to let agents learn from their mistakes.

State-dependent action saturation

In many cases, prior knowledge allows us to design state-dependent upper and lower bounds $a^{max}(s)$ and $a^{min}(s)$, respectively, on the actions we expect well-performing control policies $\pi_\theta(s)$ to take in a given state s , with

$$a^{low} \leq a^{min}(s) \leq a^{max}(s) \leq a^{high}. \quad (3.13)$$

To limit the exploration of known suboptimal state-action pairs, we modify (3.3) accordingly to force agents to follow the provided prior knowledge:

$$a(s) = \text{clip}(\pi_\theta(s) + \epsilon, a^{min}(s), a^{max}(s)). \quad (3.14)$$

Note that these bounds, stemming from prior knowledge, are also enforced at test time when $\epsilon = 0$ to ensure an agent would *never* heat a room when the temperature is already too hot, for example, neither during the training nor the deployment phase.

Actor gradient modification

The major problem with the clipping operation in (3.14) is its non-differentiability. Worse yet, its subdifferentials go to zero whenever agents are saturated (see (3.17)), making any backward flow of information on the overriding process impossible. As a countermeasure, to let agents learn from their mistakes, we also modify the actor gradient (3.6) to

$$\hat{\nabla}_\theta^{EA} \pi_\theta = -\nabla_\theta \left(\frac{1}{|B|} \sum_{(s,a) \in B} \left[Q_\phi(s, \pi_\theta(s)) - \frac{\lambda}{2} (\pi_\theta(s) - a(s))^2 \right] \right), \quad (3.15)$$

¹⁹While the presented analyses deal with deterministic actor-critic agents for clarity, the results can easily be extended to the stochastic case.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

where λ is a hyperparameter. The second term on the right-hand side of (3.15) penalizes actions chosen by the policy $\pi_\theta(s)$ if they deviate from the constrained action $a(s)$ in (3.14) that was applied to the environment, steering the agent's decisions towards expected actions.²⁰

Note that one could include this penalty in the reward function as

$$r^{RS}(s_t, a_t) = -\max\{L_t - T_t, T_t - U_t, 0\} - \alpha P_t - \frac{\lambda}{2} (\pi_\theta(s) - a(s))^2, \quad (3.16)$$

and then maximize (3.1) instead of directly changing the gradient update step. Remarkably, however, RS also impacts the learning process of the critic in (3.4) when applied to actor-critic frameworks, contrary to our method. We will show the empirical benefits of the proposed modification (3.15) over RS-based penalties in terms of convergence speed in Section 3.4.3.

Implications of the gradient modification

Let $C(s) = \{a \in \mathbb{R} : a^{\min}(s) \leq a \leq a^{\max}(s)\}$ capture the expert-designed rules for any given state s .²¹ Grouping all the parameters θ of the policy in a vector and recalling the definition of the action $a(s)$ applied to the environment in state s from (3.14), we can define its subgradient $\nabla_\theta a(s)$ as

$$a(s) = \begin{cases} a^{\min}(s), & \text{if } \pi(s) < a^{\min}(s), \\ \pi_\theta(s) + \epsilon, & \text{if } \pi_\theta(s) \in C(s), \\ a^{\max}(s), & \text{if } \pi(s) > a^{\max}(s). \end{cases} \implies \nabla_\theta a(s) = \begin{cases} \nabla_\theta \pi_\theta(s), & \text{if } \pi_\theta(s) \in C(s), \\ 0, & \text{else,} \end{cases} \quad (3.17)$$

where $\nabla_\theta \pi_\theta(s)$ is the actor gradient. We can then rewrite the gradient of EAs (3.15) as:

$$\begin{aligned} \hat{\nabla}_\theta^{EA} \pi_\theta &= -\frac{1}{|\mathcal{Z}|} \sum_{(s,a) \in \mathcal{Z}} \left[\nabla_\theta Q_\phi(s, \pi_\theta(s)) - \nabla_\theta \left(\frac{\lambda}{2} (\pi_\theta(s) - a(s))^2 \right) \right] \\ &= -\frac{1}{|\mathcal{Z}|} \sum_{(s,a) \in \mathcal{Z}} \left[\nabla_\theta Q_\phi(s, \pi_\theta(s)) - (\lambda (e_\theta(s)) (\nabla_\theta \pi_\theta(s) - \nabla_\theta a(s))) \right], \end{aligned}$$

where we introduce the error term $e_\theta(s) = \pi_\theta(s) - a(s)$. We hence get the following modified actor gradient, where we omit $(s, a) \in \mathcal{Z}$ for clarity:

$$\hat{\nabla}_\theta^{EA} \pi_\theta = \begin{cases} -\frac{1}{|\mathcal{Z}|} \sum_{\mathcal{Z}} \left[\nabla_\theta Q_\phi(s, \pi_\theta(s)) \right], & \text{if } \pi_\theta(s) \in C(s), \\ -\frac{1}{|\mathcal{Z}|} \sum_{\mathcal{Z}} \left[\nabla_\theta Q_\phi(s, \pi_\theta(s)) - \lambda e_\theta(s) \nabla_\theta \pi_\theta(s) \right], & \text{else.} \end{cases}$$

Remarkably, the additional penalty term in (3.15) hence allows us to solve the issue of the subdifferentials of the clipping operator being zero when actions are saturated, modifying

²⁰This additional penalty was also used in [245] to improve the robustness of differentiable layer-based RL for state-constrained problems.

²¹Without loss of generality, we assume that $a \in \mathbb{R}$ in this analysis for clarity. This assumption can easily be lifted for multi-dimensional problems.

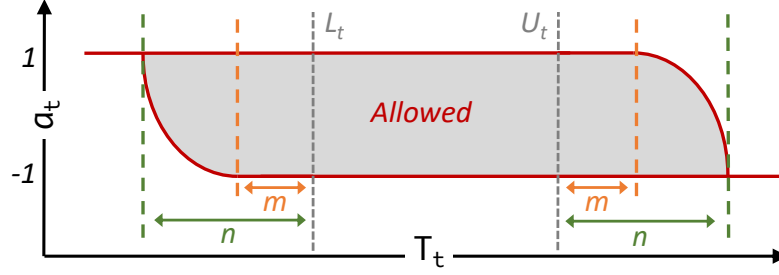


Figure 3.6: Representation of the action bounds used in this work.

the gradients *only when the constraints are not met*. Indeed, as long as the action chosen by the agent respects the constraints provided by the expert, the classical gradient (3.6) is used. On the other hand, as soon as the constraints are not met, the gradient is modified in the direction $e_\theta(s)$. This accelerates the convergence of $\pi_\theta(s)$ to $C(s)$ despite the subdifferential of the clipped action being zero, confirming the graphical intuition from [245, Fig. 2]. This allows EAs to learn from their mistakes and — we hypothesize — helps them rapidly converge to meaningful and well-performing policies.

Design of the saturation rules

In the context of room temperature control, we intuitively know that an optimal policy should gradually stop heating when the temperature reaches the upper comfort bound and slowly start heating as soon as the lower bound is not met (and vice versa for cooling), typically to avoid criticism from the occupants. To encode these simple rules, we design state-dependent action bounds as follows:

$$a^{\min}(s_t) = \text{clip}\left(\frac{(L_t - m) - T_t}{n - m}, 0, 1\right)^2 * 2 - 1 \quad (3.18)$$

$$a^{\max}(s_t) = 1 - 2 * \text{clip}\left(\frac{T_t - (U_t + m)}{n - m}, 0, 1\right)^2, \quad (3.19)$$

with $n \geq m \geq 0$ representing design parameters to leave more or less freedom to the agents. In words, we start constraining the action of the agents as soon as the temperature deviates from the comfort bounds for more than m degrees. We then quadratically increase the constraint until n degrees have been reached, where the agent is forced to use the maximum or minimum power, as pictured in Fig. 3.6. As can be seen, $a^{\min}(s_t) > -1$ only when the temperature is below the lower comfort bound, and $a^{\max}(s_t) < 1$ only when it exceeds the upper one. This means EAs are not constrained and can freely explore the state-action space to minimize energy consumption as long as the comfort of the occupants is satisfied.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

Agent	Reward	Agent	Reward	Agent	Reward
Classical 1	-2.64	Classical 2	-2.75		
RS 0 / 1	-2.58	EA 0 / 1	-2.85	EA 0.5 / 1	-2.83
RS 0 / 0.5	-2.44	EA 0 / 0.5	-2.58	EA 0.25 / 0.5	-2.74
RS 0 / 0.25	-2.37	EA 0 / 0.25	-2.51	EA 0.2 / 0.25	-2.62
RS 0 / 0.1	-3.69	EA 0 / 0.1	-2.46	EA 0.075 / 0.1	-2.46

Table 3.3: Best reward obtained by each agent on the validation set of 50 trajectories over the first 500 epochs.

3.4.3 Performance analysis of Efficient Agents

To investigate the influence of m and n on the proposed scheme, which measure how much prior knowledge is transmitted to DRL agents, we train different EAs (EA m / n). For comparison purposes, we also train agents with the classical actor gradient (3.6) but the reward (3.16) as another computationally inexpensive means to incorporate prior knowledge in DRL agents (RS m / n). Finally, we benchmark these modifications against two classical DRL agents using the actor gradient (3.6) and reward (3.8) with different random seeds (Classical 1 and 2).²²

All the agents were again trained on up to three-day-long episodes randomly sampled from three years of data. To better compare their convergence rapidity, we increase the evaluation frequency compared to Section 3.3: we assess their performance after each 96 steps of 15 min instead of 5'000, i.e., one day's worth of data, hereafter also referred to as an *epoch*. As before, all the agents are evaluated on a fixed testing set of 50 unseen three-day-long trajectories after each epoch to monitor their progress during the first 500 epochs. Throughout this section, we manually set $\lambda = 100$ for EAs to ensure the constraints are enforced as fast as possible and $\lambda = 10$ for RSs since higher penalties led to learning instability. While we empirically observed a more robust performance of EAs with various choices of λ compared to RSs, a complete sensitivity analysis of this hyperparameter is left for future work.

Final performance

The best reward obtained by all the trained agents on the 50 validation sequences over the first 500 epochs can be found in Table 3.3. The corresponding trade-offs between energy consumption and comfort violations over the entire validation set are plotted in Figure 3.7, where the gray markers were reported from Figure 3.4 for reference.

These results illustrate how tighter parameters m and n , i.e., higher levels of prior knowledge, allow EAs (colored crosses) and RSs (colored triangles) to converge to better solutions in this limited training regime. The only exception is RS 0 / 0.1: it did not converge (see Table 3.3)

²²The code and data used to generate the results are available on <https://gitlab.nccr-automation.ch/loris.dinatale/efficient-drl>.

3.4 Constraining agents and accelerating convergence

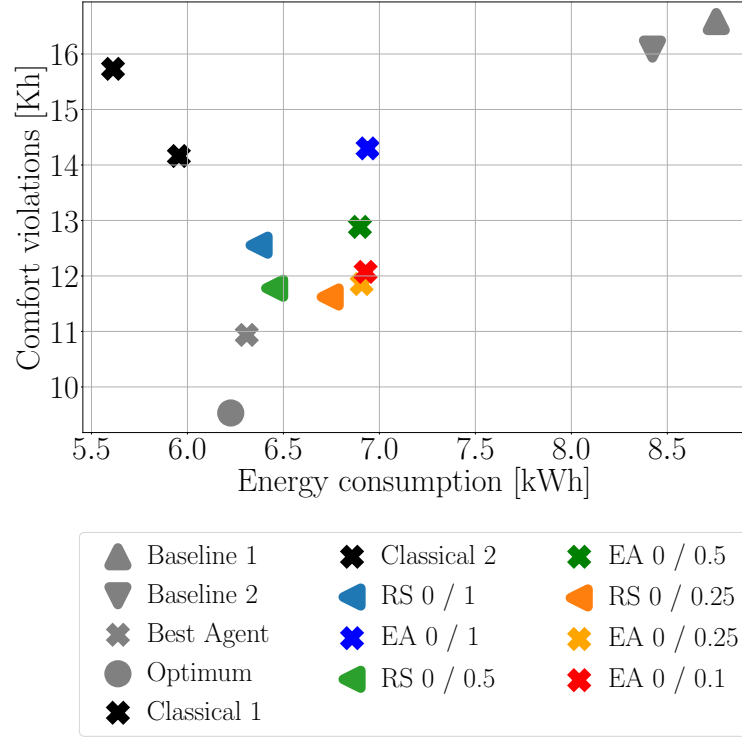


Figure 3.7: Best trade-off between average energy consumption and comfort violations obtained by each agent on almost 2'000 validation trajectories over 500 epochs. For DRL agents, it corresponds to the best rewards obtained on the 50 validation sequences reported in Table 3.3. For comparison purposes, the performance of the two industrial baselines and an agent trained for 125'000 epochs (Best Agent) and the optimal trade-off achievable (Optimum) are reported in gray from Figure 3.4.

and is hence not plotted in Figure 3.7. In particular, tighter constraints allow EAs to reduce the amount of comfort violations without significantly increasing energy consumption. On the other hand, classical DRL agents (black crosses) usually use less energy at the cost of additional comfort violations in this early training phase before converging to near-optimal solutions after longer training times, as detailed in Section 3.3.

Visualization of the impact of prior knowledge

To intuitively understand the effect of action saturation, we can visualize its impact on some EAs in Figure 3.8. The behavior of all agents is plotted *before* training on the left, and *after* on the right, for the same three days during the heating season in March. For completeness, Table 3.4 reports the aggregated metrics of each agent on the right plot. Focusing on the left plot, we see the untrained classical DRL agent in black letting the temperature diverge to

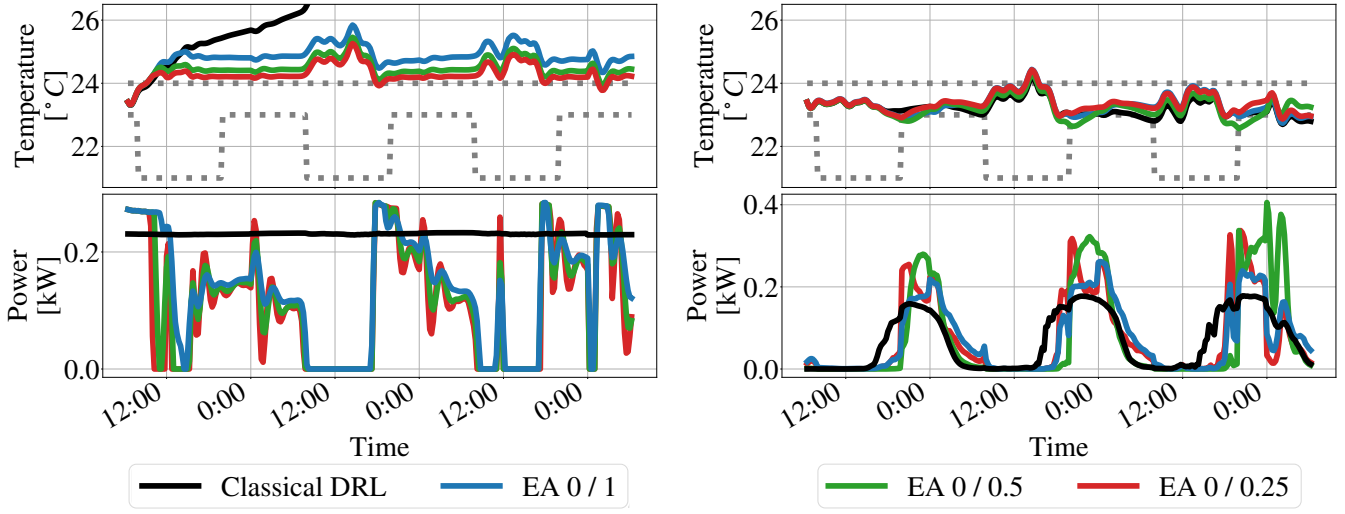


Figure 3.8: Behavior of a classical DRL agent and EAs with various m and n parameters minimizing the heating power consumption (bottom) while maintaining the temperature in the gray dotted predefined bounds (top) over three days in March. **Left:** Performance *before* training, where EAs are saturated once they exceed the bounds by n degrees. **Right:** Performance *after* training, showing how all agents converged to similar solutions, confirming the results in Table 3.3.

Agent m / n	Classical 1 -	EA (ours)		
		0 / 1	0 / 0.5	0 / 0.25
Reward	-0.68	-0.69	-0.89	-0.60
Comfort violations [Kh]	1.28	1.18	2.23	0.65
Energy consumption [kWh]	5.03	5.38	5.54	5.46

Table 3.4: Reward, sum of comfort violations, and aggregated energy consumption of each agent over the three days depicted on the right of Figure 3.8.

an uncomfortably high range (out of the bounds of the plot) as it starts exploring the state space using roughly constant heating power. On the other hand, all the EAs are forced to stop heating once they are n degrees out of bounds. Consequently, even before training, such agents will not overheat the room and keep it at acceptable temperatures for the occupants, corresponding to what we expect from good control policies. However, note that EAs can present control input oscillations due to the impact of external disturbances, mainly the solar gains around noon, triggering the saturation mechanism on and off.

On the right plot, after training, one can observe that all EAs generally make comparable decisions — still being sometimes saturated, which ensures compliance with prior expert knowledge —, which leads to similar temperature patterns. On the other hand, the classical agent presents a slightly different behavior, with smoother control profiles. Interestingly, this agent is the only one heating in the early afternoon; EAs wait until the end of the afternoon

to heat the room with higher power and meet the comfort bound tightening at 20 h instead. This allows the classical agent to use less energy than EAs over these three days but can incur additional comfort violations (see Table 3.4), as expected from Figure 3.7.

Data efficiency of the proposed gradient modification

A comparison of the convergence speed of various agents over the first 300 epochs is plotted in Figure 3.9, where the vertical lines and annotations illustrate the number of days required to attain performance on par with the two baselines. In general, EAs reach this threshold significantly earlier than classical DRL agents, in as little as 29 days instead of roughly 200, an improvement of almost an order of magnitude. In particular, the smaller n is chosen (from left to right in Figure 3.9), the faster the convergence of the EAs in green and blue. Intuitively, this makes sense, as tighter constraints enforce more prior knowledge on the EAs, allowing them to find meaningful solutions more quickly, without losing time exploring suboptimal state-action pairs. On the other hand, the influence of m is less marked, with $m \neq 0$ (blue) and $m = 0$ (green) leading to very similar convergence patterns in the bottom row of plots in Figure 3.9. Nonetheless, as expected, we still observe that smaller values of m tend to incur faster convergence since it further restrains the agents' freedom according to prior knowledge.

Remarkably, RS does not seem to drastically speed up training in this case study (red). While RS 0 / 0.25 does converge twice as fast as the classical DRL agents, we can also observe that RS 0 / 0.1 did not converge at all, hinting at the fragility of this scheme in general. Even when they find meaningful solutions, RSs remain two to three times slower than their EA counterparts, hinting at the superiority of the proposed gradient modification for accelerated convergence. On the other hand, RS seems to lead to more consistent performance than classical agents and EAs after a few hundred epochs, which is confirmed by their impressive final performance in Figure 3.7 and Table 3.3. Consequently, merging both approaches could yield impressive results in practice, initially using (3.15) to improve the convergence speed and gradually diminishing its impact to let the influence of (3.16) increase and stabilize the final performance.

3.4.4 Towards agents that can be deployed from scratch in physical buildings

Accelerating convergence (R7). Overall, these results support our claim that, as long as the rules provided to the agents are well-defined and correspond to expected behaviors, the modifications proposed in Section 3.4.2 can greatly accelerate the convergence of DRL agents, helping them to satisfy R7. Interestingly and as expected, incorporating more specific expert knowledge in EAs — through smaller m and n , hence enforcing tighter constraints — further improves their learning speed. While they remained two to three times slower than EAs in our case study, RSs benefited from a similar relationship between the amount of prior knowledge used and the subsequent convergence speed.

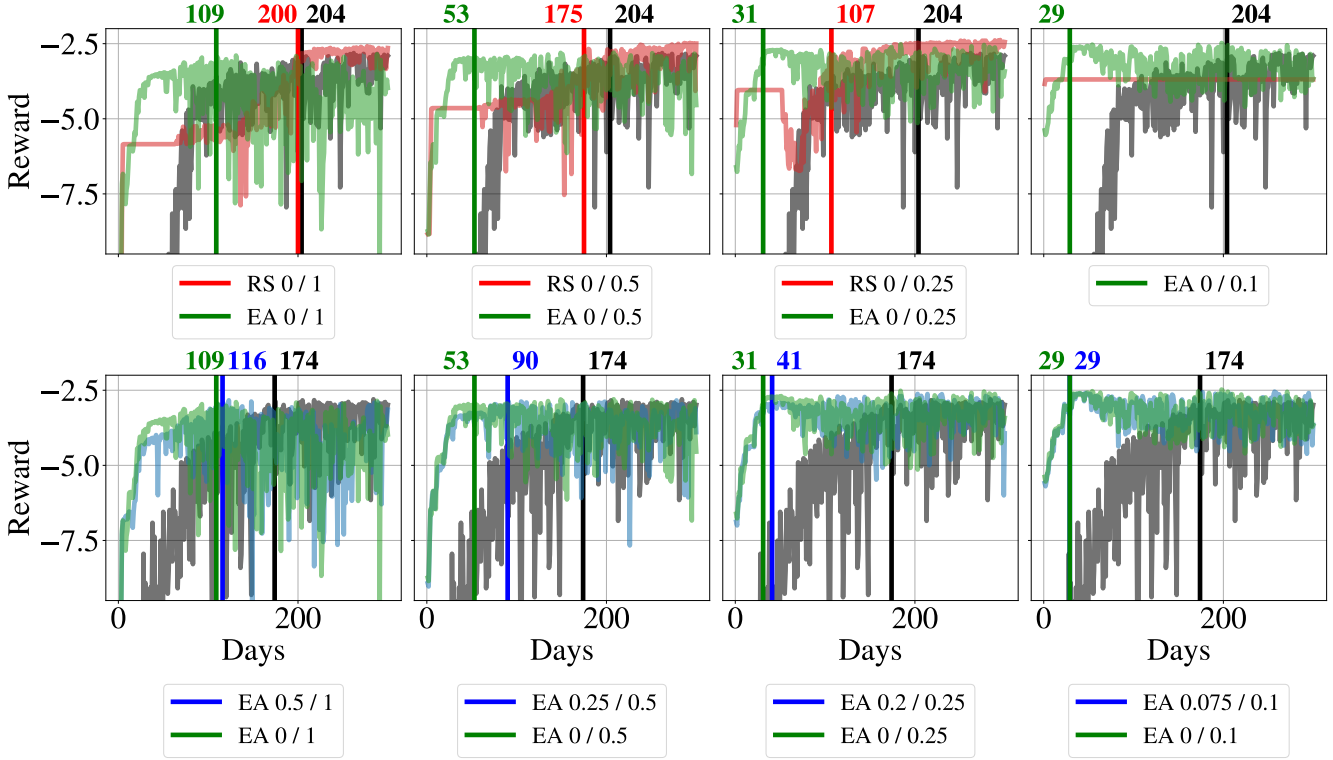


Figure 3.9: Convergence speed of various EAs with different m and n parameters in green and blue, compared to agents using RS in red and two classical agents in black (one in the top plots, one in the bottom ones). The vertical lines and annotations specify the number of days of data required to obtain a reward of -2.95 for each agent, corresponding to the average performance of the two baselines.

Interestingly, the proposed actor gradient update modification provided the desired speedup for various choices of m and n , contrary to RS, hinting at its robustness. Despite converging 6–7 faster than vanilla agents, however, EAs still required 30 days to achieve performance on par with the baselines in this case study. Furthermore, these findings are not guaranteed to transfer to other applications or more complex case studies. This could be a hurdle for real-world deployments as occupants might expect better performance than baseline controllers after a few days, irrespective of building characteristics.

Near-optimality (R1). Critically, these convergence speed improvements do not significantly impact the quality of the final solution compared to classical unconstrained agents, as shown in Figure 3.7. This hints that the proposed modifications do not significantly hinder the ability of EAs to satisfy R1, and we postulate that the analyses proposed in Section 3.3.2 for vanilla DRL policies would also hold for EAs in the long-term.

Satisfying constraints (R3). Notably, the action saturation in (3.14) enforces *soft guarantees*

to respect the comfort of the occupants, creating DRL agents able to meet R3. Indeed, while we cannot ensure that the indoor temperature will *never* leave the predefined bounds, EAs will always react as expected whenever violations occur, turning the heating on if the temperature inside gets too cold, for example. Such a response would probably avoid complaints from the occupants in real-world deployments, especially if EAs quickly learn to avoid these situations.

Improving robustness (R2). Finally, these soft guarantees also pave the way towards DRL agents simultaneously meeting R2 to some extent. Indeed, they ensure the inside temperature will remain comfortable for the occupants at all times, under any weather conditions, for example. However, ensuring near-optimal performance under any external condition remains a challenge in general: while we can ensure NN-based control policies do not behave catastrophically when subjected to disturbances they have not been trained to handle with such constraints, we cannot expect them to make optimal decisions in these cases.

3.5 Conclusion and outlook

In this chapter, we started by discussing seven requirements of an ideal building controller, arguing in favor of online model-free DRL algorithms, which have the potential to bypass the need for building and disturbance models and the related pitfalls in terms of scalability and transferability (R5–R6). However, the final performance of such controllers, their ability to handle disturbances and constraints, and their convergence speed (R1–R3, R7) remain open questions. Consequently, we then provided in-depth analyses of the potential of DRL policies to meet these requirements in a zone temperature case study. First, we reported evidence pointing towards their capability to achieve near-optimal performance in different settings (R1) in Section 3.3. Then, Section 3.4 proposed computationally inexpensive modifications of actor-critic algorithms to find well-performing policies in a few weeks of data (addressing R7 to some extent) while providing adequate comfort under different internal and external conditions (towards the satisfaction of R2–R3).

3.5.1 Limitations of our experiments

Naturally, the main limitation of our investigations comes from the single low-complexity framework the agents were evaluated in throughout this chapter, namely the thermal control of a single bedroom in UMAR in simulation. Furthermore, while PCNNs grasp nonlinear dynamics through their black-box module, they remain input-affine and might not capture the full complexity of building thermal dynamics to evaluate DRL agents adequately. However, experimental demonstrations in UMAR, such as the one depicted in Figure 3.5, revealed similar behaviors to what could be observed in simulations, hinting at the efficacy of PCNNs to produce accurate dynamics. Nonetheless, to validate our findings, one would have to expand all the analyses to different buildings and scale to more complex problems, incorporating interactions with batteries and PV panels, for example. In other words, this would **assess the ability of the proposed methods to handle R5–R6 simultaneously to R1–R3 and R7**.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

Apart from these considerations, it is noteworthy that we did not strive to achieve the best performance in the proposed case study, instead focusing on comparisons and analyses of different agents. It would probably be possible to improve the quality of each solution through a thorough hyperparameter selection procedure, for example. In practice, one should also investigate the impact of the state-space \mathcal{S} ; some variables might not be required, while other ones, typically weather forecasts, could be very valuable to improve performance [215]. Similarly, more informative reward functions might help DRL agents converge faster to expected behaviors. For example, one could reward them according to the relative performance improvement they achieve compared to a baseline instead of their absolute performance, similarly to [251].

Note that although we focused on building applications throughout this chapter, it would be of interest to carry out similar investigations in other fields. In particular, one could analyze whether DRL policies generally achieve near-optimality and whether adapting the modifications proposed in Section 3.4 to other domains can also accelerate learning. However, the latter would require principled solutions to transfer generic prior knowledge into rules enforceable on the agents' actions. Moreover, its applicability would be constrained to soft-constrained systems, as the proposed EA implementations cannot ensure the fulfillment of state constraints.

3.5.2 Potential pathways towards the satisfaction of R1–R7

Overall, our investigations hint that computationally inexpensive interventions on model-free DRL agents can allow them to meet almost all the desired requirements R1–R7. Nonetheless, additional efforts are still required to achieve robustness and reasonable convergence times in practice (R2 and R7) and assess whether our conclusions hold for larger-scale case studies and different buildings (R5–R6). Let us now mention possible pathways toward DRL agents meeting these requirements, opening the discussion on potentially interesting future works. A combination of progress in several fields mentioned below will probably be required to design controllers that can be deployed from scratch in any building.

Accelerating convergence. First, although we managed to accelerate the convergence of DRL agents by a factor of six to seven in Section 3.4, EAs still required a month of data to achieve performance on par with the two baselines in a low-complexity zone temperature control case study. To make matters worse, this training time is bound to increase when scaling DRL policies to control whole buildings. It could thus lead to high energy bills and occupant discomfort — even if soft guarantees are enforced — during a relatively long initial learning phase. This calls for measures leading to even faster online convergence to satisfy R7 and achieve widespread acceptance.

To that end, it would be interesting to investigate the impact of data selection in the training process of DRL agents. For example, convergence to a meaningful solution might be accelerated by considering consecutive days of data in the training phase instead of randomly

sampled ones. Remarkably, this corresponds to the situation a DRL agent deployed and trained from scratch on a physical building would face. On one hand, well-performing policies might be found faster since it would avoid training the NN policy to react to widely different inputs. On the other hand, it could lead to issues once the external conditions change severely — from summer to winter, for example — if the policy converges to a local minimum too early during training.

As an additional and potentially concurrent approach to improve the convergence speed of DRL algorithms, one could explore the systematic decomposition of building control problems into simpler and more complex sub-tasks. This would allow practitioners to use standard rule-based controllers in the former cases and only leverage DRL for hard-to-specify tasks, for example, which might significantly accelerate learning, as in [213]. Note that this is closely related to the proposed EAs in Section 3.4, where we force agents to follow expert knowledge when the temperature is too far from the comfort bounds; one could instead let a baseline take over in that case, solely training the DRL agent to optimize the control inputs when the temperature is in the comfort range and the optimal behavior unknown. Similarly, one could train several agents, each handling different situations (for example, one per season), to simplify their learning task. However, while each agent would converge faster, whether the total training time, energy consumption, and occupant discomfort would decrease is an interesting question.

Improving EAs. To achieve strong final performance (R1) despite constraining the agents as proposed in Section 3.4 to satisfy R2–R3 and R7, it might be interesting to investigate annealing strategies on λ or leverage primal-dual optimization tools on (3.15). The latter could indeed adaptively tune the influence of the additional penalty in the actor gradient and let agents learn more expressive policies after the initial exploration phase [283]. Additionally, given the final performance of RS-based agents, it might be worth analyzing how to merge their modified reward function (3.16) with EAs to simultaneously achieve near-optimal final performance and rapid convergence towards it.

Physics-inspired DRL. Even if imposing constraints on the agents’ actions to enforce soft guarantees on the indoor conditions does ensure DRL control policies cannot act inadequately, they might still perform significantly sub-optimally under new disturbances and not meet R2. While pre-training with extensive simulations is the best and most widely used countermeasure to date, it is limited by its reliance on accurate simulators. Instead, knowledge-informed or Physics-inspired DRL (PiDRL) provides an interesting alternative pathway to alleviate the brittleness of NN control policies. Similarly to what was done in Chapter 2 to guarantee PCNNs respect the underlying laws of thermodynamics, one could indeed ensure DRL agents understand the main factors driving thermal building dynamics to steer their policies towards expected behaviors.

While the system’s physics is often leveraged to characterize unsafe actions [223, 224, 278] or design more informative reward functions [225], it might be leveraged to capture more

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

abstract concepts. For example, in the case of building control, it could be of interest to design control policies explicitly encoding expected dependences on indoor temperatures, presence of occupants, or any other impactful factor. Inspired by the architecture of PCNNs in Chapter 2, one could similarly design a modular control policy enforcing agents to react to some conditions according to expert intuition through a knowledge-infused module while letting an NN learn the best responses to disturbances harder to optimize against in parallel. This could lead to control policies guaranteed to react as expected in critical cases, as a further step towards the general satisfaction of R2. Note that the modifications proposed in Section 3.4 are one possible way to enforce DRL agents to react to out-of-bounds temperatures as expected and might hence provide a basis for more generic PiDRL building control policies.

Transfer Learning. Since it would allow one to warm-start the training of DRL agents from a well-performing initial control policy, TL is another potential solution to increase their data efficiency and go toward the satisfaction of R6–R7. However, which part of control policies can be transferred without significant performance drop remains a challenging field of research [156]. For example, when a DRL agent learned in a single zone was subsequently deployed to control the whole building, it showed overfitting, and manual interventions were required to avoid catastrophic behaviors in [284]. Nonetheless, if significant advances were made in TL, it could further accelerate the online convergence speed of building DRL control policies — and potentially significantly.

Going beyond traditional TL, one could investigate the potential of designing a *general* building control policy, i.e., pre-training one or a few large-scale NNs to optimize different building architectures, appliances, comfort parameters, weather conditions, and so on, *a priori*, similarly to [42]. This would call for a systematic simulation procedure and an extensive learning phase but could then be leveraged to potentially accelerate convergence online (R7) in any new building (R6) and help ensure DRL agents behave adequately under all the simulated disturbances (R2).

Multi-agent Reinforcement Learning. As advocated by [14, 31, 206], for example, significant progress still needs to be made on multi-agent RL algorithms to optimize interconnected complex systems like large-scale buildings or neighborhoods. While single DRL agents might have interesting scalability properties, they indeed cannot be expected to control whole districts, and cooperation among several agents acting on a smaller scale will be required to tackle R5 to its full extent. Even at the building level, letting a single agent control multiple zones might not be optimal, as observed in [285], where learning one distinct control policy for each zone, while increasing the computational burden, led to better final performance, for example.

The field of multi-agent RL is however still nascent, with many open problems linked to non-stationarity, communication (what to communicate to which agent(s) and when), coordination (how to reach consensus), credit assignment (if the agents share a common goal, how to recover which action(s) of which agent(s) led to good rewards), scalability, and partial

observability (agents do not have access to the full state of the environment or other agents' internal information) [286, 287]. Note that given the sensitivity of building operation data, letting several agents communicate would furthermore incur privacy issues, advocating for federated learning-inspired techniques, such as in [288].

Occupant-centric control. Given the importance of the occupants — stemming from their preferences and behavior [12] —, more efforts should be directed towards occupant-centered solutions. While most studies still only consider maintaining the inside temperature in a comfortable range [206], accommodating the thermal comfort of the occupants to some extent, it is nowadays well-known that the concept of “comfort” goes beyond these simple considerations [144].

However, how to represent comfort preferences more accurately is an open field of research and a challenging problem. This is due to the lack of data — and its quality whenever it is available — and the significant related privacy issues, among other factors [289]. Remarkably, model-free DRL agents could incorporate personalized comfort preferences [10]. For example, it would be possible to directly learn control policies from the feedback of the occupants [290], opening many avenues for occupant-centered yet energy-efficient solutions.

Interpretable controllers. We hypothesized that an ideal controller would satisfy R1–R7 in this chapter, implying that it should perform near-optimally under any circumstance (R1–R2). With these two requirements, we implicitly assumed that such a controller would receive the people's trust since it would always make meaningful decisions. Thus, we did not list *interpretability* amongst the requirements. However, in practice, especially in the early development phase, *explainability* of a controller's decisions might be crucial to foster its acceptance.

To bypass the limited expressiveness of RBC tuning methods like [161, 167], which are interpretable but inherently upper-bounded by the structure of the underlying RBC, one could turn to policy distillation techniques instead [291]. The idea here is to first fit an NN policy to the control problem, typically through DRL, and then reduce it to an interpretable tree or set of rules with minimal performance loss. In other words, a low-complexity interpretable controller replaces the NN policy, mimicking its behavior. Such efforts to make DRL algorithms more transparent are inscribed in the scope of explainable RL, a field which has been growing in popularity in the last years [292].

3.5.3 A practical perspective

To conclude this chapter, we want to emphasize here that we only investigated the *theoretical* promises of model-free DRL agents in-depth as we argue they exhibit strong potential to fulfill the pre-defined seven requirements of ideal building controllers. However, in practical applications, the best choice of method often hinges on the “engineering budget” and the availability of models or data.

Chapter 3. Prospects and hurdles of Deep Reinforcement Learning for building control

Naturally, if an accurate building model exists, it should be leveraged in model-based DRL methods or to pre-train model-free DRL agents to accelerate their convergence online and train them to handle different disturbances and constraints (R2–R3, R7). If disturbance forecasts are also easily accessible,²³ MPC-based applications would probably achieve impressive performance — provided the optimization problem can be solved quickly enough with the hardware on-site at each time step.

Otherwise, if only data is available — in quantity and quality —, one could use it to build an accurate simulator and recover a building model to be leveraged as discussed above. Alternatively, one could rely on IL techniques to warm-start DRL control policies (R7) or follow DPC-based approaches, for example. Provided disturbance forecasts are accessible, the latter could attain compelling performance, especially if the model could be updated online without compromising the controller's quality.

Finally, online methods would have to be applied in cases where neither a model nor an extensive data set is available at deployment time. Model-based approaches might achieve strong performance but require learning and maintaining an accurate building model online. Alternatively, model-free DRL algorithms, especially with the proposed modifications in Section 3.4, could directly learn how to optimize building operations while treating the system as a black box. However, they might still require several months to converge to a well-performing policy. In sum, neither of these approaches can provide performance guarantees for practical applications yet, and the best solution to date might be to collect data for a few days or weeks and then fall back to one of the options discussed above.

²³In the case of building control, the main disturbances usually stem from weather conditions and occupant behavior. While weather forecasts are generally accessible, predicting the latter is more challenging.

4 Leveraging automatic differentiation for system identification

While Neural Networks recently gained massive attention thanks to their ability to solve very complex modeling tasks [59–64], they are generally system-agnostic (Section 2.1). Even when known physics is enforced upon them, as discussed with PCNNs throughout Chapter 2, for example, they remain *uninterpretable*, and people often prefer more structured and explainable models in practice [293]. Indeed, it might not be desirable to use black-box NNs in applications where some properties of the model to identify are known — like stability or physical properties. Additionally, NNs might perform sub-optimally in such cases, where learning a model with the expected structure can lead to better performance [294]. Contrary to NNs, traditional System Identification (SI) approaches, which matured decades ago [47], can enforce desired system properties — but they struggle to scale in general.

In this chapter, we start from the fact that the availability of state-of-the-art open-source libraries like PyTorch [140] and TensorFlow [295] has been key to the success of NNs in fitting million of parameters on large-scale problems. In particular, Automatic Differentiation (AD), at the core of the backpropagation algorithm [296], the backbone of NN training, nowadays benefits from extremely efficient implementations. Consequently, we propose to leverage these recently developed tools to help scale traditional SI methods and identify structured models from data.

The main output of these investigations is the **SIMBa open-source toolbox (System Identification Methods leveraging Backpropagation)** on <https://github.com/Cemempamoi/simba>. Relying on novel linear-matrix-inequality-based free parameterizations of Schur matrices, SIMBa can enforce desired system properties without jeopardizing the stability of linear state-space models. Extensive numerical simulations show it consistently outperforms traditional methods — and sometimes significantly — when identifying different systems with and without state measurements, from simulated or real-world data, and while enforcing various properties. Finally, we also propose one extension of this framework to identify nonlinear systems following the laws of thermodynamics in Section 4.6.

Chapter 4. Leveraging automatic differentiation for system identification

Notations used throughout the chapter

Let \mathbb{I}_q and $\mathbb{1}_{q \times q}$ be the identity and all-one matrix of dimension q , respectively. Given a matrix $H \in \mathbb{R}^{2q \times 2q}$, we define its block components $H_{11}, H_{12}, H_{21}, H_{22} \in \mathbb{R}^{q \times q}$ as

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} := H.$$

For a symmetric matrix $F \in \mathbb{R}^{q \times q}$, $F \succ 0$ means it is positive definite. For a matrix $K \in \mathbb{R}^{q \times q}$, $\lambda_{\min}(K)$ and $\lambda_{\max}(K)$ refer to its minimum and maximum eigenvalue, respectively, and

$$|\lambda(K)|_{\max} := \max_{i=1, \dots, n} |\lambda_i(K)|$$

to its maximum absolute eigenvalue. $\sigma := \mathbb{R} \rightarrow]0, 1[$ corresponds to the sigmoid function

$$\sigma(r) = \frac{1}{1 + e^{-r}}$$

and σ^{-1} to its inverse. $\|\cdot\|_p$ represents the p -norm of a vector. Finally, A matrix J is *skew-symmetric* if $J = -J^\top$. The *Poisson bracket* of $Z, G \in C^\infty(\mathbb{R}^n)$ with respect to a skew-symmetric matrix J is defined as

$$\{Z, G\}_J = \frac{\partial Z^\top(x)}{\partial x} J \frac{\partial G(x)}{\partial x}.$$

4.1 Towards structured stable linear system identification

Given their effectiveness at grasping complex nonlinear patterns from data, NNs have recently been used for nonlinear system identification, where traditional SI methods struggle to compete [297–299]. NNs can be leveraged to create deep state-space models [294], deep subspace encoders [300], or deep autoencoders [301], for example. While applying NNs to identify nonlinear systems can perform well, it might underperform for linear systems, for example, where methods assuming model linearity can achieve better accuracy [294].

Although nonlinear SI has attracted significant attention in recent years, the identification of Linear Time Invariant (LTI) models is, however, still of paramount importance to many applications. Indeed, linear models come with extensive theoretical properties [302] and lead to convex optimization problems when combined with convex cost functions in a Model Predictive Controller [125], for example. Moreover, to date, numerous industrial applications still rely on the availability of linear models to conduct simulations, perform perturbation analysis, or design robust controllers following classical model-based techniques, such as \mathcal{H}_2 , \mathcal{H}_∞ , and μ -synthesis [303].

In Sections 4.2–4.5, we show how one can leverage ML tools — backpropagation and unconstrained Gradient Descent (GD) — for the identification of stable linear models, presenting a novel toolbox of System Identification Methods leveraging Backpropagation (**SIMBa**). Our

research is related to the efforts in [146, 183, 304–306], where backpropagation was also used to identify LTI state-space models but only as part of specific frameworks and without considering stability constraints.

4.1.1 Problem setting

Sections 4.2–4.5 are concerned with the identification of discrete-time linear time-invariant state-space models of the form

$$x_{k+1} = Ax_k + Bu_k \quad (4.1a)$$

$$y_k = Cx_k + Du_k, \quad (4.1b)$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ are the states, inputs, and outputs, respectively. The objective is to identify $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, and $D \in \mathbb{R}^{p \times m}$ from data.

Throughout this chapter, we assume access to a data set $\mathcal{D}^{i/o} = \{(u(0), y(0)), \dots, (u(l_s), y(l_s))\}_{s=1}^N$ of N input-output measurement trajectories s of length l_s . Note that, for some applications, one might have direct access to state measurements, in which case (4.1b) is omitted and only A and B need to be identified from a data set of input-state measurements $\mathcal{D}^{i/s} = \{(u(0), x(0)), \dots, (u(l_s), x(l_s))\}_{s=1}^N$. In our experiments, we split the data into a training, a validation, and a test set of trajectories \mathcal{D}_{train} , \mathcal{D}_{val} , and \mathcal{D}_{test} , respectively, as often done in ML pipelines [307].

When we want to enforce the asymptotic stability of (4.1), we need to ensure A is Schur, i.e., all its eigenvalues $\lambda_i(A)$ satisfy $|\lambda_i(A)| < 1$, $\forall i = 1, \dots, n$ [308]. Finally, to discuss *sparsity patterns* of various matrices, we will use *binary masks* $\mathcal{M} \in \{0, 1\}^{q \times s}$ and denote with $M := \mathcal{M} \odot \bar{M}$ sparse matrices $M \in \mathbb{R}^{q \times s}$, where \mathcal{M} is the corresponding sparsity pattern, \bar{M} can be any matrix of appropriate dimensions, and \odot denotes the Hadamard product between two matrices.

4.1.2 Subspace identification for linear systems

State-of-the-art implementations of linear state-space SI often rely on subspace identification [54], such as the acclaimed MATLAB system identification toolbox [309] or the SIPPY Python package [50]. Both of them provide the three *traditional* Subspace Identification Methods (SIMs), namely N4SID [310], MOESP [311] and CVA [312]. Remarkably, these three methods were later unified under a single theory in [313], proving they rely on similar concepts.

In addition to these traditional methods, SIPPY also proposes an implementation of *PARsimonious* SIMs (PARSIMs), namely PARSIM-S [314], PARSIM-P [315], and PARSIM-K [316], which enforce causal models by removing non-causal terms. While the former two methods do not work with closed-loop data since they assume no correlation between the output noise and the input, PARSIM-K was specifically designed to alleviate this assumption.

4.1.3 Enforcing stability

In practice, when the system is known to be stable, one usually requires the identified model to be stable as well [48]. In this case, *post-hoc* corrections can be applied to the state-space matrices identified by SIMs to guarantee stability [49]. However, this is impossible for PARSIMs and might cause severe performance drops on traditional SIMs [50]. Apart from such *post-hoc* modifications, one can also modify the Least Squares (LS) estimation at the heart of many identification procedures to ensure stability. This can be achieved by either introducing custom weighting factors [317] or rewriting it as a constrained optimization problem [318, 319], for example.

Alternatively, one can leverage parametrizations of Schur matrices, such as the ones proposed in [320, 321], and then use projected gradients to approximate the LS solution while ensuring the resulting model remains stable at each step, for example [51]. A similar idea was utilized in [322], where the Perron-Frobenius theorem was leveraged to bound the eigenvalues of A and hence ensure the system remains stable *at all times*, even during the learning phase.

Finally, instead of directly constraining the state-space matrices, one can simultaneously learn a model and a corresponding Lyapunov function for it, typically NN-based, thereby ensuring its stability by design [323]. This approach presents the advantage of naturally extending to nonlinear SI, contrary to all the others, but comes with a significant computational burden. Note that while SIMBa does not explicitly learn a Lyapunov function, it implicitly defines one to guarantee stability (see Section 4.2). However, instead of learning it with an NN, we leverage Linear Matrix Inequalities (LMIs) to parametrize Schur matrices, inspired from [324].

4.1.4 Prior knowledge integration

In addition to maintaining the stability of the system, it can be beneficial, and sometimes necessary, to convey expert knowledge or desired properties to the identified model in practice. Specifically, there is a growing interest in methods that can incorporate known properties in the state-space matrices to identify — to enforce desired sparsity patterns, for example. This information might indeed be known *a priori*: one may have insights on which states are measured, which inputs impact which states, or which states exchange information, i.e., the topology of a networked system. Such requirements led to the development of SIMs specifically tailored for distributed systems with different topologies, where the state-space matrices are known to have specific sparsity patterns [52, 325–327].

Beyond sparsity patterns, an expert might have prior knowledge about the structure of the system stemming from known physical properties, for example. To ensure the identified model follows the desired dynamics, one typically writes down the corresponding state and output equations manually and then identifies the unknown parameters from data. Such *gray-box* modeling approaches have been successfully applied to building [122], chemical process [328], or robotic system [329] modeling, among others. These considerations were

recently unified in the COSMOS framework, which allows one to identify structured linear systems from input-output data [53]. It is based on rewriting the LS estimate in SIMs as a rank-constrained optimization problem to enforce the matrices to belong to the desired parametrized set. However, COSMOS minimizes the one-step-ahead prediction error and does not guarantee the stability of the identified system.

We note here that knowledge integration also encompasses the careful initialization of all the parameters, which has already been shown to improve performance in the context of nonlinear SI [146]. While this is not needed for classical subspace methods, which rely on deterministic LS solutions [54], it can have a significant impact on the convergence rate and quality of the solutions of gradient-based algorithms like SIMBa [330].

4.1.5 Summarized contributions

To summarize, none of the above methods allowing prior knowledge integration considered the stability of the resulting system. Additionally, they usually rely on one-step-ahead fitting criteria. To maintain state-of-the-art performance without losing stability guarantees when enforcing desired system properties, such as predefined sparsity patterns or known values of state-space matrices, we introduce SIMBa, a structured linear SI toolbox that allows for detailed prior knowledge integration without jeopardizing the stability of the identified model. Leveraging novel free parameterizations of Schur matrices and well-established ML tools for multi-step prediction error minimization, we show how SIMBa can significantly outperform traditional stable SI methods found in the MATLAB SI toolbox [309] through extensive numerical experiments.

SIMBa is open-sourced and system-agnostic: it can seamlessly identify both multi-input-multi-output and multi-input-multi-state data, optimize different multi-steps-ahead performance metrics, deal with large-scale systems, multiple trajectories, and missing data, and comes with smooth GPU-integration. Due to its GD-based backbone, it incurs significant computational burdens, requiring from several minutes to over an hour to train compared to the few seconds needed for conventional methods. However, it consistently — and sometimes significantly — outperforms traditional approaches on a wide variety of problems in terms of accuracy, even while enforcing prior knowledge on the state-space matrices. SIMBa could hence be very beneficial in applications where performance is critical or system properties must be respected.

Altogether, Sections 4.2–4.5 propose a new paradigm for SI of large-scale structured linear systems without losing stability guarantees. While it comes with a large computational burden, SIMBa also presents interesting extension potential, for example, to include tailored nonlinearities, similarly to what is proposed in Section 4.6, or to facilitate stable Koopman-based approaches like [304–306, 331].

4.2 Free parametrizations of Schur matrices

To efficiently leverage PyTorch’s automatic differentiation implementations, which cannot deal with constrained optimization problems, we require A to be Schur *by design*. This will then allow us to run unconstrained Gradient Descent (GD) in the search space without jeopardizing the stability of the identified system. Throughout this section, inspired by [324], we leverage Linear Matrix Inequalities (LMIs) to design matrices that simultaneously guarantee stability and capture various system properties.

Credit assignment. The results of this section stem from a collaboration between Muhammad Zakwan and the author of this thesis in [56, 57]. Although their development was primarily spearheaded by Muhammad Zakwan, they are reported here for completeness since they will be leveraged in Section 4.4, but the proofs are deferred to the appendix.

4.2.1 Dense Schur matrices

Let us first provide one possible free parametrization of Schur matrices with arbitrary structure and bounded eigenvalues.

Proposition 3. *For any $W \in \mathbb{R}^{2n \times 2n}$, $V \in \mathbb{R}^{n \times n}$, $0 < \gamma \leq 1$, and $\epsilon > 0$, let*

$$S := W^\top W + \epsilon \mathbb{I}_{2n}. \quad (4.2)$$

Then

$$A = S_{12} \left[\frac{1}{2} \left(\frac{S_{11}}{\gamma^2} + S_{22} \right) + V - V^\top \right]^{-1} \quad (4.3)$$

is Schur with $|\lambda_i(A)| < \gamma, \forall i = 1, \dots, n$.

Proof. See Appendix B.1. □

Note that γ is a user-defined parameter bounding the eigenvalues of A in a circle of the corresponding radius centered at the origin, potentially enforcing desired system properties on the learned matrix. Importantly, Proposition 3 captures *all* Schur matrices, as detailed in the following corollary.

Corollary 2. *For any given Schur matrix A and $\epsilon > 0$, there exists $W \in \mathbb{R}^{2n \times 2n}$, $V \in \mathbb{R}^{n \times n}$ satisfying (4.3) for S as in (4.2).*

Proof. See Appendix B.2. □

4.2.2 Discretized continuous-time systems

Many linearized real-world systems are continuous-time, i.e., of the form

$$\dot{x} = \bar{A}x + \bar{B}u \quad (4.4a)$$

$$y = Cx + Du. \quad (4.4b)$$

After a forward Euler discretization, (4.4) becomes

$$\begin{aligned} x_{k+1} &= (\mathbb{I}_n + \delta \bar{A}) x_k + \delta \bar{B} u_k \\ y_k &= C x_k + D u_k, \end{aligned}$$

where $\delta > 0$ is the discretization step.¹ In particular, the matrix A we want to identify from discrete-time data samples while ensuring stability now takes the form

$$A := \mathbb{I}_n + \delta \bar{A}. \quad (4.5)$$

In other words, if the data in $\mathcal{D}^{i/o}$ or $\mathcal{D}^{i/s}$ has been collected from a continuous-time system, then A will be *close to identity*. Note that a similar behavior would also be expected from slow-changing systems. The following proposition offers a parametrization of A that takes this desired structure into account.

Proposition 4. *For any $W \in \mathbb{R}^{2n \times 2n}$, $V \in \mathbb{R}^{n \times n}$, and $\epsilon > 0$, let*

$$S := W^\top W + \epsilon \mathbb{I}_{2n}. \quad (4.6)$$

Then

$$A = \mathbb{I}_n - 2(S_{11} + V - V^\top)^{-1} S_{12} S_{22}^{-1} S_{21} \quad (4.7)$$

is a Schur matrix.

Proof. See Appendix B.3. □

Contrary to Proposition 3 and as evident from (4.7), the matrix A will be steered towards the identity matrix here, as desired. If the given data stems from a continuous-time or slow-changing discrete-time system, this might ease SIMBa's learning procedure, and we will leverage it in Section 4.4.6. Importantly, Proposition 4 does not sacrifice any representation power, as stated in the following corollary.

Corollary 3. *For any given Schur matrix A and $\epsilon > 0$, there exists $W \in \mathbb{R}^{2n \times 2n}$, $V \in \mathbb{R}^{n \times n}$ satisfying (4.7) for S as in (4.6).*

Proof. See Appendix B.4. □

¹While different discretization schemes exist, we focus on the forward Euler one herein as it allows us to derive another meaningful parametrization of Schur matrices.

Remark 20. The discretization step δ does not appear in (4.7). While it might seem counter-intuitive at first glance, this is not an issue in practice: changing the discretization step would indeed modify the data collection for $\mathcal{D}^{i/o}$ or $\mathcal{D}^{i/s}$, thereby naturally changing the solution found by SIMBa through GD and hence the form of A . In other words, A implicitly depends on δ , as expected.

4.2.3 Sparse Schur matrices

Let us now assume the sparsity pattern \mathcal{M} of A is given. This might arise in cases where the system to identify is a networked system with a known topology, for example. The following proposition shows how to parameterize such sparse matrices without losing stability guarantees.

Proposition 5. For a given sparsity pattern $\mathcal{M} \in \{0, 1\}^{n \times n}$, any $W \in \mathbb{R}^{2n \times 2n}$, $V \in \mathbb{R}^{n \times n}$, and $\epsilon > 0$, let

$$S := W^\top W + \epsilon \mathbb{I}_{2n} \quad (4.8)$$

and construct the diagonal matrix N with entries

$$N_{ii} := \max \left\{ \sum_{j \neq i} \mathcal{M}_{ij}, \sum_{j \neq i} \mathcal{M}_{ji} \right\} + \epsilon, \quad \forall i = 1, \dots, n. \quad (4.9)$$

Then, the matrix

$$A = \mathcal{M} \odot \left(S_{12} \left[N \odot \left(\frac{1}{2} (S_{11} + S_{22}) + V - V^\top \right) \right] \right)^{-1} \quad (4.10)$$

is Schur and presents the desired sparsity pattern \mathcal{M} .

Proof. See Appendix B.5. □

For a given sparsity pattern \mathcal{M} and small positive constant ϵ , one can thus define N and use the free parametrization (4.10) to compute Schur matrices presenting the desired sparsity pattern from some V and W .

Remark 21. Contrary to Propositions 3 and 4, Proposition 5 is conservative; it cannot capture all sparse Schur matrices. This stems from two steps in Appendix B.5. First, we have to restrict our search to systems admitting diagonal Lyapunov functions to leverage the associative property of Hadamard products with diagonal matrices. Second, satisfying (B.10)–(B.11) is only a sufficient condition for (B.9) to hold.

Remark 22. Setting $\mathcal{M} := \mathbb{1}_{n \times n}$ would provide another parametrization of dense Schur matrices, potentially replacing Proposition 3. However, this is not advised in practice since (4.10) is more conservative than (4.3), as discussed in Remark 21.

4.2.4 An alternative general parametrization

To showcase the power of PyTorch, which can differentiate through the computation of eigenvalues, the following proposition offers an alternative free parametrization of any Schur matrix. Contrary to the other parametrizations, it relies on scaling arguments instead of LMIs.

Proposition 6. *For a given sparsity pattern $\mathcal{M} \in \{0, 1\}^{n \times n}$, $0 < \gamma \leq 1$, and any matrix $V \in \mathbb{R}^{n \times n}$ and constant $\eta \in \mathbb{R}$, leveraging the sigmoid function σ , the matrix*

$$A = \frac{\sigma(\eta)\gamma}{|\lambda(\mathcal{M} \odot V)|_{\max}} (\mathcal{M} \odot V) \quad (4.11)$$

is Schur stable with $|\lambda_i(A)| < \gamma$, $\forall i = 1, \dots, n$, and presents the desired sparsity pattern \mathcal{M} .

Proof. See Appendix B.6. □

As can be seen, Propositions 5 and 6 can be used interchangeably; they both provide a free parametrization of sparse and stable matrices. Contrary to its conservative counterpart, however, Proposition 6 can capture *all* Schur matrices – including sparse ones —, as shown in the following corollary.

Corollary 4. *Any Schur matrix A satisfies (4.11) for some $\mathcal{M} \in \{0, 1\}^{n \times n}$, $0 < \gamma \leq 1$, $V \in \mathbb{R}^{n \times n}$, and $\eta \in \mathbb{R}$.*

Proof. See Appendix B.7 □

Interestingly, defining $\mathcal{M} := \mathbb{1}_{n \times n}$ in Proposition 6, we recover a free parametrization of generic matrices, providing an alternative to Proposition 3. However, according to Corollary 4, this would not come at the cost of expressiveness, contrary to Proposition 5. Similarly, parametrizing V as in (4.5) and using $\mathcal{M} := \mathbb{1}_{n \times n}$, we recover a parametrization of matrices close to identity interchangeable with Proposition 4. Overall, Proposition 6 hence allows us to characterize *any* type of Schur matrix discussed throughout this Section.

Remark 23. *Proposition 6 is philosophically related to [323], where a Lyapunov function is learned simultaneously to nominal system dynamics. At each step, the dynamics are then projected onto the Lyapunov function to guarantee asymptotic stability. Similarly, (4.11) can be seen as a projection onto some (unknown) Lyapunov function. However, the latter is implicitly defined through the scaling of A instead of being learned, hence alleviating the associated computational burden.*

Remark 24. *Propositions 3 and 5 can be adapted for continuous-time systems of the form (4.4) leveraging techniques similar to [332]. On the other hand, the scaling approach deployed in Proposition 6 to control the magnitude of the eigenvalues of A cannot be straightforwardly adapted to the continuous-time setting, where the real part of each eigenvalue has to be negative to ensure stability.*

```

## Import SIMBa and the default parameters
from simba.model import Simba
from simba.parameters import base_parameters as parameters

## Prepare the data (unused data can be set to None,
# e.g., X for input-output systems)
U, X, Y, x0 = train_data
U_val, X_val, Y_val, x0_val = validation_data
U_test, X_test, Y_test, x0_test = test_data

## Customize the parameters to the problem
parameters['input_output'] = True
# ...

## Declare Simba, train it, and save the results
simba = Simba(nx=nx, nu=nu, ny=ny, parameters=parameters)
simba.fit(U, U_val, U_test, X, X_val, X_test,
         Y, Y_val, Y_test, x0, x0_val, x0_test)
simba.save(directory=directory, save_name=save_name)

```

Figure 4.1: Main steps of running SIMBa in Python.

4.2.5 Using free parametrizations

Propositions 3–6 imply one can choose *any* $\epsilon > 0$, $0 < \gamma \leq 1$, $\mathcal{M} \in \{0, 1\}^{n \times n}$, $V \in \mathbb{R}^{n \times n}$, $W \in \mathbb{R}^{2n \times 2n}$, and $\eta \in \mathbb{R}$ — depending on the setting — and construct a stable matrix A as in (4.3), (4.7), (4.10), or (4.11). In practice, ϵ should be set to a small constant² and γ and \mathcal{M} are problem-specific and user-defined since they stem from prior knowledge about the system. Therefore, all *constrained* parameters are defined by the user *a priori*.

SIMBa then searches for V , W , and η optimizing some performance criterion, as detailed in Section 4.3. Since these parameters are *not* constrained, SIMBa can use unconstrained GD for this task. Propositions 3–6 hence allow us to leverage the full power of PyTorch’s AD to fit the data without jeopardizing stability, constructing a Schur matrix A from the *free parameters* V , W , and η at every iteration.

4.3 The SIMBa toolbox

SIMBa is implemented in Python to leverage the efficient AD framework of PyTorch [140]. It is open-sourced on <https://github.com/Cemempamoi/simba>. For given data sets \mathcal{D}_{train} , \mathcal{D}_{val} , and \mathcal{D}_{test} , SIMBa can be initialized, fit, and saved in a few lines of codes, as exemplified in Figure 4.1. For completeness, we also provide a MATLAB interface inspired by the traditional MATLAB SI toolbox [309], as shown in Figure 4.2. The rest of this section details the main parameters of SIMBa.

²We use $\epsilon = 1\text{e-}6$ in our experiments.

```

%% Prepare the data
data = iddata(Y, U, Ts);
data_val = iddata(Y_val, U_val, Ts);
data_test = iddata(Y_test, U_test, Ts);

% Set paths to Python virtual env and Simba's main directory
path_to_python_env = fullfile('.', '.venv', 'bin', 'python');
path_to_simba = '..';

%% Launch python and load default options
[pe, opt] = SIMBaOptions(path_to_python_env, path_to_simba);
% Modify parameters (some of them are required)
opt.delta = 'None';
opt.input_output = true;
% ...

%% Run SIMBa
sys = run_simba(opt, data, data_val, data_test, Ts);

```

Figure 4.2: Main steps to call SIMBa from MATLAB.

Credit assignment. The SIMBa toolbox stems from a collaboration between Muhammad Zakwan and the author of this thesis in [56, 57], although the software development was spearheaded by the author of this thesis.

4.3.1 Optimization framework

Given input-output data $\mathcal{D}^{i/o}$, SIMBa iteratively runs gradient descent on batches of trajectories $Z \in \mathcal{D}_{train}$ randomly sampled from the training data set — thus seamlessly handling training data sets consisting of several trajectories. We leverage PyTorch’s implementation of AD to solve the following optimization problem:

$$\min_{A, B, C, D, x_0^{(s)}} \frac{1}{|Z|} \sum_{s \in Z} \left[\frac{1}{l_s} \sum_{k=0}^{l_s} m_k^{(s)} \mathcal{L}_{train}(y^{(s)}(k), y_k^{(s)}) \right] \quad (4.12)$$

$$\text{s.t. } y_k^{(s)} = Cx_k^{(s)} + Du^{(s)}(k) \quad (4.13)$$

$$x_{k+1}^{(s)} = Ax_k^{(s)} + Bu^{(s)}(k). \quad (4.14)$$

In words, SIMBa minimizes the multi-step-ahead prediction error, using the training loss \mathcal{L}_{train} as performance criterion. In this paper, we rely on the Mean Square Error (MSE), i.e., $\mathcal{L}_{train}(y, \hat{y}) = \|y - \hat{y}\|_2^2$. However, SIMBa’s flexibility — backed by PyTorch’s ability to handle any differentiable function — allows one to design custom (differentiable) loss functions and pass them through the `train_loss` parameter. In some applications, it might be interesting to optimize the Mean Absolute Error (MAE) or the Mean Absolute Percentage Error (MAPE), for example, which are more robust against outliers or different output magnitudes, respectively.

In many cases, identifying the matrix D is not required, which is achieved in SIMBa by setting `id_D=False`, removing the second term of (4.13). Similarly, if $x^{(s)}(0)$ is known, `learn_x0` can

Chapter 4. Leveraging automatic differentiation for system identification

be toggled to `False` and SIMBa will fix $x_0^{(s)} := x^{(s)}(0)$ instead of optimizing it. The number of sequences $|Z|$ used for each gradient update can be controlled through the `batch_size` parameter. Finally, $m_k^{(s)} \in \{0, 1\}$ in (4.12), with

$$m_k^{(s)} = \begin{cases} 0, & \text{with probability } p \text{ or if } y^{(s)}(k) \text{ is NaN,} \\ 1, & \text{otherwise.} \end{cases}$$

In words, these binary variables let SIMBa discard missing values from the objective function, allowing it to seamlessly work with incomplete data sets. Additionally, the user can define a `dropout = p` parameter to randomly remove data points from the objective with probability p , providing empirical robustness to the training procedure.

4.3.2 Input-state identification

When state measurements are available in $\mathcal{D}^{i/s}$, one can set `input_output=False`, dropping (4.13), and modifying the objective to

$$\min_{A,B} \frac{1}{|Z|} \sum_{s \in Z} \left[\frac{1}{l_s} \sum_{k=0}^{l_s} m_k^{(s)} \mathcal{L}_{train}(x^{(s)}(k), x_k^{(s)}) \right]. \quad (4.15)$$

Furthermore, for autonomous systems, the autonomous flag can be toggled, in which case the minimization on B is also discarded, as well as the corresponding second term on the right-hand-side of (4.14). Similarly to Section 4.3.1, $m_k^{(s)}$ are binary variables that can be forced to zero either to discard missing values or as a means of regularization.

Since the state x is known, one can break given training trajectories into *segments* of length `horizon`. The `stride` defines how many steps should be taken between the *starts* of two segments. Note that if `stride` is smaller than `horizon`, then segments of data will overlap, i.e., data points will appear several times in consecutive segments. If the user is interested in the model performance over a specific horizon length, this can be specified with `horizon_val`, and the number of segments can also be controlled with `stride_val`. Note that setting `horizon` or `horizon_val` to `None` keeps entire trajectories.

4.3.3 Training procedure

SIMBa iteratively runs one step of gradient descent on (4.12) or (4.15) for `max_epochs` epochs. Here, we define an *epoch* as one pass through the training data, i.e., every trajectory has been drawn in a batch Z . After each epoch, the validation data is used to assess the current model performance by computing

$$\frac{1}{|\mathcal{D}_{val}|} \sum_{s \in \mathcal{D}_{val}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s} \mathcal{L}_{val}(y^{(s)}(k), y_k^{(s)}) \right] \quad \text{or} \quad \frac{1}{|\mathcal{D}_{val}|} \sum_{s \in \mathcal{D}_{val}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s} \mathcal{L}_{val}(x^{(s)}(k), x_k^{(s)}) \right],$$

in the input-output or input-state case, respectively. This evaluation metric is used to avoid SIMBa overfitting the training data: the best parameters, stored in memory, are only overwritten if the current parameters improve the performance on the validation set, and not the training one [307]. At the end of the training, to get a better estimate of the true performance of the identified model, we evaluate it on the unseen test data set with

$$\frac{1}{|\mathcal{D}_{test}|} \sum_{s \in \mathcal{D}_{test}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s} \mathcal{L}_{val}(y^{(s)}(k), y_k^{(s)}) \right] \quad \text{or} \quad \frac{1}{|\mathcal{D}_{test}|} \sum_{s \in \mathcal{D}_{test}} \left[\frac{1}{l_s} \sum_{k=0}^{l_s} \mathcal{L}_{val}(x^{(s)}(k), x_k^{(s)}) \right].$$

Throughout this work, we also rely on the MSE for validation and testing, setting $\mathcal{L}_{val} = \mathcal{L}_{train}$, but this can be modified through the `val_loss` parameter. This gives the users the freedom to evaluate SIMBa on a different metric than the training one. It can be tailored to specific applications: SIMBa would then return the model performing best with respect to the chosen evaluation criterion, irrespective of the training procedure.

Note that (4.12) or (4.15) can be highly nonconvex, in which case gradient descent cannot be expected to find the global optimum and will most likely settle in a local one instead. SIMBa is thus sensitive to its initialization and some hyperparameters and might converge to very different solutions depending on these choices.

4.3.4 Initialization

To start SIMBa in a relevant part of the search space, one can set `init_from_matlab_or_ls` to `True`. This prompts SIMBa to run a traditional SI method, i.e., either

- the MATLAB SI toolbox [309] or the Python SIPPY package [50] for input-output systems,³ or
- a traditional LS optimization in the input-state case,

before training. Depending on the setting, the chosen *initialization method* returns matrices A^* and potentially B^* , C^* , and D^* . These are then used as initial choices of state-space matrices in SIMBa, so that it starts learning from the best solution found by traditional SI methods.

However, to ensure the often-desired stability of A , SIMBa relies on the free parameterizations in Propositions 3–6, in which case it is not possible to directly initialize the matrix A to A^* . We thus again resort to PyTorch to approximate it by solving the following optimization problem with unconstrained GD:

$$\min_{W, V, \eta} \mathcal{L}_{init}(A, A^*) \tag{4.16}$$

$$\text{s.t. } A \text{ as in (4.3), (4.7), (4.10), or (4.11)}, \tag{4.17}$$

³Since several traditional SI methods are available, SIMBa uses the one achieving the best performance on the validation set.

Chapter 4. Leveraging automatic differentiation for system identification

with \mathcal{L}_{init} the desired loss function. We use the MSE throughout our numerical experiments but custom functions can be passed through the `init_loss` parameter.

Similarly to what was mentioned in Section 4.3.3 concerning SIMBa’s training, this procedure is not guaranteed to find the global optimal solution, i.e., to converge to A^* . Consequently, even initialized instances of SIMBa might perform very differently from the initialization method, sometimes incurring large performance drops. Nonetheless, throughout our experiments, it usually worked well, finding an A close to A^* .

Remark 25. *Note that A might be initialized exactly in the specific case when Proposition 6 is leveraged for stability and $\gamma > |\lambda(A^*)|_{max}$, as detailed in the proof of Corollary 4. On the other hand, although Corollaries 2–3 demonstrate that any Schur matrix A^* can be parametrized as (4.3) or (4.7), respectively, they only show the existence of such a parametrization and cannot be used in practice to construct it.*

4.3.5 Prior knowledge integration

If certain sparsity patterns are desired for A , B , C , or D , they can be passed through `mask_{X}`, replacing `{X}` with the name of the corresponding matrices. If the mask of B is given as \mathcal{M}_B , for example, (4.14) is modified to

$$x_{k+1}^{(s)} = Ax_k^{(s)} + (\mathcal{M}_B \odot B)u^{(s)}(k) \quad (4.18)$$

to force the desired entries of B to zero while letting SIMBa learn the others.

Similarly, `{X}_init` is used to initialize a given matrix to a specific value, and `learn_{X}=False` drops the corresponding matrix from the optimization, fixing it at its initial value. To control the magnitude of the eigenvalues of A in the free parameterizations of Propositions 3 and 6, one can set `max_eigenvalue = γ` .

Finally, `stable_A=True` enforces the stability of A : setting `naive_A=True` leverages Proposition 6 while toggling `LMI_A` uses Propositions 3–5. Specifically, if `delta` is not `None` but takes the value δ , hinting we are expecting A to be close to the identity matrix, then Proposition 4 is used instead of Proposition 3. Similarly, if `mask_A` is not `None`, hence requiring a sparse system, Proposition 5 is leveraged.

When `stable_A=True`, the minimization over A in (4.12) or (4.15) is replaced by a minimization over W , V , and/or η — depending on which of the four Propositions is used — and constraint (4.17) is added to the corresponding optimization problem.

4.3.6 Tuning of critical hyperparameters

As for NNs, which heavily rely on the same backpropagation backbone, the `learning_rate` is an important parameter: too large values lead to unstable training while too small ones

4.4 Benchmarking SIMBa through numerical experiments

slow the convergence speed. In general, throughout our empirical analyses, the default value of $1e-3$ showed very robust performance, and we couple it with a high number of epochs to ensure SIMBa can converge close to a local minimizer. However, this is problem-dependent, and the rate might be increased to accelerate learning if no instability is observed. Similarly, `init_learning_rate` controls the learning rate of the initialization in (4.16)–(4.17) when required. It also defaults to $1e-3$ due to its robust performance when coupled with a high number of `init_epochs`.

To promote stable learning, we implemented a gradient clipping operation, pointwise saturating the gradients of all the parameters to avoid taking overly aggressive update steps during GD. These can be controlled through `grad_clip` and `init_grad_clip` during the training and initialization phase, respectively. The default values of 100, respectively 0.1, were empirically tuned to achieve good performance. Although it might come at the cost of a slower convergence, our numerical investigations showed gradient clipping can significantly improve the quality of the solution found by SIMBa.

4.4 Benchmarking SIMBa through numerical experiments

As described in Section 4.3, SIMBa can identify models from input-output and input-state measurements while seamlessly enforcing desired system properties such as stability or prior knowledge on the state-space matrices. This Section provides numerical examples showcasing its ability to outperform traditional SI methods in a wide variety of case studies. It exemplifies how SIMBa leverages Propositions 3–6 to guarantee the stability of the identified model while achieving state-of-the-art fitting performance. Interestingly, our investigations hint that integrating prior knowledge, while being easy, does not impact the quality of the solution found by SIMBa in general. On the contrary, it seems that domain knowledge injection can be helpful to improve performance.

Sections 4.4.1–4.4.4 first analyze the behavior of SIMBa on different simulated input-output data sets. Throughout Sections 4.4.1–4.4.2, we fix $x_0 = 0$, which allows us to compare SIMBa with SIPPY’s implementations of SIMs and PARSIMs [50].⁴ To assess the impact of prior knowledge integration, Sections 4.4.3–4.4.4 subsequently compare SIMBa’s performance to the one of the MATLAB SI toolbox [309]. When dealing with real-world input-output data in Section 4.4.5, however, enforcing $x_0 = 0$ — as is done in SIPPY — leads to suboptimal performance, and we compare SIMBa with the performance of MATLAB’s SI toolbox when x_0 is estimated. Section 4.4.6 then exemplifies how SIMBa can surpass the standard LS method and the state-of-the-art SOC approach for stable SI from [333] on a real-world input-state SI task. Finally, Section 4.4.7 details the computational burden associated with SIMBa.⁵

⁴In these cases where $x_0 = 0$, the results found by MATLAB’s SI toolbox [309] turn out to be either comparable or slightly worse than SIPPY’s solutions. They are thus not reported herein.

⁵For reproducibility, the code and data used for these experiments can be found on <https://gitlab.nccr-automation.ch/loris.dinatale/simba-ecc> [56] and <https://gitlab.nccr-automation.ch/loris.dinatale/simba> [57].

Method	0.25-quantile	Median	0.75-quantile
SIMBa	1.00	1.02	1.14
CVA	1.12	1.30	1.80
MOESP	1.15	1.34	1.93
N4SID	1.20	1.42	1.97
PARSIM-K	1.00	1.04	1.22
PARSIM-S	1.53	3.65	37.5
PARSIM-P	1.73	5.37	84.2

Table 4.1: Normalized MSE of each method compared to the best one, corresponding to Figure 4.3.

Credit assignment. The numerical experiments in this section stem from a collaboration between Muhammad Zakwan and the author of this thesis in [56, 57], although it was spearheaded by the author of this thesis.

4.4.1 Comparison using random stable models

To assess the performance of SIMBa on standard SI problems, we started by generating 50 random stable discrete-time state-space models, from which we simulated one trajectory of 300 steps, starting from $x_0 = 0$, for the training, validation, and testing data, respectively. For the three trajectories, each dimension of $u \in \mathbb{R}^m$ was generated as a Generalised Binary Noise (GBN) signal with a switching probability of 0.1 [50]. We then added white output noise $v \sim \mathcal{N}(0, 0.25)$ to the training data. For this experiment, we arbitrarily chose $n = 5$, $m = 3$, $r = 3$, set LMI_A=True to leverage Proposition 3, and kept the other parameters of SIMBa at their default value, except for the number of epochs, increased to 50'000 to ensure convergence.

The performance of each SI method on the testing trajectory is plotted in Figure 4.3, where green indicates SIMBa, blue other stable SI methods, and red PARSIMs, which cannot enforce stability. For each system, the MSE of each method x was normalized with respect to the best-attained performance by any approach as $\text{MSE}_x / \text{MSE}_{\text{best}}$ to generate the box plots, and the corresponding key metrics are reported in Table 4.1. For a better visual representation, we overlaid the corresponding clouds of points, where we added random noise on the x-axis to distinguish them better. Note that this zoomed-in plot does not show one instance where SIMBa did not converge and attained poor performance, while it discards three such instances for PARSIM-K and many points with a normalized MSE between 3 and 7 for the methods in blue.

Overall, SIMBa shows the most robust performance, with 75% of its instances achieving an error within 14% of the best performance and half of them being near-optimal (see Table 4.1). The only method coming close is PARSIM-K, but its performance is slightly more spread out and it cannot guarantee stability. If we only look at other stable SI methods, their median accuracy is at least 30% worse than the best one half of the time. In fact, their performance

4.4 Benchmarking SIMBa through numerical experiments

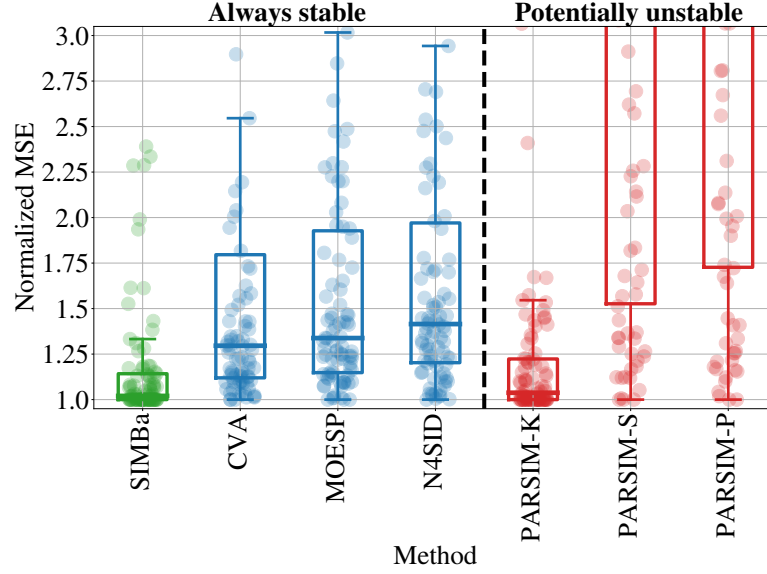


Figure 4.3: Performance of input-output state-space SI methods on 50 randomly generated systems, where the MSEs have been normalized by the best-obtained error for each system. The performance of SIMBa (ours) is plotted in green, other stable SI methods in blue, while red indicates methods without stability guarantees. Key metrics are reported in Table 4.1 for clarity.

drop is more than 12% three times out of four, compared to approximately one-fourth of the time for SIMBa, and their accuracy on the various systems is significantly more spread out. To summarize, **SIMBa takes the best out of both worlds, simultaneously achieving state-of-the-art performance and stability guarantees.**

4.4.2 Comparing Propositions 3 and 6 using random systems

To complement the results in Section 4.4.1 showing the superiority of SIMBa on randomly generated systems when leveraging Proposition 3 to guarantee stability, we generated 50 additional stable discrete systems using the same settings to compare the performance when Proposition 6 is used instead. Note that both free parametrizations can capture *all* stable matrices and thus find the true solution (see Corollaries 2 and 4). For each system, we defined two instances of SIMBa, one with `LMI_A=True` (*SIMBa-3*) and another with `naive_A=True` (*SIMBa-6*).

First, the normalized performance of each SI method on the testing trajectory from 30 systems is plotted in Figure 4.4, where green indicates SIMBa, blue other stable SI methods, and red PARSIM methods, which cannot enforce stability. We observe similar performance between both parametrizations, with a slight edge on Proposition 3. It achieved a median performance 8.2% worse than the best method on the different systems, compared to the 9.8% of *SIMBa-6*,

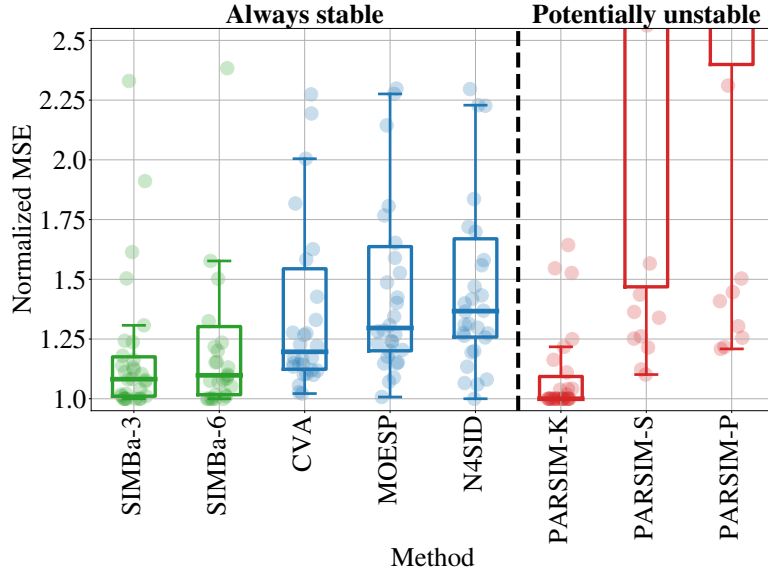


Figure 4.4: Performance of input-output state-space identification methods on 30 randomly generated systems, where the MSEs have been normalized by the best-obtained error for each system. The performance of SIMBa (ours) — either relying on the parametrization proposed in Proposition 3 or 6 — is plotted in green, other stable SI methods in blue, and unstable ones in red.

as reported in Table 4.2 for clarity. Note that PARSIM-K achieves impressive performance in this setting — it is the only traditional SI method able to compete with SIMBa in terms of accuracy, confirming the trend observed in Section 4.4.1. However, it cannot guarantee the stability of the identified model. On the other hand, other stable SI methods reached a performance drop of more than 20% compared to the best method half of the time (see Table 4.2).

Note that *SIMBa-6* additionally shows a slightly less robust performance than *SIMBa-3*, with a wider interquartile range. This hints that while the free parametrization in Proposition 6 does capture all stable matrices, it might be numerically less stable than the LMI-based one from Proposition 3.

Finally, for completeness, we used the other 20 generated systems to assess the impact of data standardization on the final performance. Before the SI procedure, each dimension of the dataset was processed to have zero mean and unit standard deviation, removing the effect of different dimensions having different magnitudes, as is often done in practice. As pictured in Figure 4.5, however, little impact can be seen, with the different methods reaching similar performance to what was observed in Figure 4.4. On the contrary, there seems to be a slightly wider gap between *SIMBa-3* and the other stable SI approaches in blue. Similarly to the previous case, *SIMBa-6* again slightly underperformed compared to its counterpart leveraging Proposition 3, providing additional indications that the parametrization in Proposition 6

4.4 Benchmarking SIMBa through numerical experiments

Method	0.25-quantile	Median	0.75-quantile
SIMBa-3	1.01	1.08	1.18
SIMBa-6	1.02	1.10	1.30
CVA	1.12	1.20	1.54
MOESP	1.20	1.30	1.64
N4SID	1.26	1.37	1.67
PARSIM-K	1.00	1.00	1.09
PARSIM-S	1.47	4.75	30.91
PARSIM-P	2.40	7.00	223.76

Table 4.2: Performance drop of each method compared to the best one, reported from Figure 4.4.

might be numerically more challenging.

Since little performance difference can be observed between Figures 4.3–4.5, data standardization does not seem to impact SIMBa’s performance significantly in general. Interestingly, this means even the gradient-based SIMBa can be run to fit data with different orders of magnitudes accurately. We suspect gradient clipping to be an important reason behind this strong performance, but further analyses would be required to understand the behavior of GD in SIMBa fully.

4.4.3 Introducing prior knowledge

As a next case study, we analyzed the effect of incorporating various levels of prior knowledge — i.e., enforcing known sparsity patterns or true values of one or several of the state-space matrices — into SIMBa without jeopardizing stability. To that end, we used the same simulation settings as in Sections 4.4.1–4.4.2 to create 10 systems but with $n = 7$, $m = 6$, $p = 5$, and trajectories of length 500. Before generating the data, however, we randomly set 60% of the entries of A , B , C , and D to zero.⁶ We let SIMBa run for 25’000 epochs. To ensure stability, we set `LMI_A=True` to leverage Proposition 5 or 3 when `mask_A` is known or not, respectively, or `naive_A=True` to use Proposition 6. When the latter parametrization of Schur matrices was leveraged, we additionally ran several instances of SIMBa to assess the impact of randomness — the random seed and initialization of the parameters — on its performance, and we report here the median and minimum error achieved on each of the ten generated systems. This shows what can be expected on average but also the best attainable performance with *SIMBa*-6. For comparison, the same tasks were solved with the `ssest` function from the MATLAB SI toolbox [309].

The resulting normalized errors are presented in Figure 4.6, where the plot has been generated as in Sections 4.4.1–4.4.2, and the bottom figure is a zoomed-in version for better visualization of the differences between the various instances of SIMBa. The known system properties

⁶We made sure that A remained stable after this sparsification procedure.

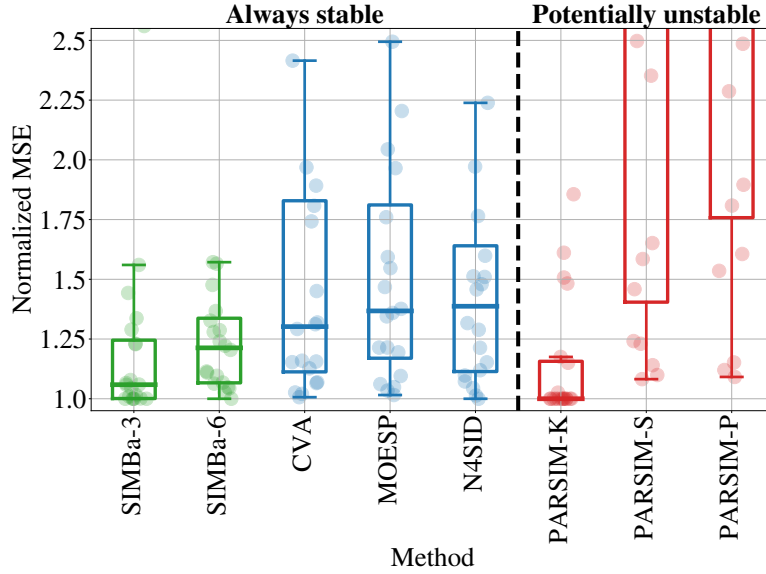


Figure 4.5: Performance of input-output state-space identification methods on 20 randomly generated systems, where the data has been standardized and the MSEs have been normalized by the best-obtained error for each system. The performance of SIMBa (ours) — either relying on the parametrization proposed in Proposition 3 or 6 — is plotted in green, other stable SI methods in blue, and potentially unstable ones in red.

incorporated in each SIMBa or MATLAB instance is encoded in square brackets in their name, where “ X ” or “ m_X ” indicates that the true matrix X or its true sparsity pattern was given through `{X}_init` or `mask_{X}`, respectively.⁷ For example, `[mBCD]` represents instances with knowledge of C , D , and the sparsity pattern of B . In practice, this could correspond to a system where $D \equiv 0$ and we know which states are measured (i.e., C is known) and which inputs act on which states but not their exact impact (i.e., the sparsity pattern of B is known). This is encoded in SIMBa by setting `learn_C=learn_D=False`, passing the known matrices C and D as `C_init` and `D_init`, respectively, and defining `mask_B` to be the true known sparsity pattern of B .

In general, except for `SIMBa-3 [mBCD]`,⁸ increasing levels of prior knowledge are positively reflected in SIMBa’s performance when leveraging Propositions 3 or 6 in this case study (from right to left in Figure 4.6), hinting at the efficacy of system properties incorporation in SIMBa. On the one hand, this makes intuitive sense since we pass *true* information to SIMBa, restricting the search space. On the other hand, enforcing fixed matrices or sparsity patterns reduces the expressiveness of the model to fit the training data well — and GD might get stuck in a poor local minimum. Indeed, the set of all possible state-space matrices, over

⁷When the true matrix X is given to SIMBa, `learn_{X}` is set to `False`, so that it is not modified during learning.

⁸Interestingly, MATLAB also struggled to converge to meaningful solutions in this case, hinting that the task of fitting a Schur A and selected entries of B was not trivial in this experiment.

4.4 Benchmarking SIMBa through numerical experiments

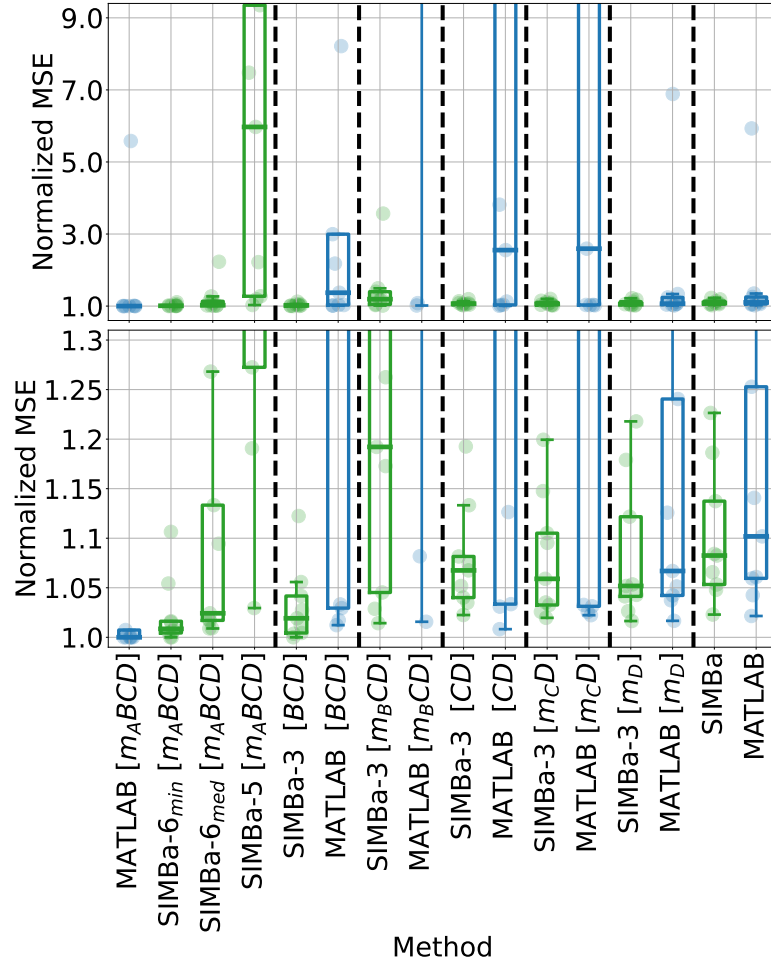


Figure 4.6: Normalized MSE of each method on test input-output data from 10 randomly generated systems with sparse matrices A , B , C , and D . The letters in square brackets encode which matrices X or sparsity pattern m_X , respectively, are assumed to be known and fixed. Both plots show the same data with a different zoom to appreciate the difference between SIMBa (ours) — either relying on Proposition 3, 5 or 6 — in green and the `ssest` function in the MATLAB SI toolbox (in blue). Note that *SIMBa-6* was run with eight different random seeds on each system, and we report both the median and minimum error.

which *SIMBa* optimizes,⁹ contains the sparse matrices the other instances are optimizing over. *SIMBa* might hence find state-space matrices achieving a better MSE without respecting the desired system properties. Although *SIMBa-5* [m_{ABCD}] seems to have been impacted and stuck in local minima, the other informed versions of *SIMBa* all achieved errors within 25% of the best one in almost all cases.

In contrast, MATLAB only achieved reasonable accuracy in three of the seven experiments, when either no or little prior knowledge was enforced (on the right of Fig. 4.6), or when only the values of A with known sparsity pattern needed to be learned from data (on the left of the plot). These results provide evidence that enforcing desired system properties, even if our assumptions are correct, might deteriorate MATLAB's performance significantly. On the contrary, *SIMBa* converged to accurate solutions throughout our experiments, typically significantly outperforming standard SI methods.

The only setting where MATLAB outperformed *SIMBa* was when only selected entries of A had to be learned, assuming all other matrices to be known (m_{ABCD}). Nevertheless, as can be seen in the top of Fig. 4.6, MATLAB did not converge to a meaningful solution on one system, only obtaining a testing accuracy more than five times lower than the best one. In contrast, *SIMBa-6_{min}* [m_{ABCD}] showed a more consistent performance across the different systems, never achieving an error more than 11% off the best one.

Altogether, our investigations hint that ***SIMBa* indeed allows one to impose known or desired system properties without sacrificing significant model performance in general, contrary to MATLAB**. There is however one critical exception: Proposition 5 seems to impose overly conservative conditions on sparse Schur matrices, in line with Remark 21, and often led to poor testing accuracy in this case study (*SIMBa-5* [m_{ABCD}]).

4.4.4 Identifying sparse Schur matrices

To complement Section 4.4.3 and assess the efficacy of Proposition 6 in parametrizing sparse Schur matrices, this Section offers another set of more challenging identification experiments, where Schur A matrices with known sparsity patterns have to be identified simultaneously to other state-space matrices. To that end, we used the same simulation settings and ten systems as in Section 4.4.3, running several randomly initialized instances of *SIMBa-6* and reporting both the corresponding median and best performance. All the results on the testing data are plotted in Fig. 4.7, with the bottom figure being a zoomed-in version to better appreciate the impact of prior knowledge on *SIMBa*.

As can be seen, apart from the m_{ABCD} case already discussed in the previous Section, where MATLAB could attain lower errors, *SIMBa* always performed significantly better than the baseline. In fact, MATLAB failed to converge to meaningful solutions in all the other experiments, consistently producing errors that were often orders of magnitude more severe than

⁹More specifically, it optimizes over the space of stable matrices A and generic matrices B , C , and D .

4.4 Benchmarking SIMBa through numerical experiments

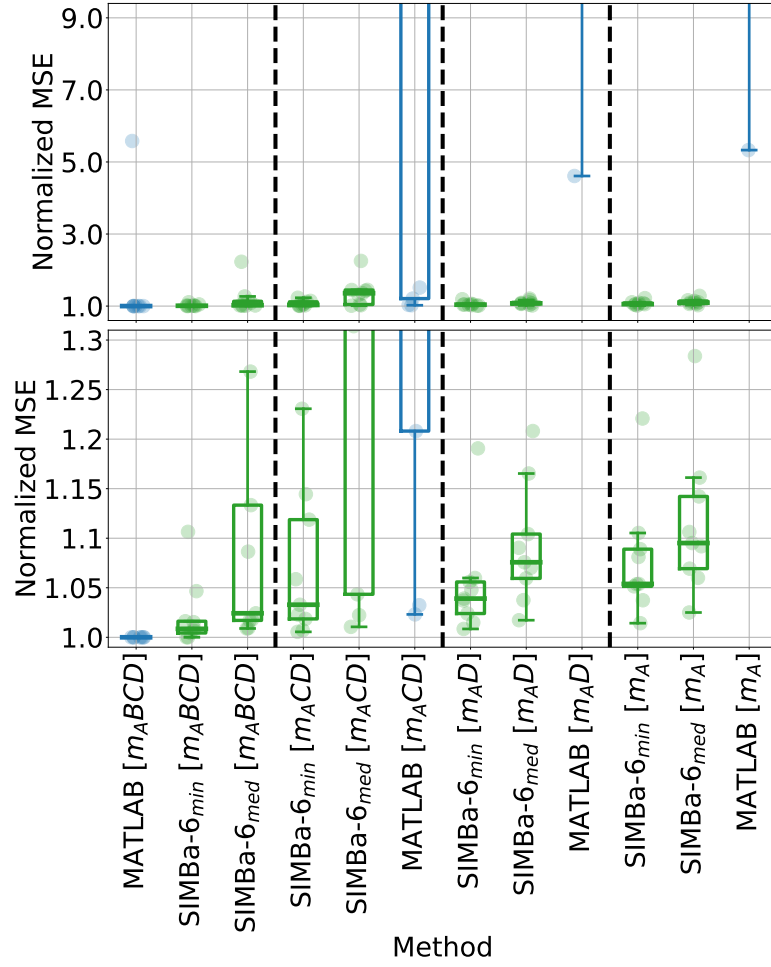


Figure 4.7: Performance of each method, normalized by the best one, on test input-output data from 10 randomly generated systems with sparse matrices A , B , C , and D . The performance of SIMBa (ours) — relying on Proposition 6 — is plotted in green, the one of the `ssest` function in the MATLAB SI toolbox in blue, and the bottom plot is a zoomed-in version of the top one for better visualization. Note that SIMBa was run with 10 different random seeds on each system, and we report both the median and minimum error.

Chapter 4. Leveraging automatic differentiation for system identification

the best result attained by SIMBa. Notably, while the quality of the solutions found by MATLAB significantly deteriorates with the number of parameters to identify simultaneously to the sparse Schur matrices A (from left to right in Figure 4.7), SIMBa could keep similarly low errors in all cases. Furthermore, in line with the observations made in the previous Section, we can observe a positive correlation between SIMBa’s accuracy and the quantity of pre-existing knowledge about the state-space matrices to identify.

On the other hand, SIMBa showed sensitivity to randomness on some systems, with the minimum error achieved being distinctly lower than the median one. However, this did not seem to significantly impact SIMBa’s best performance. Indeed, *SIMBa-6_{min}* could achieve low testing error on all the systems, with its accuracy remaining within 22% of the best one in all cases and within 10% three times out of four. Although running several instances of SIMBa naturally incurs additional computational cost, these results hint that it can be beneficial to find a better-performing model.

Collectively, these results show that introducing prior knowledge on the sparsity of A does not necessarily come at the price of performance, complementing what was observed in Section 4.4.3 for other knowledge integration schemes. Altogether, this shows how **SIMBa can conform to system properties desired by the user without significantly suffering in terms of performance**.

4.4.5 Performance on real-world input-output data

After extensive analyses in simulation, we now leverage DAISY, a database for SI [334], to test our framework on real-world data. In particular, we investigate the performance of SIMBa in detail on the data collected in a 120 MW power plant in Pont-sur-Sambre, France, where $m = 5$ and $p = 3$. It gathers 200 data points with a sampling time of $\delta = 1228.8$ seconds. Here, we first standardized the input and output data so that each dimension is zero-mean and has a standard deviation of one.¹⁰ We used the first 100 and 150 samples for training and validation, respectively, and held out the last 50 ones for testing the final performance of the models.¹¹

We investigate four variations of SIMBa, encoded in their names: an “*i*” indicates instances with `init_from_matlab_or_ls=True`, and an “*L*” that SIMBa was run for more epochs to ensure convergence. Specifically, the number of epochs with “*L*” is pushed from 10’000 to 20’000 for *SIMBa_i* and from 25’000 to 50’000 otherwise. We set `dropout=0`, `learn_x0=True` — since it is unknown —, and leave the other parameters at their default values. Since the true order of the system is unknown, one could leverage MATLAB’s SI toolbox to first find the most appropriate n and then run SIMBa to gain time. Here, we instead show that SIMBa dominates all the other methods from the MATLAB SI toolbox for *any* choice of n . The PARSIMs are

¹⁰Standardization generally has little impact on the performance, as analyzed in Section 4.4.2.

¹¹Since x_0 is estimated by SIMBa at training time, overlapping the training and validation data allows us to use the same initial state to validate its performance after each epoch. However, we let it run for 50 more steps to assess its extrapolation capability and avoid overfitting the training data. For testing, we rely on MATLAB’s `findstate` function to estimate x_0 .

4.4 Benchmarking SIMBa through numerical experiments

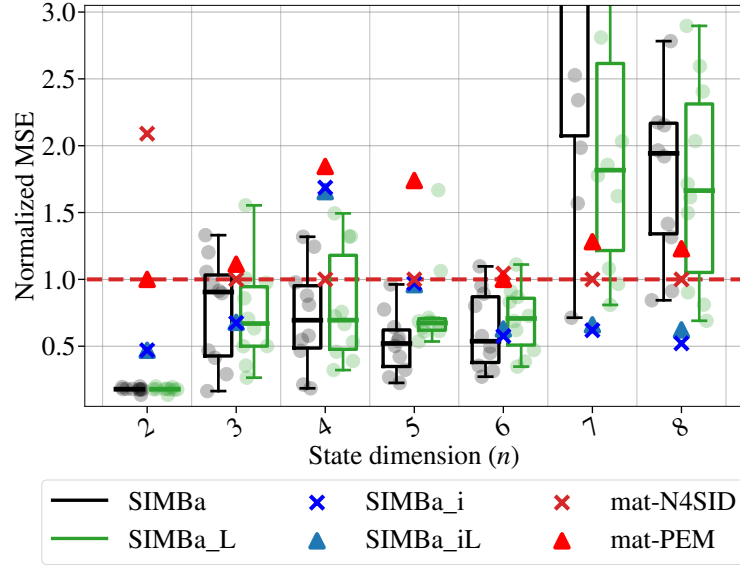


Figure 4.8: MSE of the different methods on the power plant test data for different choices of state dimension n , normalized by the best performance obtained by MATLAB’s SI toolbox (red crosses and triangles). Black and green data show the performance of SIMBa over 10 runs with random initialization for shorter (*SIMBa*) and longer (*SIMBa_L*) training times, respectively. Finally, blue crosses triangles represent the performance of one initialized version of SIMBa, *SIMBa_i*, and a prolonged version *SIMBa_iL*.

not analyzed here since they all diverged for at least one value n due to instability. Similarly, the stable methods from SIPPY achieved poor accuracy due to their assumption that $x_0 = 0$, which seems too restrictive for this data, and are thus omitted for clarity.

Figure 4.8 reports the MSE of the different methods on the testing data normalized by the best performance obtained by MATLAB’s SI toolbox. This either corresponds to *N4SID*¹² (red crosses) or the Prediction Error Method (PEM) (orange triangles). The latter aims at improving the performance of the model found by *N4SID* and is thus expected to perform better on the training data. Both randomly initialized versions of SIMBa, reported in black and green, were run with 10 different seeds and the boxplot and clouds of points were generated as for Figure 4.3. Since randomness has much less impact on initialized versions of SIMBa, we only report one instance of *SIMBa_i* and *SIMBa_iL* for clarity. Note that fitting a model with $n \geq 7$ on $N = 100$ data samples is an ill-posed problem, with more parameters than data points. Interestingly, however, SIMBa still manages to outperform MATLAB in some cases in this overparametrized setting, especially when it is initialized from MATLAB’s solution.

Impressively, SIMBa consistently attained the best performance for meaningful choices of

¹²Note that we set `N4Weight='auto'` — to automatically recover the best performance between the classical *N4SID*, *CVA*, and *MOESP* methods — and `Focus='simulation'` for a fair comparison with SIMBa, which is optimizing for the performance over the entire trajectory.

$n \leq 6$ given the training data size. While the influence of randomness is non-negligible for randomly initialized versions, **SIMBa always achieves the best accuracy, with improvements of up to 73–86% compared to MATLAB for different choices of n** . Furthermore, half of the time, it outperforms the latter by more than 30–50% and 82% for $n \geq 3$ and $n = 2$, respectively.

When being initialized with the solution of traditional SIMs, SIMBa always started from state-space matrices identified by MATLAB’s PEM method, which achieves the best performance amongst the baselines on the validation data.¹³ Interestingly, SIMBa always improves PEM’s performance on the testing set — but sometimes not beyond N4SID, which attains the best accuracy amongst the baselines on this unseen data. In other words, PEM tends to overfit the training data and might start off SIMBa near a poor local minimum. **While initializing with MATLAB’s solution allows one to converge faster, cutting the associated computational burden, it might hence not always improve the final performance.**

A note on the training complexity

Here, *SIMBa_L* was run for five times more epochs than *SIMBa_i*, for example. Despite the small overhead required to fit A during the initialization procedure in (4.16), training *SIMBa_i* still takes approximately only 20% of the time required to fit *SIMBa_L*. In this experiment, training SIMBa ranged from 5 to 25 min on a MacBook Pro 2.6 GHz 6-Core Intel Core i7 laptop, irrespective of the choice of n . Additionally, the training time is directly proportional to the training data length: doubling its size would double SIMBa’s fitting time. More run-time analyses can be found in Section 4.4.7.

4.4.6 Performance on real-world input-state data

To showcase SIMBa’s versatility, we finally turn to an input-state data set collected from the Franka Emika Panda robotic arm and provided in [333]. We have access to eight trajectories of length $N = 400$ collected at 50 Hz with $n = 17$ and $m = 7$. We fixed one validation and one test trajectory, respectively, and used a subset of the remaining six trajectories as training data. Since the robot is a continuous-time system, we leveraged Proposition 4, setting $\text{delta} = \frac{1}{50}$ and $\text{LMI_A} = \text{True}$. As we are now dealing with input-state data, traditional SIMs performed poorly, and we hence compare SIMBa to LS and its state-of-the-art stable version, SOC [333].

Here, we also used the ability of SIMBa to work with batched data, breaking the training trajectories into 10-step long segments, overlapping at each time step, — i.e., setting $\text{horizon} = 10$ and $\text{stride} = 1$ — to facilitate training for this more complex problem. This gave rise to approximately 400 to 2’400 training sequences of 10 steps whether one to six trajectories were used for training. Since we are interested in the final performance of SIMBa over entire trajectories, we set $\text{horizon_val} = \text{None}$ to select the best model accordingly. We let instances initialized with the LS solution (*SIMBa_i*) and randomly initialized ones (*SIMBa*) run for 20’000 and

¹³Except for $n = 2$, where SIMBa was initialized from PARSIM-K.

4.4 Benchmarking SIMBa through numerical experiments

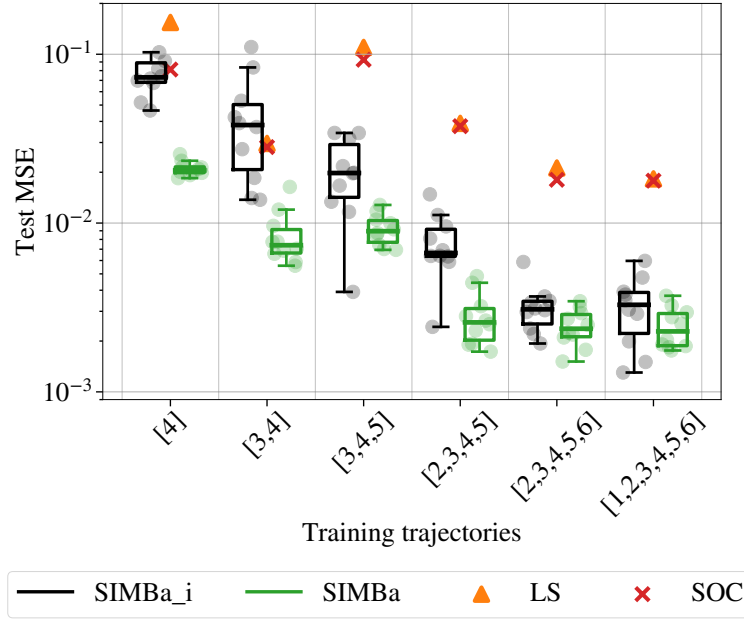


Figure 4.9: MSE of each method on the test trajectory of the Franka data set after training from different trajectories. Black and green data show the performance of SIMBa over 10 runs with random initialization for shorter (*SIMBa*) and longer (*SIMBa_L*) training times, respectively. The MSE of LS and SOC are reported in orange triangles and red crosses, respectively.

40'000 epochs, respectively, with a batch size of 128.

The MSE of each method on the test trajectory is reported in Figure 4.9 with a logarithmic scale, where the x-axis enumerates which trajectories were used for training. Similarly to the previous Section, SIMBa was run 10 times in each case to assess the impact of randomness. As expected, we see a general tendency of all the methods to find more accurate solutions with more training data. Interestingly, *SIMBa* often performs better than *SIMBa_L* on this data, hinting that initializing SIMBa with the matrices found through LS might stick it in relatively poor local minima.

Overall, SIMBa generally outperforms LS and SOC, and often significantly, especially when more training data is available. The only exceptions come from *SIMBa_L* when one or two trajectories only are used for training, where we can see performance drops for some instances. On the other hand, ***SIMBa* always outperforms LS and SOC, and with an impressive median performance improvement compared to SOC of over 70% and as high as 95%**, as reported in Figure 4.10. Here, the *improvement* is computed as

$$\text{Improvement} = 100 \left(1 - \frac{\text{MSE}_{\text{SIMBa}}}{\text{MSE}_{\text{SOC}}} \right).$$

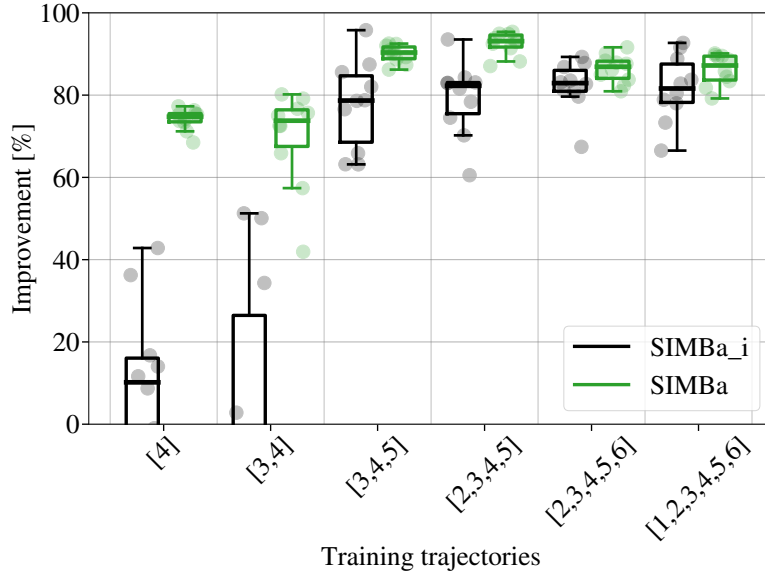


Figure 4.10: Improvement of SIMBa over SOC on the test data from the Franka robotic arm for different training trajectories, reported from Figure 4.9.

Moreover, looking at the best-achieved performance of *SIMBa* and *SIMBa_i*, they attained improvements of over 90% compared to SOC as soon as more than three training trajectories were used. When only one or two trajectories were leveraged for training, *SIMBa_i* achieved 42% or 51% better performance than SOC, respectively, and this improvement increases to 77% or 80% for *SIMBa*.

In general, we suspect the observed performance gaps to be heavily impacted by the ability of *SIMBa* to minimize the error over *multiple steps*, compared to myopic classical LS-based methods. This showcases the usefulness of backpropagation-based approaches, which can handle complex fitting criteria instead of the classical one-step-ahead prediction error.

4.4.7 Training complexity

To conclude these numerical investigations, this section provides insights into the training time of different versions of *SIMBa*. Except for Section 4.4.5, all the experiments were run on a Bizon ZX5000 G2 workstation. Note that while a GPU interface is implemented, we did not use it to obtain the results presented in this section, setting `device='cpu'`. Indeed, using GPUs for such small-scale problems generally slows the overall training time since the overhead required to move data and models to the GPU at each iteration is higher than the subsequent optimization time gain.

First, as reported in Table 4.3, each instance of *SIMBa* ran for slightly less than 1 h in Section 4.4.3, which is several orders of magnitude slower than the few seconds required to fit

4.4 Benchmarking SIMBa through numerical experiments

Table 4.3: Average running time in seconds of each method in Section 4.4.3.

Prior knowledge	MATLAB	SIMBa
–	1.67	3729
m_D	1.05	3713
m_CD	0.96	3512
CD	0.82	3388
m_BCD	0.83	3671
BCD	0.55	3130
m_ABCD	0.28	3100

traditional SI methods, typically using the MATLAB SI toolbox. Interestingly, however, enforcing prior knowledge did not significantly impact its run-time. Since fewer parameters need to be learned from data, informed versions tended to take less time per epoch. This stems from SIMBa’s architecture and unconstrained training procedure in (4.12) or (4.15). It allows SIMBa to seamlessly guarantee the desired system properties through modifications of (4.13)–(4.14) — for example, to (4.18) — without additional computational burden.

Note that these SIMBa instances were run for 25’000 epochs, and doubling that number would hence double their training time to approximately 2 h. The training procedure would then be comparable to *SIMBa_L* in Section 4.4.5, where SIMBa needed approximately 25 min for 50’000 epochs, i.e., around five times less. However, the latter was trained over 100 data points, compared to 500 in Section 4.4.3, revealing an approximately linear relationship between the horizon length and training complexity.

For completeness, the training times of the SIMBa instances analyzed in Section 4.4.6 are shown in Figure 4.11, exposing the expected linear impact of leveraging more and more training data. However, five to six times more data can be used before doubling the training time, leveled by PyTorch’s capability to process several trajectories in parallel. Unsurprisingly, on the other hand, doubling the number of iterations approximately yields twice longer fitting times (comparing *SIMBa_i* to *SIMBa*).

As a final remark, we would like to highlight here that all the analyzed instances of SIMBa were usually run for more epochs than required to ensure convergence. In practice, one could interrupt the training once the validation error stagnates or augments, showing SIMBa started to overfit the training data. To illustrate this, Figure 4.12 reports the time required by SIMBa to achieve its best performance on the validation data set — these state-space matrices are the ones ultimately employed to assess its performance on the testing data. As can be seen, SIMBa sometimes identifies the best-performing solution in considerably less time than the total allowed training time. Similarly, the learning rate could be increased in practice to converge faster to these solutions; however, a comprehensive analysis of its influence was out of the scope of this thesis.

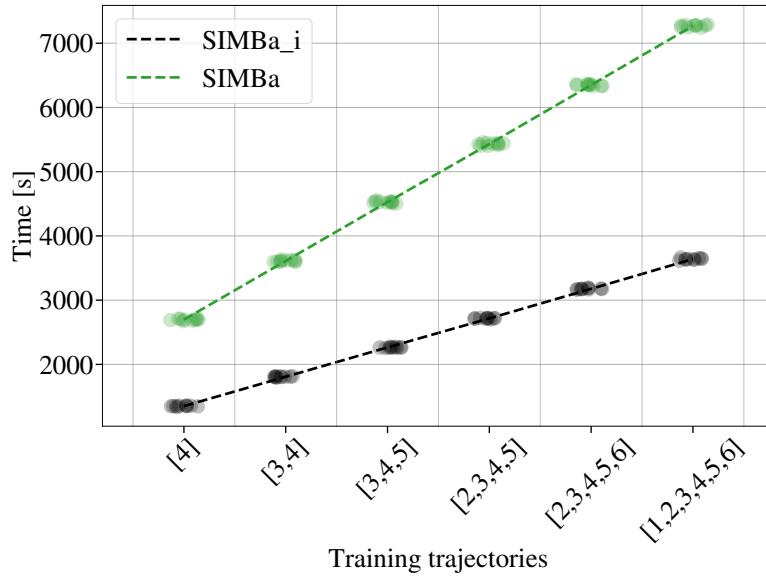


Figure 4.11: Time required by the different instances of SIMBa (in seconds) to obtain the results reported in Figure 4.9. Note that this does not include approximately 130 s required to initialize *SIMBa_i* for 150'000 epochs.

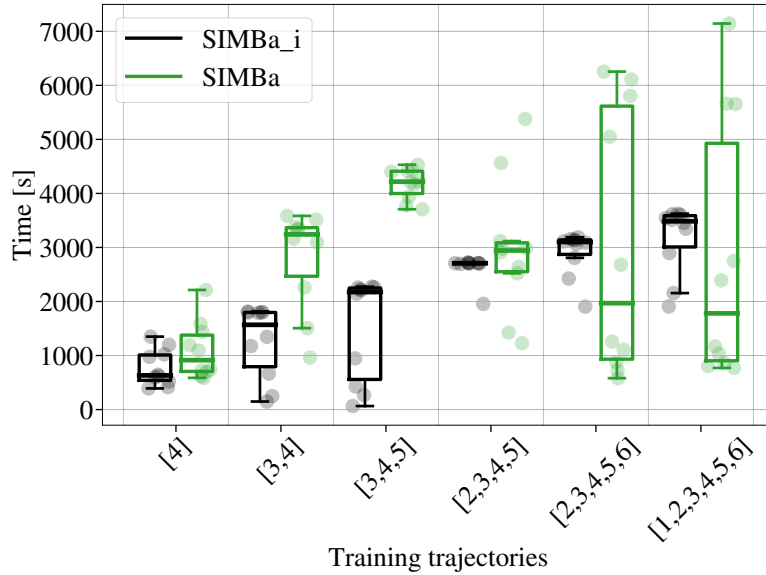


Figure 4.12: Time required for SIMBa to achieve its best validation error (in seconds), to compare with the corresponding total training times shown in Figure 4.11.

4.5 Discussion of SIMBa's potential and limitations

With the previous investigations showcasing SIMBa's performance on various system identification tasks, let us now briefly summarise how to best use the toolbox to maximize its performance and discuss potentially interesting extensions to it.

4.5.1 A summary of SIMBa's capabilities

Computational complexity. While SIMBa can achieve significant performance gains over traditional methods in many different settings, the associated computational burden can be significant. Indeed, even simple systems take several minutes to be fit, and this can grow significantly with longer training trajectories or when more epochs are required, as detailed in Section 4.4.7. Despite its ability to leverage out-of-the-box ML tools and GPUs, SIMBa is hence not well-suited for problems that demand fast solutions. In practice, traditional methods, which can be trained in a matter of seconds, are thus generally a good starting point. On the other hand, for cases where achieving the best performance is critical or when desired system properties need to be preserved, SIMBa can be extremely beneficial, as presented in Section 4.4.

During our investigations, we saw that randomness could play a significant role; running several instances of SIMBa might greatly improve its performance. Although initializing its state matrices with the one found by classical methods usually accelerates convergence, we also observed cases where randomly initialized versions achieve better final performance. Both options should hence be considered in practice.

Enforcing additional system properties. The free parameterizations from Propositions 3–6 can be leveraged to guarantee the stability of the identified model. While Proposition 6 can characterize any Schur matrix, it seems to be more sensitive to randomness and more numerically challenging than the other parametrizations. In practice, Propositions 3 and 4 should thus be preferred, and only the too-conservative Proposition 5 should be discarded and replaced by Proposition 6.

Hyperparameter tuning. In general, the provided default parameters perform well — they have indeed demonstrated robust performance across the variety of case studies analyzed in Section 4.4. To get the most out of SIMBa, one might however want to increase the number of training epochs, for example. On the other, the learning rate of 0.001 chosen in this work is sometimes slower than required. This default option proved to be robust across various tasks, but it might be possible to accelerate learning by taking larger parameter updates at each step on some problems.

4.5.2 Potential extensions

Augmenting the training procedure. Throughout Section 4.4, we did not strain to obtain the best performance on each case study but focused on fair comparisons between different methods under identical conditions. In practical applications, one should combine SIMBa — or any SI method — with data processing procedures, such as standardization, detrending, or filtering, to improve performance. In general, an interface with MATLAB’s SI toolbox, which comes with many useful helper functions, would be an interesting extension to SIMBa.

Apart from better integration of such existing tools, to facilitate training on nonconvex long-horizon objectives, *curriculum learning* could be adopted. In this framework, the training starts with the minimization of the one-step-ahead prediction error and then gradually increases the prediction horizon toward the desired one [149]. This could help SIMBa in the early stage of training, accelerating the first iterations by simplifying the problem — similar in spirit to the proposed initialization from the solution of classical SI methods — before leveraging the full power of automatic backpropagation for long-horizon optimization.

Enforcing additional system properties. Interestingly, Proposition 6 could also be used to generate affinely parametrized Schur matrices, similar to those examined in [335], for example. The generality of this free parametrization could thus allow SIMBa to guarantee stability while enforcing desired properties on A beyond specific sparsity patterns.

Introducing nonlinearities. In parallel with these efforts to enforce additional system properties, including nonlinearities may be crucial for some applications. Indeed, linear models might not be flexible enough to fit more complex systems. Thanks to the AD backbone used by SIMBa, NNs can be seamlessly added on top of the linear model, learning patterns that are not well-captured by the linear part. This would lead to models with an architecture similar to the PCNNs discussed in Chapter 2. Otherwise, inspired by the SINDy toolbox [336], if the class of nonlinearities impacting the dynamics are known, one could extend the state description to $f(x) \in \mathbb{R}^{n'}$, for example, including polynomials like $f(x) = [x^\top, (x^2)^\top]^\top$, and then fit a linear model of the form $x_{k+1} = Af(x_k) + Bu(k)$. Finally, a more cumbersome approach would be to discard the linear framework altogether, write custom dynamics, and then leverage automatic backpropagation as proposed herein to find the required parameters, similar to what is proposed in Section 4.6 for irreversible port-Hamiltonian systems.

Integration in Koopman-based methods. To conclude this discussion, we want to point out a potential link to Koopman-based approaches like [331], where traditional SI methods were used to identify linear models in the corresponding lifted space. Thanks to SIMBa’s construction relying on unconstrained GD, the lifting functions could also be learned simultaneously with the lifted linear model, similar in spirit to [304–306], potentially improving the accuracy of the end-to-end pipeline without jeopardizing stability.

4.6 A nonlinear extension: Physically Consistent Neural ODEs

This section proposes an extension of SIMBa to identify models respecting the laws of thermodynamics. To that end, we leverage nonlinear Irreversible Port-Hamiltonian (IPH) dynamics throughout this section, allowing us to train NNs without jeopardizing the first and second laws of thermodynamics.

4.6.1 The need for physically consistent neural ODEs

Simultaneously to the development of PiNNs to encode prior knowledge in NNs discussed in Section 2.1, higher-level connections between NNs and dynamical systems have also been studied. It shows that some classes of NNs can be interpreted as discretized dynamical systems [337]. On the other hand, Neural Ordinary Differential Equations (NODEs) were proposed in [338], where inputs are transformed through a continuous-time ODE embedding trainable parameters. In other words, NODEs learn the parameters of an ODE to fit the data, making them particularly suitable to model complex dynamical systems [38, 339]. Furthermore, their interpretation as ODEs allows one to borrow tools from dynamical system theory to analyze their properties [340–342]. However, similarly to classical NNs and PiNNs (see Section 2.1), NODEs can be physically inconsistent in general.

As one possible countermeasure to this brittleness, this section proposes **Physically Consistent NODEs (PC-NODEs)**. They leverage IPH dynamics instead of linear models and are trained like NODEs, which corresponds to the learning procedure of SIMBa. **Thanks to the IPH formulation, we can guarantee that PC-NODEs respect the first and second laws of thermodynamics at all times and by construction**, solving the issue of physically inconsistent NODEs for thermodynamic systems. Moreover, unlike black-box NNs, since they rely on the same AD backbone as SIMBa, PC-NODEs allow us to embed desired structural properties into the trainable parameters of the IPH model. This guarantees that the required skew-symmetry of the interconnection matrix and prescribed sparsity patterns, for example, are satisfied.

Thanks to their modularity IPH models characterize many multi-physics systems, including thermodynamic, mechanical, chemical, or electrical systems [343, 344]. Furthermore, identifying system dynamics in the IPH form provides several benefits, as one can then design stabilizing controllers and scale to distributed systems via interconnection with other passive port-Hamiltonian systems [343]. To showcase the flexibility of the proposed PC-NODEs, we demonstrate how they can be leveraged to model thermal building dynamics and the dynamics of a simulated gas-piston system.

Remark 26. *PC-NODEs can be seen as both an extension of SIMBa (Sections 4.2–4.5) to a specific case study or as a particular application of the PCNNs proposed in Chapter 2, which also respect the laws of thermodynamics. Indeed, replacing the linear physics-inspired module of PCNNs used throughout Section 2.3 by the IPH framework described in this section would lead to the same model structure, with an NN running in parallel with the physics-grounded*

module. However, PC-NODEs do not require separation of the inputs going through the NN and the physical module, as detailed in (4.19).

4.6.2 Learning Irreversible port-Hamiltonian dynamics with PC-NODEs

An IPH system is described as [343, 345]:

$$\dot{x} = R\left(x, \frac{\partial H(x)}{\partial x}, \frac{\partial S(x)}{\partial x}\right) J \frac{\partial H(x)}{\partial x} + g\left(x, \frac{\partial H(x)}{\partial x}\right) u + W\left(x, \frac{\partial H(x)}{\partial x}\right), \quad (4.19)$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ the control input, and the different functions and matrices satisfy the following properties:¹⁴

- P1. the Hamiltonian function H and the entropy function S are maps from $\mathcal{C}^\infty(\mathbb{R}^n)$ to \mathbb{R} ;
- P2. the interconnection matrix $J \in \mathbb{R}^{n \times n}$ is constant and skew-symmetric;
- P3. the real function R is defined as

$$R\left(x, \frac{\partial H}{\partial x}, \frac{\partial S}{\partial x}\right) = \gamma\left(x, \frac{\partial H}{\partial x}\right) \{S, H\}_J, \quad (4.20)$$

where $\gamma \geq 0$ is a nonnegative function of the states and co-states of the system;

- P4. the two vector fields W and g satisfy $W(x, \frac{\partial H}{\partial x}) \in \mathbb{R}^n$ and $g(x, \frac{\partial H}{\partial x}) \in \mathbb{R}^{n \times m}$.

We have used the **blue color** to denote functions that can be parameterized — for example, using NNs — and identified from data as described below, giving rise to the proposed PC-NODEs. As long as the trainable parameters respect the constraints and properties listed above, the learned model will obey the first and second laws of thermodynamics *by construction*. Indeed, by the skew-symmetry of J (P2), setting $W = u \equiv 0$, we have

$$\frac{dH}{dt} = \frac{\partial H^\top}{\partial x} \dot{x} = \frac{\partial H^\top}{\partial x} \left[R J \frac{\partial H}{\partial x} \right] \stackrel{(P3)}{=} R \left[\frac{\partial H^\top}{\partial x} J \frac{\partial H}{\partial x} \right] = 0, \quad (4.21)$$

which proves the conservation of energy when no input or disturbance affects the system. Similarly, we can show the irreversible creation of entropy in the system as follows [345]:

$$\frac{dS}{dt} = \frac{\partial S^\top}{\partial x} \dot{x} = \frac{\partial S^\top}{\partial x} \left[R J \frac{\partial H}{\partial x} \right] \stackrel{(P3)}{=} R \left[\frac{\partial S^\top}{\partial x} J \frac{\partial H}{\partial x} \right] \stackrel{(4.20)}{=} \gamma\left(x, \frac{\partial H}{\partial x}\right) \{S, H\}_J^2 \geq 0,$$

as long as $\gamma \geq 0$ (P3) when $W = u \equiv 0$.

Training PC-NODEs

Several NODE training procedures have been proposed in the literature relying on the adjoint sensitivity method [338] or AD [346], for example. In this work, as in [337], we first discretize

¹⁴To have concise notation throughout the section, the dependence on x and partial derivatives is dropped when it is clear from the context.

PC-NODE (4.19) using the forward-Euler method with sampling period $h > 0$, leading to

$$x_{i+1} = x_i + h \left(\textcolor{blue}{R}\textcolor{blue}{J} \frac{\partial H(x_i)}{\partial x_i} + \textcolor{blue}{W} + \textcolor{blue}{g}u_i \right), \quad (4.22)$$

where x_i and x_{i+1} represent the current and next state, respectively. In practice, the step-size h is chosen sufficiently small to interpret the states in (4.22) as a sampled version of the state $x(t)$ of system (4.19).

We then assume to have access to a dataset of N sampled full-state trajectories

$$\mathcal{D} := \left\{ (u(0)^{(s)}, x(0)^{(s)}), (u(1)^{(s)}, x(1)^{(s)}), \dots, (u(\ell)^{(s)}, x(\ell)^{(s)}) \right\}_{s=1}^N,$$

where ℓ is the total number of time steps for each trajectory of measured states x and inputs u . Similarly to SIMBa in (4.15), we train system (4.22) to minimize

$$\begin{aligned} \min_{\textcolor{blue}{R}, \textcolor{blue}{J}, \textcolor{blue}{H}, \textcolor{blue}{W}, \textcolor{blue}{g}} \quad & \frac{1}{|Z|} \sum_{s \in Z} \left[\frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(x(i)^{(s)}, x_i^{(s)}) \right] \\ \text{s.t.} \quad & x_{i+1}^{(s)} = x_i^{(s)} + h \left(\textcolor{blue}{R}\textcolor{blue}{J} \frac{\partial H(x_i^{(s)})}{\partial x_i^{(s)}} + \textcolor{blue}{W} + \textcolor{blue}{g}u(i)^{(s)} \right) \\ & x_0^{(s)} = x(0)^{(s)}, \end{aligned} \quad (4.23)$$

where Z is a batch of data randomly sampled from the training data set. While we optimize the MSE $\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|_2^2$ in this section, this can easily be replaced by other loss functions.

As for SIMBa, we implement the proposed PC-NODEs using PyTorch [140], which allows us to easily propagate the inputs through the NODE and then rely on automatic BPTT [347] to run GD on the trainable parameters. However, in general, it does not allow one to introduce constraints on the parameters directly. In particular, it cannot guarantee that either $\textcolor{blue}{J}$ or $\textcolor{blue}{R}$ satisfy P2 or P3, respectively. Nonetheless, as exemplified with two case studies in the next section, prior knowledge of the systems often allows one to design free parametrizations of $\textcolor{blue}{J}$ and $\textcolor{blue}{R}$ satisfying these conditions *by design* and hence allowing for unconstrained GD, similarly to what was proposed in Sections 4.2–4.5.

Remark 27. Besides modifying the loss function \mathcal{L} , one can also introduce weighted penalty terms in equation (4.23), to promote sparse solutions with $\|\textcolor{blue}{J}\|_1$, for example.

4.6.3 Irreversible port-Hamiltonian model formulations

To demonstrate the variety of systems that can be represented with IPH dynamics, let us detail how to model the thermal dynamics of a building and a gas-piston system and how to design the [blue parameters](#) to ensure properties P1–P4 are respected.¹⁵

¹⁵The code and data can be found on <https://gitlab.nccr-automation.ch/loris.dinatale/pc-node>.

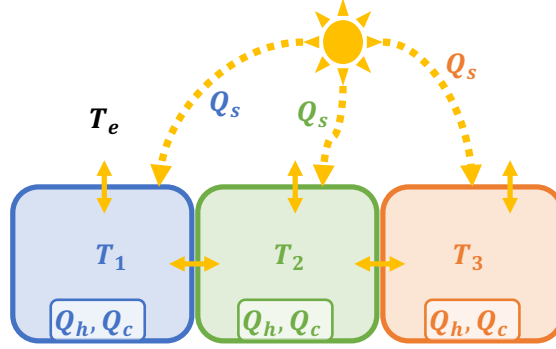


Figure 4.13: A pictorial description of the thermal behavior of the three zones in UMAR, where yellow arrows represent energy flows.

Credit assignment. The developments in this section stem from a collaboration between Muhammad Zakwan and the author of this thesis in [58]. Although they were spearheaded by Muhammad Zakwan, they are reported here for completeness since they will be used in Section 4.6.4, but the proofs are deferred to the appendix.

Thermal building dynamics

The thermal dynamics of a building can be seen as n connected thermal zones exchanging energy among themselves and with the outside. In this section, we assume that they are additionally impacted by various heat gains stemming from heating or cooling operations and solar irradiation. Here we consider the thermal dynamics of UMAR (Section 2.4.1), which can be pictorially represented as depicted in Figure 4.13.

Inspired by the IPH formulation of heat exchangers [343], we model the entropy $S \in \mathbb{R}^n$ in each zone as

$$\dot{S} = \tilde{J}(T) \frac{\partial H(S)}{\partial S} + B_e(T) T_e + \begin{bmatrix} B_s & B_h & B_c \end{bmatrix} \begin{bmatrix} Q_s \\ Q_h \\ Q_c \end{bmatrix}, \quad (4.24)$$

where $T \in \mathbb{R}^n$ represents the temperature in each zone. For clarity, we separated the different external inputs u , with $T_e \in \mathbb{R}$ corresponding to the ambient temperature, and $Q_s, Q_h, Q_c \in \mathbb{R}^n$ to solar, heating, and cooling gains for each zone, respectively. B_s, B_h , and B_c are $n \times n$ diagonal matrices gathering trainable scaling parameters reflecting the impact of these gains on the entropy of each zone. $B_e(T) \in \mathbb{R}^n$ is a diagonal matrix modeling the heat losses to the outside, with entries

$$B_e(T)_{zz} = \lambda_{ze} \frac{(T_e - T_i)}{(T_i T_e)}$$

for each zone z , where $\{\lambda_{ze}\}_{z=1}^n$ are the trainable parameters. Finally, the skew-symmetric

4.6 A nonlinear extension: Physically Consistent Neural ODEs

matrix $\tilde{J}(T) \in \mathbb{R}^{z \times z}$, lumping together R and J in this case, is parametrized as

$$\tilde{J}_{ij}(T) = -\tilde{J}_{ji}(T) = \begin{bmatrix} 0 & \lambda_{12} \frac{(T_2 - T_1)}{(T_1 T_2)} & 0 \\ \lambda_{12} \frac{(T_1 - T_2)}{(T_1 T_2)} & 0 & \lambda_{23} \frac{(T_3 - T_2)}{(T_2 T_3)} \\ 0 & \lambda_{23} \frac{(T_2 - T_3)}{(T_2 T_3)} & 0 \end{bmatrix}.$$

This reflects the topology of the building, i.e., the fact that Zone 1 and 3 are only exchanging energy with the adjacent Zone 2 and not among each other.

Interestingly, by the definition of entropy, and recalling that the Hamiltonian H represents the energy of the system, we have $\frac{\partial H(S)}{\partial S} = T$. Hence, there is no need to parametrize the partial derivatives of the Hamiltonian function in this case since they can be computed explicitly from the state of the system — assuming a constant volume for each zone — as

$$T_{i+1} = \left[\exp \left(\frac{S(t_f) - S(t_i)}{mc} \right) \right] T_i,$$

as detailed in Appendix B.8.

Proposition 7 (Consistency and monotonicity). *PC-NODE (4.24) is consistent with the first and second laws of thermodynamics and monotonic with respect to all inputs, i.e., T_e , Q_s , Q_h , and Q_c , if the learned parameters satisfy*

$$B_s, B_h, B_c \geq 0, \text{ and } \lambda_{zy}, \lambda_{ze} \in \mathbb{R}_+, \forall z, y = 1, \dots, n.$$

Proof. See Appendix B.9 for a sketch of the proof and [344] for more details. \square

Remark 28. *The dependence of \tilde{J} on T in (4.24) violates property P2 stating it should be a constant matrix. While state-dependent connection matrices break the consistency of the system with the first and second laws of thermodynamics in general [343], we show that PC-NODE (4.24) remains consistent in the proof of Proposition 7. The key idea is to decompose \tilde{J} into a sum of terms with constant interconnection matrices.*

Remark 29. *Exploiting the linearity of the PC-NODE (4.24), one can show that it is almost equivalent to well-known RC dynamics (Appendix A.1.1), which model the energy of each zone instead of their entropy. This relation holds since both quantities are linked by definition as $dS = \frac{dH}{T} \iff dH = T dS$. Multiplying (4.24) by the temperature of each zone elementwise, one can hence recover an energy model of the building. The only difference with RC modeling will then be that the training parameters in the diagonal matrices $B_s(T)$, $B_h(T)$, and $B_c(T)$ depend on the corresponding zone temperatures instead of being constant. Since the zone temperatures are however approximately constant (in Kelvin) in buildings, IPH building models are indeed similar to classical RC ones.*

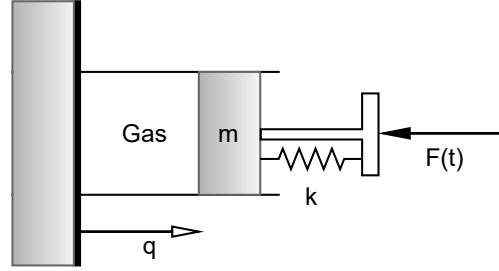


Figure 4.14: Sketch of the gas piston system.

Gas Piston system

Consider a typical gas piston system where the piston is subject to friction, influenced by an external force $F(t) = u$, and its elasticity is modeled by a spring, as sketched in Figure 4.14. Let us define the state of the system as $x = [S, V, q, p]^\top$, where S is the entropy and V the volume of the gas, and q and p are the position and momentum of the piston, respectively. Inspired by [345], the system can be described by the following nonlinear IPH dynamics:

$$\dot{x} = \left[R \left(x, \frac{\partial S}{\partial x}, \frac{\partial H(x)}{\partial x} \right) J_0 + J_1 \right] \frac{\partial H(x)}{\partial x} + G^\top u, \quad (4.25)$$

with

$$J_0 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}, \quad J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha \\ 0 & 0 & 0 & \beta \\ 0 & -\alpha & -\beta & 0 \end{bmatrix}, \quad \frac{\partial H(x)}{\partial x} = \begin{bmatrix} T \\ -P \\ Kq \\ v \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad R = \frac{\mu v}{T},$$

where T is the temperature and P the pressure of the gas, K the spring constant, and $v = \frac{p}{m}$ represents the speed of the piston with mass m and friction coefficient μ .

Since the entropy is a state of the system, $\frac{\partial S}{\partial x} = [1, 0, 0, 0]^\top$, which implies that P2 becomes

$$R \left(x, \frac{\partial S}{\partial x}, \frac{\partial H(x)}{\partial x} \right) = \gamma \left(x, \frac{\partial H(x)}{\partial x} \right) \frac{\partial S}{\partial x}^\top J_0 \frac{\partial H(x)}{\partial x} = \gamma \left(x, \frac{\partial H(x)}{\partial x} \right) \frac{\partial H(x)}{\partial p}. \quad (4.26)$$

The function R is thus well-defined and can be derived from γ and H . To showcase the flexibility of the proposed PC-NODEs, we assume the Hamiltonian to be unknown and parametrize it as a single-layer NN with the form

$$H(x; \theta) = \log [\cosh(Kx + b)]^\top \mathbb{1}_4, \quad (4.27)$$

where $\theta = \{K, b\}$. Such an architecture is chosen for its elegance because it allows us to

compute the required partial derivatives in closed form [342]:

$$\frac{\partial H(x; \theta)}{\partial x} = K^\top \tanh(Kx + b). \quad (4.28)$$

We parametrize γ as a single layer NN $\gamma: \mathbb{R}^8 \rightarrow \mathbb{R}^+$, where positivity is obtained by feeding the output through a sigmoid function, which is sufficient to satisfy P2. Finally, we assume the sparsity pattern of J_1 to be known but not its parameters $\{\alpha, \beta\}$ to demonstrate how prior knowledge might be incorporated into the learning process.¹⁶

Remark 30. *Although PC-NODE (4.25) is slightly different from the generic representation in (4.19), the key system properties are still conserved. Indeed, one can always decompose the product between R and J in a sum of products without violating the first and second laws of thermodynamics as long as each term in the sum respects condition (4.20) and the skew-symmetry of J . See the proof of Proposition 7 in Appendix B.9 for more details.*

Remark 31. *In practice, any NN can be chosen to parametrize $H(x)$ since the gradients $\frac{\partial H(x)}{\partial x}$ are readily available through backpropagation. Furthermore, if one is not interested in the Hamiltonian itself—which is not required in (4.22)—, it is also possible to directly parametrize the gradient flow with an NN, bypassing the need for expensive backpropagation operations.*

Remark 32. *Since we assume no thermal exchanges between the gas and the ambient air, the gas entropy can never decrease according to the second law of thermodynamics.*

4.6.4 Applications and results

Credit assignment. The software and numerical experiments in this section stem from a collaboration between Muhammad Zakwan and the author of this thesis in [58], although they were spearheaded by the author of this thesis.

Thermal dynamics of UMAR

For this first application, we use the data described in Section 2.4.1 and analyze our model's accuracy over more than 750 sequences of three days of validation data. Note that Q_s in (4.24) represents solar gains in each zone, derived from irradiation measurements on a flat surface as detailed in Appendix A.3. Averaged over the three zones and all the time series, the MAE propagation over the 72 h horizon is depicted in Figure 4.15, computed as in (2.39). Since PC-NODE (4.24) is linear, we also plot the performance of a classical linear ARX model with 12 lag terms for reference, where the number of lags was tuned empirically and the parameters fit to the data through LS identification, similarly to [348], for example.

As can be readily seen, thanks to the underlying physics captured by the Hamiltonian framework, the PC-NODE can fit the data significantly better, especially over long horizons. Indeed,

¹⁶In the true system, α is the area of the piston and $\beta = 1$.

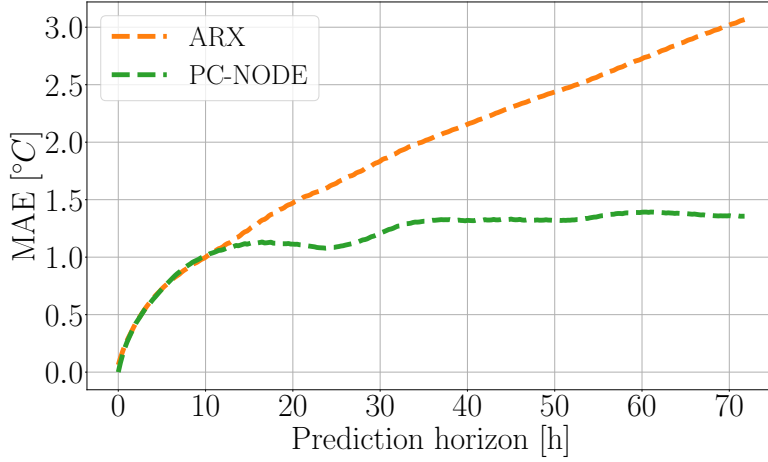


Figure 4.15: MAE of the ARX model and PC-NODE over the prediction horizon averaged over the three zones and the validation time series.

it seems to be less prone to compounding errors: it improves the accuracy by 38.9% compared to the ARX on average over the entire prediction length, but this proportion rises to 55.8% at the end of the 72 h-long horizon.

To provide a visual comparison of the behavior of both models, we plotted their temperature predictions over a sampled 72 h-long trajectory in August 2021 in Figure 4.16, with the ground truth measurements for reference. This figure hints that the ARX model is more sensitive to the various external gains, tending to overestimate their impact. This can, for example, be observed towards the end of the horizon, just before noon: when the sun rises, increasing the temperature of the building, the ARX cannot accurately capture this behavior, contrary to the PC-NODE. While only one sampled trajectory is presented here for brevity, these effects generally hold across the validation data and explain the better performance of the PC-NODE in general observed in Figure 4.15.

Remark 33. *The performance of the PC-NODE in Figure 4.15 can be qualitatively compared to the one of PCNNs in Figure 2.14. Indeed, they all model UMAR over a three-day-long horizon from three years of data. However, the PC-NODE receives engineered solar gains as input while PCNNs in Chapter 2 process them through their black-box module. Since the both achieved similar performance, this confirms the ability of NNs to accurately capture the impact of the sun in the black-box module of PCNNs.*

Gas piston system

For the gas piston system, we generated a synthetic dataset of 10'000 samples from system (4.25) using the `odeint` framework from `scipy` [349]. We simulated it from $T(0) = 290$ K and $x(0) = [0, 0.001, 0.3, 0]^T$ with $m = 5$ kg, $\alpha = 0.033$ m², $\beta = 1$, $\mu = 1$, and $K = 10$ N m⁻¹, and a sampling time of $h = 0.01$ s. The gas temperature over the horizon has been computed as

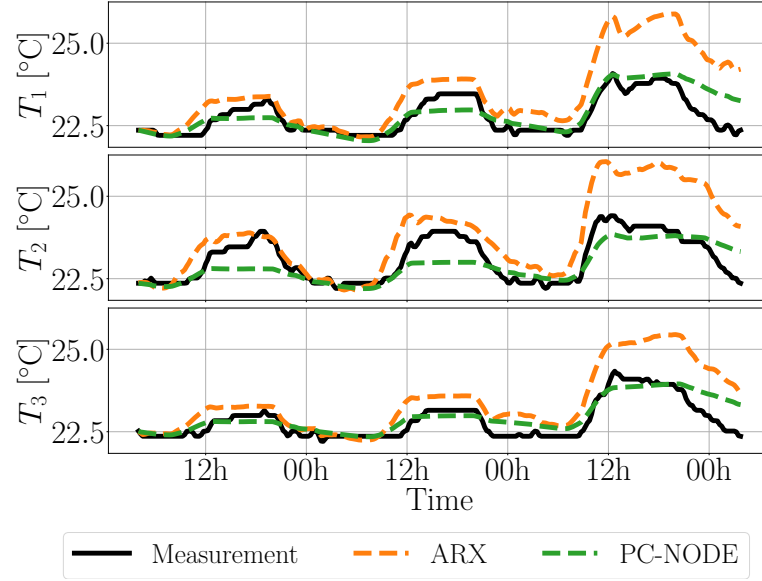


Figure 4.16: August 18–20, 2021: Temperature predictions of the PC-NODE and ARX model on a sampled validation trajectory, compared to the true measurements.

presented in Appendix B.8 and the pressure P was derived from the ideal gas law $PV = nRT$.

The data was split into 40 trajectories of 250 steps and we added Gaussian noise $\mathcal{N}(0, 0.2\sigma_d)$ on each dimension d of the state — where σ_d corresponds to the standard deviation of the d th dimension of x — before training the models. To ease the training of the NNs used in PC-NODE (4.25), the data was additionally normalized between 0.1 and 0.9 for each dimension.

Despite not having access to the true Hamiltonian function and learning it as an NN from data, and even in the presence of white noise, PC-NODE (4.25) accurately recovered the position of the system, as pictured in Figure 4.17 (bottom) for four sampled trajectories. However, a vanilla NODE, i.e., $\dot{x} = f_\theta(x)$ [338], where f_θ is an NN with two hidden layers of 32 neurons each, is also able to fit this data very well. On the other hand, the evolution of the entropy is more challenging to capture, as pictured on the top of Figure 4.17, where we scaled it and removed the noisy data for clarity. In that case, the PC-NODE clearly outperforms the vanilla NODE thanks to its embedded physical consistency. Remarkably, the NODE sometimes predicts a decrease in entropy, which is inconsistent with the underlying physics (see Remark 32) and does not happen with the PC-NODE.

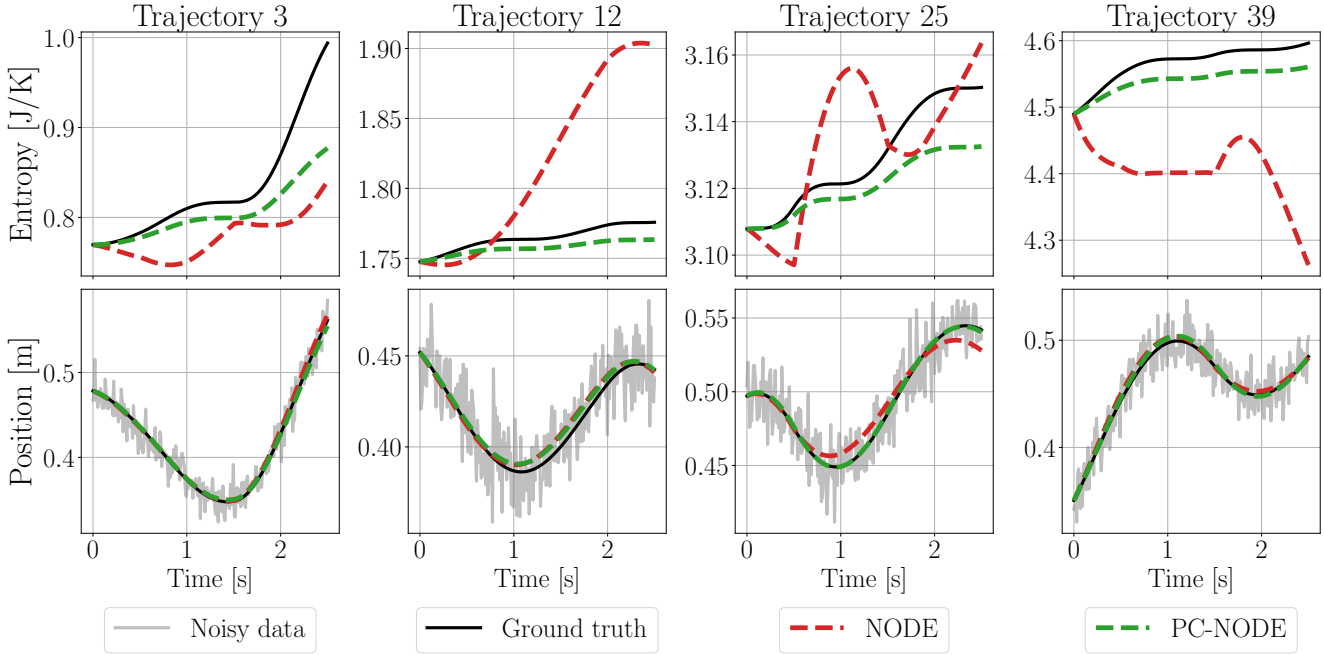


Figure 4.17: Sampled trajectories of the piston position and the entropy of the gas ($\times 10^3$) over time for a classical NODE (red) and the proposed PC-NODE (green). The noisy data can be found in shaded gray for the position and the ground truth in black.

4.7 Conclusion and outlook

In this chapter, we proposed to leverage automatic differentiation to help scale traditional system identification procedures.

SIMBa. We first presented the [open-source SIMBa toolbox for the identification of stable discrete-time linear state-space models](#). Relying on novel LMI-based free parametrizations of Schur matrices to ensure the stability of the identified model despite enforcing desired properties, such as sparsity, we showed how SIMBa outperforms traditional SI methods, and often by more than 25%. Throughout our experiments, this performance gap increased significantly when sparsity patterns or known values of the state-space matrices to identify needed to be respected, in which case MATLAB often failed to recover meaningful solutions while SIMBa still achieved state-of-the-art accuracy. Furthermore, on a real-world robot data set, SIMBa often improved state-of-the-art input-state identification methods by more than 70%, with the gap widening as more and more training data was made available.

On the other hand, the consistent and significant performance gains observed across the different experiments proposed in this work come with a large computational burden. SIMBa indeed incurred training times ranging from several minutes to two hours in our experiments. This might be mitigated in practice by reducing the number of training epochs, augmenting the step size, or initializing SIMBa with matrices known to perform well. However, the latter

does not necessarily improve the final performance.

PC-NODEs. We then tackled the more complex case where models need to respect the first and second laws of thermodynamics. To that end, we extended SIMBa beyond linear models and leveraged IPH dynamics, leading to PC-NODEs. The latter again showed strong performance gains over standard methods while respecting the underlying physical laws. Similarly to what was observed in the rest of this thesis, enforcing desired properties did not come at the cost of accuracy, with PC-NODEs outperforming vanilla NODEs.

Link with Chapter 2. Thanks to their AD backbone, both SIMBa and PC-NODEs could train NNs in parallel with their main module to capture unmodeled dynamics. Apart from minor modification, this would allow one to recover the PCNN architecture discussed in Chapter 2. In other words, grounding NNs with prior knowledge (PCNNs) or augmenting traditional SI methods with NNs — leveraging the backpropagation-based training procedure introduced in this chapter — would lead to similar solutions.

Outlook. As detailed in Section 4.5, in future works, it would be interesting to analyze the theoretical implications of SIMBa and the potential links to traditional SI methods. In a similar vein, SIMBa’s potential to be incorporated in Koopman-based approaches with stability guarantees is worth investigating. Leveraging the seamless capacity of PyTorch to incorporate various differentiable nonlinear functions, as proposed in Section 4.6, for example, **we postulate that SIMBa has the potential to serve as the foundation for a general tool for knowledge-grounded structured nonlinear system identification.**

5 Concluding remarks

The escalating impact of climate change is compelling the world into a global energy transition. Addressing the resulting challenges necessitates the introduction of innovative and revolutionary technologies to the market. Throughout this thesis, we postulated that Neural Networks (NNs) and Machine Learning (ML) are poised to assume an increasingly crucial role in this fight — as well as in many other applications.

Grounding Neural Networks with established expert knowledge

In general, relying on brittle vanilla *black-box* NNs for real-world deployment is not advisable, especially for systems as critical as energy infrastructures. There is thus a widespread need to enforce desired properties on NNs and guarantee they behave adequately under diverse circumstances. Consequently, Chapters 2 and 3 introduced various approaches to ensure NNs adhere to the domain knowledge accumulated over years of engineering, both for modeling and control purposes.

First, the numerical experiments in Chapter 2 hinted that the proposed Physically Consistent NNs (PCNNs) can simultaneously achieve state-of-the-art performance and compliance with the underlying laws of thermodynamics for building thermal modeling applications. Second, the investigations in Chapter 3 pointed to the potential of model-free Deep Reinforcement Learning (DRL) agents to satisfy all the requirements of *ideal* building controllers — and to how fusing expert knowledge in NN control policies might be key in attaining this objective.

While these two chapters focus primarily on one single case study building, the majority of insights and conclusions are likely to extrapolate to other buildings and domains. This hence contributes to the expanding body of literature showing how constraining NNs according to expert knowledge can be extremely beneficial — if not necessary — to facilitate their widespread adoption in real-world applications.

Leveraging Machine Learning tools for traditional system identification methods

Rather than focusing on grounding NNs in prior system knowledge, which might perform suboptimally due to their complexity, Chapter 4 suggested leveraging ML tools to help identify traditional models. In addition to enhancing *interpretability*, which makes them more reliable than NNs in practical scenarios, these knowledge-grounded models can ensure that desired system properties are respected at all times. However, calibrating such models with traditional System Identification (SI) methods is generally challenging. To mitigate this issue, Chapter 4 demonstrated the benefits of leveraging backpropagation — the core of NN training — for this task.

In the case of linear systems, this led to the development of the open-source SIMBa toolbox. It can guarantee the stability of the identified model and incorporate desired properties on the state-space matrices, such as sparsity patterns, while consistently and often significantly outperforming traditional SI approaches. To showcase how SIMBa could be extended to more complex case studies beyond linear systems, we then detailed a nonlinear extension relying on irreversible port-Hamiltonian dynamics. It could guarantee compliance with the first and second law of thermodynamics *by design* without compromising performance.

Notably, thanks to the automatic differentiation backbone of the proposed training procedure, NNs could seamlessly be learned on top of the knowledge-infused dynamics to capture unmodeled aspects. In this particular scenario, one would recover the PCNN architecture from Chapter 2, which instead introduces a physics-inspired module in parallel to NNs to ensure adherence to specific properties of the modeled system.

The positive impact of merging expert knowledge and Machine Learning

In summary, our investigations hint at the **advantages of fusing prior system knowledge into *any* model or controller. When learning is involved, we argue one should restrict the search space to plausible or desired solutions.** This might introduce sometimes significant conservatism but will ensure the solution behaves as expected. Notably, when certain aspects of the problem are challenging to quantify, NNs might be leveraged to learn them from data, typically running in parallel with a knowledge-based module.

Interestingly, prior knowledge integration did not induce significant performance drops throughout our analyses. On the contrary, it generally improved the quality of the solution, showing how constraining learning methods can help find well-performing solutions consistent with desired properties. To conclude, we hope our investigations will **lay the groundwork for universal and scalable methods fusing prior knowledge and ML tools for modeling and controlling diverse systems.**

A Appendices - Chapter 2

A.1 Resistance-capacitance building model

A.1.1 General resistance-capacitance models

In general, we can describe the thermal dynamics of a thermal zone with the following ODE:

$$C \frac{dT}{dt} = \frac{dQ^{heat}}{dt} + \frac{dQ^{irr}}{dt} + \frac{dQ^{occ}}{dt} + \sum \frac{dQ^{cond}}{dt} + \sum \frac{dQ^{conv}}{dt},$$

where T is the temperature, C the heat capacitance of the air mass, and the various heat flows Q respectively represent the impact of the heating/cooling system (negative values represent cooling energy), the solar irradiation, the occupants, heat conduction, and heat convection, where both sums are taken over the number of surfaces adjacent to the measured volume of air.

In this work, we lump conductive and convective transfers together and split them into two heat transfers: one to represent transfers to the neighboring zone — assuming there is only one for ease of exposition — and the other to gather losses to the environment. Both are proportional to the corresponding temperature gradient between the zone temperature and the neighboring zone or the outside temperature, respectively. Additionally, we process the horizontal solar irradiation data to reflect the solar gains through the windows as presented in Appendix A.3 and group the heat gains from the occupants and other unmodeled effects in Q^{rest} , scaled by a parameter η .

Altogether, we can rewrite the thermal dynamics as:

$$\frac{dT}{dt} = \frac{1}{C} \frac{dQ^{heat}}{dt} + \frac{\epsilon}{C} \frac{dQ^{irr}}{dt} + \frac{\eta}{C} \frac{dQ^{rest}}{dt} - \frac{1}{CR_{out}} \frac{d(T - T^{out})}{dt} - \frac{1}{CR_{neigh}} \frac{d(T - T^{neigh})}{dt}$$

with ϵ representing the lumped permissivity of the windows and exterior walls, R_{out} and R_{neigh} the thermal resistance of the walls adjacent to the outside, respectively the neighboring

Appendix A. Appendices - Chapter 2

zone, and T^{out} and T^{neigh} the temperature outside, respectively in the neighboring zone. We then discretize this ODE with the Euler forward method and the time step Δ_t , yielding:

$$T_{k+1} = T_k + \Delta_t * [\frac{1}{C}Q_k^{heat} + \frac{\epsilon}{C}Q_k^{irr} + \frac{\eta}{C}Q_k^{rest} - \frac{1}{CR_{out}}(T_k - T_k^{out}) - \frac{1}{CR_{neigh}}(T_k - T_k^{neigh})]$$

Grouping the constants together and defining new parameters a , b , c , e_1 and e_2 , we can reformulate it as follows:

$$T_{k+1} = T_k + aQ_k^{heat} - b(T_k - T_k^{out}) - c(T_k - T_k^{neigh}) + e_1Q_k^{irr} + e_2Q_k^{rest} \quad (A.1)$$

A.1.2 Single-zone linear model

In this work, to create a simple linear model to use as a comparison baseline to single-zone PCNNs, we assume no knowledge of the occupants and other heat gains and discard the corresponding term $e_2Q_k^{rest}$ in (A.1). Reordering the terms, we get:

$$T_{k+1} - T_k = \begin{bmatrix} Q_k^{heat} \\ -(T_k - T_k^{out}) \\ -(T_k - T_k^{neigh}) \\ Q_k^{irr} \end{bmatrix}^T \begin{bmatrix} a \\ b \\ c \\ e_1 \end{bmatrix}$$

$$\Delta T_{k+1} := y_k^T p,$$

where ΔT represents the temperature difference, y groups the factors influencing it, and p the unknown parameters. Doing this for every time step, we can create matrices of data, grouping all the temperature differences in matrix X and the external factors in Y :

$$\begin{bmatrix} \Delta T_1 \\ \vdots \\ \Delta T_N \end{bmatrix} = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} p$$

$$X := Y p$$

Finally, we can use Least Squares to identify the parameters:

$$Y^T X = Y^T Y p$$

$$p = (Y^T Y)^{-1} Y^T X.$$

A.2 Proofs of the main theoretical results

A.2.1 Proof of Proposition 1

The proof works by induction on i . Based on (2.17), we can immediately write, $\forall z, y \in \mathcal{B}$:

$$\frac{\partial T_{j+1}^z}{\partial T_j^y} = \begin{cases} 1 - b^z - \sum_{y \in \mathcal{N}(z)} c^{zy}, & \text{if } y = z, \\ c^{zy}, & \text{if } y \in \mathcal{N}(z), \\ 0, & \text{otherwise,} \end{cases} \quad (\text{A.2})$$

where we used the definition of ΔT in (2.9). By definition, if (2.19) and (2.20) hold, we hence get positive derivatives if $y = z$ or $y \in \mathcal{N}(z)$ and zeros for any other choice of y , satisfying (2.18) and completing the base case of the induction. Let us now assume that:

$$\frac{\partial T_h^x}{\partial T_j^y} \geq 0, \forall y, x \in \mathcal{B}, \forall j < h < i, \quad (\text{A.3})$$

with equality if and only if $y \notin \mathcal{N}^{(h-j)}(x)$, and show that the proposition holds for time step i . Since we know the temperature in zone z at time i is potentially impacted by the temperature in the entire building at the previous step, we can decompose the partial derivative of interest as follows:

$$\frac{\partial T_i^z}{\partial T_j^y} = \sum_{x \in \mathcal{B}} \frac{\partial T_i^z}{\partial T_{i-1}^x} \frac{\partial T_{i-1}^x}{\partial T_j^y}, \quad (\text{A.4})$$

for all $y, z \in \mathcal{B}$. Since (2.17) is time-invariant, we know that:

$$\frac{\partial T_i^z}{\partial T_{i-1}^x} = \frac{\partial T_{j+1}^z}{\partial T_j^x} \geq 0,$$

with equality if and only if $x \notin \mathcal{N}(z)$ by the base case of the induction (A.2) if (2.19) and (2.20) hold. Similarly, by the induction hypothesis (A.3), we know that:

$$\frac{\partial T_{i-1}^x}{\partial T_j^y} \geq 0, \forall y, x \in \mathcal{B},$$

with equality if and only if $y \notin \mathcal{N}^{(i-j-1)}(x)$. Putting the last two equations together, we see that:

$$\frac{\partial T_i^z}{\partial T_j^y} \geq 0,$$

with equality only if each term of the sum in Equation (A.4) is zero. By the previous arguments, this means $y \notin \mathcal{N}^{(i-j-1)}(x)$ or $x \notin \mathcal{N}(z)$ for all zones x . This is equivalent to say that there is no path from y to z in $(i-j)$ steps, i.e., $y \notin \mathcal{N}^{(i-j)}(z)$, which concludes the inductive step. \square

A.2.2 Proof of Proposition 2

We start by noticing that $\forall y \in \mathcal{B}$, (2.17) implies:

$$\frac{\partial T_{j+1}^y}{\partial u_j^y} = \frac{\partial T_{j+1}^y}{\partial g^y(u_j^y)} \frac{\partial g^y(u_j^y)}{\partial u_j^y} = \begin{cases} a_h^y \frac{\partial g^y(u_j^y)}{\partial u_j^y}, & \text{if } u_j^y > 0, \\ a_c^y \frac{\partial g^y(u_j^y)}{\partial u_j^y}, & \text{if } u_j^y < 0, \\ 0, & \text{otherwise,} \end{cases} \quad (\text{A.5})$$

$$\frac{\partial T_{j+1}^x}{\partial u_j^y} = \frac{\partial T_{j+1}^x}{\partial g^y(u_j^y)} \frac{\partial g^y(u_j^y)}{\partial u_j^y} = 0 \quad \forall x \in \mathcal{B}, x \neq y, \quad (\text{A.6})$$

$$\frac{\partial T_{j+1}^y}{\partial T_j^{\text{out}}} = b^y, \quad (\text{A.7})$$

Note that this proves that (2.21) and (2.22) hold for the case $i = j + 1$ as long as $a_h^y, a_c^y, b^y > 0$, $\forall y \in \mathcal{B}$, $\frac{\partial g(u)}{\partial u} > 0$, and $g(u) = 0$.

When $i > j + 1$, Proposition 1 implies that

$$\frac{\partial T_i^z}{\partial T_{j+1}^y} \geq 0, \quad \forall z, y \in \mathcal{B}, \forall 0 \leq j < i - 1, \quad (\text{A.8})$$

with equality if and only if $y \notin \mathcal{N}^{(i-j-1)}(z)$ if the conditions in (2.19) and (2.20) hold. Relying on the fact that the temperatures at time $k + i$ are potentially influenced by the temperatures in the whole building at time $j + 1$, we have:

$$\frac{\partial T_i^z}{\partial u_j^y} = \sum_{x \in \mathcal{B}} \frac{\partial T_i^z}{\partial T_{j+1}^x} \frac{\partial T_{j+1}^x}{\partial g^y(u_j^y)} \frac{\partial g^y(u_j^y)}{\partial u_j^y} \quad (\text{A.9})$$

$$= \frac{\partial T_i^z}{\partial T_{j+1}^y} \frac{\partial T_{j+1}^y}{\partial g^y(u_j^y)} \frac{\partial g^y(u_j^y)}{\partial u_j^y} \geq 0, \quad (\text{A.10})$$

where the second equality follows from (A.6) and the inequality holds as long as (2.19) and (2.20) are respected and $a_h^y, a_c^y > 0$, $\forall y \in \mathcal{B}$, by Proposition 1. Furthermore, equality is only reached if $y \notin \mathcal{N}^{(i-j-1)}(z)$.

Similarly, we have:

$$\frac{\partial T_i^z}{\partial T_j^{\text{out}}} = \sum_{y \in \mathcal{B}} \frac{\partial T_i^z}{\partial T_{j+1}^y} \frac{\partial T_{j+1}^y}{\partial T_j^{\text{out}}} = \sum_{y \in \mathcal{B}} \frac{\partial T_i^z}{\partial T_{j+1}^y} b^y > 0, \quad (\text{A.11})$$

where the strict inequality is respected as long as $b^y > 0$, $\forall y \in \mathcal{B}$. Indeed, since $z \in \mathcal{N}(z)$ by definition, Proposition 1 then implies that at least one of the terms in the sum is strictly positive, while the others are nonnegative. \square

A.3 Solar irradiation preprocessing

To compute the solar irradiation on the windows of a thermal zone z from the measured irradiation on a horizontal surface Q^{sun} , we rely on the altitude and azimuth angles, respectively ϕ and θ , of the sun. The former captures the elevation of the sun above the horizon while the latter represents its deviation from the north, in the clockwise direction.

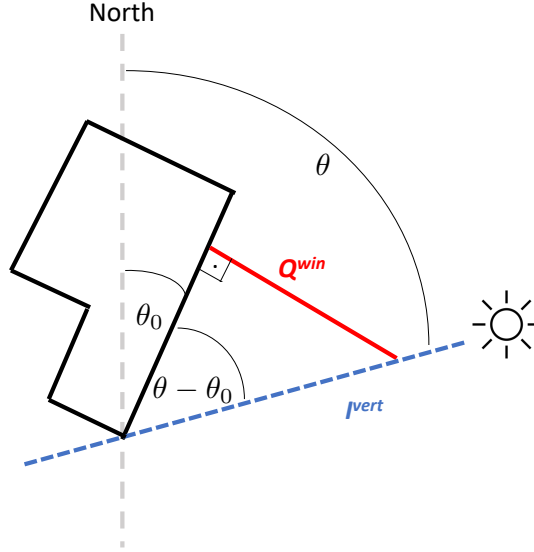


Figure A.1: Sketch of the azimuth angles used to compute the solar irradiation on the windows of a building from the irradiation on a fixed vertical surface.

First, using the altitude of the sun and basic trigonometry, one can easily show that the measured irradiation on a horizontal surface corresponds to $Q^{sun} = I \sin \phi$, where I is the global solar irradiation. Similarly, we know that the irradiation on a vertical surface following the sun, i.e., tracking its azimuth angle to stay perpendicular to the incoming rays, can be computed as $I^{vert} = I \cos \phi$, or

$$I^{vert} = Q^{sun} \frac{\cos \phi}{\sin \phi}. \quad (\text{A.12})$$

Since building facades and windows have a fixed orientation in practice and do not follow the sun azimuth, we again use basic trigonometry to compute the irradiation on a north-south aligned surface facing east as $I^{vert} \sin \theta$. Finally, if the facade is not exactly facing east, we also need to account for its own "azimuth" θ_0 , i.e., how much it is rotated clockwise starting from an east-facing position (Figure A.1), which leads to:

$$Q^{win} = I^{vert} \sin(\theta - \theta_0) = Q^{sun} \frac{\cos \phi}{\sin \phi} \sin(\theta - \theta_0). \quad (\text{A.13})$$

Once this has been done for each zone z , we can populate the required vector \mathbf{Q}^{win} used by gray-box architectures in this work.

As one can readily be observed, this processing only requires access to the elevation and azimuth angles of the sun, and to the orientation of the facade of interest. Furthermore, both solar angles solely depend on the geographical position of the building, i.e., its latitude and longitude, and the time at which the measurement was taken. The position and orientation of a building can easily be found on plans or Google Maps, and we used the `Astral` Python library (<https://astral.readthedocs.io/en/latest/>) to compute the solar angles corresponding to each time step in our data.

Note that, while this processing works very well for unobstructed facades when its orientation is known, it cannot be used when for example other buildings or trees exist in front of the windows and create shading patterns. In that case, one has to rely on architectures that can automatically process horizontal solar irradiation measurements depending on time information, such as the LSTMs used in the black-box module of PCNNs. Nonetheless, we can use it in this paper since UMAR is not obstructed, leading to a very efficient computation of the true solar irradiation patterns on the windows of each zone.

A.4 Details on the data processing

A.4.1 NEST data

Data from all the sensors in NEST is sampled and stored at a frequency of one minute. Concerning the solar irradiation data, we delete constant streaks of more than 20 h than indicate a fault of the sensor – where *deleting* refers to setting the values to *NaN* – and clip the measurement at 0 since it cannot be negative. For the outside temperature, we delete constant streaks of more than 30 min. Both measurements are then smoothed with a Gaussian filter with $\sigma = 2$. For power inputs, we delete constant streaks of more than 1 day and smooth the measurements with a Gaussian filter with $\sigma = 1$. Finally, the temperature measurements in both the room of interest and the neighboring one are smoothed with $\sigma = 5$.

Before using the created data, we linearly interpolate all the missing values when less than 30 min of information is missing. When we use it to train and test PCNNs, the data is subsampled to 15 minute intervals through averaging.

The month and time of day variables are represented by sine and cosine functions to introduce periodicity, so that the last month has a value close to the first month of the year for example. Mathematically, two variables are created:

$$t_m^{sin} = \sin\left(\frac{m}{12}2\pi\right), \quad t_m^{cos} = \cos\left(\frac{m}{12}2\pi\right), \quad (\text{A.14})$$

where the months m are labeled linearly and in order from 1 to 12. The same processing is done for the time step during the day, replacing the factor 12 in Equation (A.14) by 96, the number of 15 min intervals in one day.

Once the data had been subsampled, discarding the 23% of incomplete measurements, i.e. where at least the information from one sensor is missing, we were left with over 80'000 data points. Since the proposed PCNN architectures are based on NNs in our implementations (see Section 2.4.3), they are not able to handle missing values, which prompted us to create a data set of time series without missing values.

As we aimed to design models that can predict the temperature dynamics over three-day-long horizons, we truncated each sequence to a maximum of three days and separated the heating and cooling seasons. We allowed the time series to overlap each hour, i.e., every four steps, to increase the data efficiency of the approach. Finally, since we implemented a warm-start period of 3 h for all the models, we also made sure the last 3 h of data exists for each time series. To avoid very short time series, we also ensured they always span at least 12 h. Altogether, this allowed us to create more than 11'000 sequences of data without missing values, which were split in a training and a validation set with proportions 80%-20%, respectively denoted \mathcal{D}_t and \mathcal{D}_v , and where $\mathcal{D}_t \cap \mathcal{D}_v = \emptyset$. Finally, the data is normalized between 0.1 and 0.9. For all NNs, the validation set is used to select the best set of weights along the training procedure.

A.4.2 Individual room energy consumption

As mentioned in Section 2.4.1, UMAR has a unique power meter and we need to disaggregate this global measurement P^{tot} into individual consumption for each room. To that end, we use the design mass flow \dot{m}^i of room i , something known from technical construction sheets. At each time step t , we then approximate the power consumed by each room, P^i , as follows:

$$P_t^i = \frac{u_t^i \dot{m}_t^i}{\sum_k u_t^k \dot{m}_t^k} P_t^{tot}, \quad (\text{A.15})$$

where u^i is the amount of time the valves are opened and we sum over all the $k = 5$ rooms in UMAR. In words, we approximate the individual energy consumption of each to be proportional to the amount of water flowing through its ceiling panels.

A.5 Linear model identification

As is classically done in linear system identification, we first used the least squares method to find the parameters a_h^z , a_c^z , b^z , c^{zy} , e^z best fitting the training data for each thermal zone z and neighboring zone $y \in \mathcal{N}(z)$, such as in Appendix A.1.2. However, ensuring none of these parameters is negative, which is necessary to respect the underlying physics, produced $c^{23} = 0$. This is clearly not physically meaningful, as it would mean there is no heat transfer from Zone 3 to Zone 2. Consequently, we also implemented a BO framework, relying on the `bayes_opt` Python library [350]. This allowed us to extensively search for the best physically consistent parameters — constraining them to be positive — for each zone over a five-step prediction horizon. We let BO run for 2'300 iterations, starting from 200 random initial points.

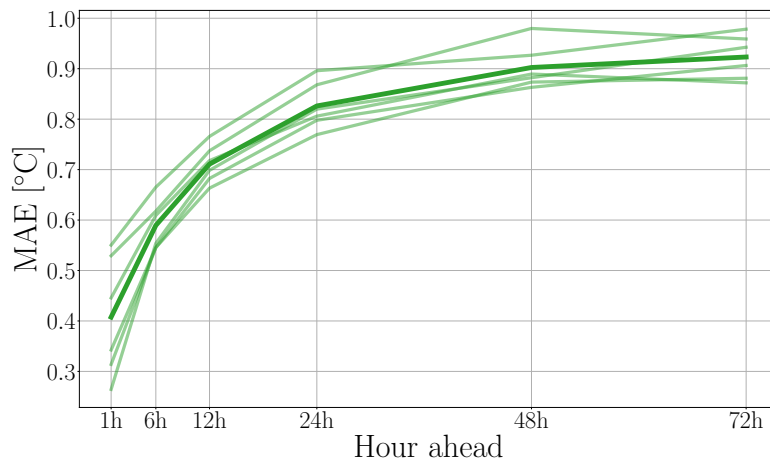


Figure A.2: MAE of six PCNNs with different random seeds at six chosen prediction steps in gray and the average in green, where the statistics were computed from almost 2000 predictions from the validation set.

A.6 Additional single-zone PCNN results

A.6.1 Robustness of PCNNs

To analyze the robustness of the PCNN discussed in Section 2.5.1, we trained five other PCNNs with the same architecture but different random seeds. As pictured in Figure A.2, all six models provide similar accuracy over the validation and the horizon of three days, except at the beginning of the prediction horizon. Two out of the six PCNNs trained indeed showed oscillatory behavior on the first prediction steps, leading to higher errors.

A.6.2 Flexibility of PCNNs

To test the flexibility of our approach, we additionally trained five PCNNs to model Zone 1, again with five different random seeds. As can be observed in Figure A.3, the models again present a similar accuracy over the prediction horizon, hinting at the robustness of the approach. Furthermore, the shown MAEs are similar to the errors obtained on Zone 3 (see Figure A.2), except towards the end of the prediction horizon, where the error is 20 – 40% higher. Nonetheless, the performance of PCNNs is comparable for both rooms, which is particularly interesting since no engineering was required to transfer the model between them: we used the same architecture for both bedrooms, simply changing the training and validation data sets. The training and validation errors displayed in Table A.1 confirm these conclusions.

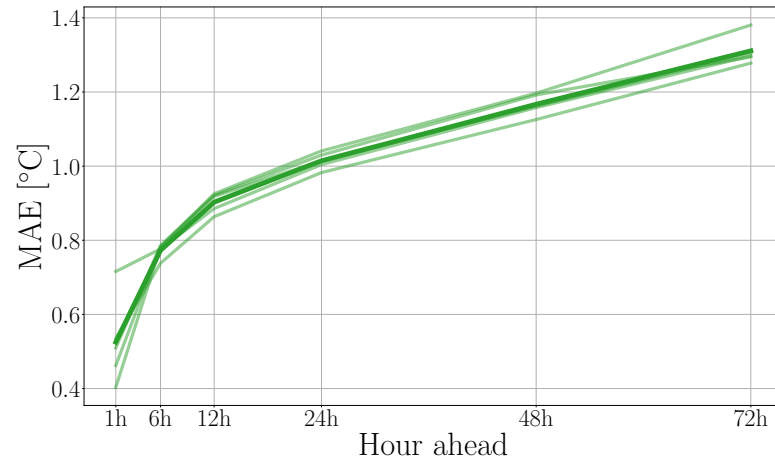


Figure A.3: MAE on the other bedroom in UMAR at key time steps of the prediction horizon for the PCNN with five different random seeds, where the statistics were computed from almost 2000 predictions from the validation set.

Seed	Training loss	Validation loss
0	1.82	2.42
1	1.66	2.44
2	1.58	2.52
3	1.66	2.54
4	1.66	2.39
Mean	1.68	2.46

Table A.1: Training and validation losses of five PCNNs on Zone 1, scaled by 10^3 .

Parameter	Starting value	Learned value
a_h	2	2.01
a_c	2	1.97
b	1.5	1.50
c	1.5	1.51

Table A.2: Comparison between the initial and learned values of the PCNN parameters, in degrees Celsius. For a_h and a_c , it represents how many degrees are gained in 4 h when heating/cooling at full power, while for b and c it represents how many degrees are lost through heat transfer in 6 h when the exogenous temperature is 25 °C lower.

A.6.3 Learned parameters

To complete the analysis of the PCNN presented in Section 2.5.1, we also display the final values of the parameters a_h , a_c , b , and c in Table A.2. Overall, we see that the parameters do not change much, and the same conclusion was drawn for the other PCNNs trained during our experiments. Out of the six PCNNs plotted in Figure A.2, only two modified the values substantially, even though by a maximum of 10% – 15%, and they correspond to the two models showing the worst performance overall.

A.7 Visualization of predictions

To complement Figure 2.15, the same experiment was carried on with the linear model and an LSTM, and zoomed-in predictions can be found in Figures A.4 and A.5. Note that each subplot uses a custom scale to better visualize the impact of different power inputs. We additionally shaded physically inconsistent behaviors in each subplot in gray, i.e., whenever the predicted temperature when cooling is applied is higher than when heating is applied or no power input is used, or when the temperature when heating is applied is lower than when no power is used. This confirms that the identified linear model failed to fully capture the impact of heating and cooling but still behaves in a physically consistent manner, e.g., with heating leading to higher temperatures than cooling, similar to the behavior that can be observed for the S-PCNN in Figure 2.15. On the other hand, both the PiNN and LSTM show inconsistent behaviors, especially in Zone 2 around the beginning of the prediction horizon.

A.8 X-PCNN gradients

In the case of X-PCNNs, at inference time, we use each single-zone PCNN to predict the next temperature in the corresponding zone. The new temperatures in the building are then updated in the data of all the single-zone PCNNs so they can predict the next step. This is required because the single-zone PCNNs cannot evolve independently over the prediction horizon since they depend on temperatures in neighboring zones at each step. However,

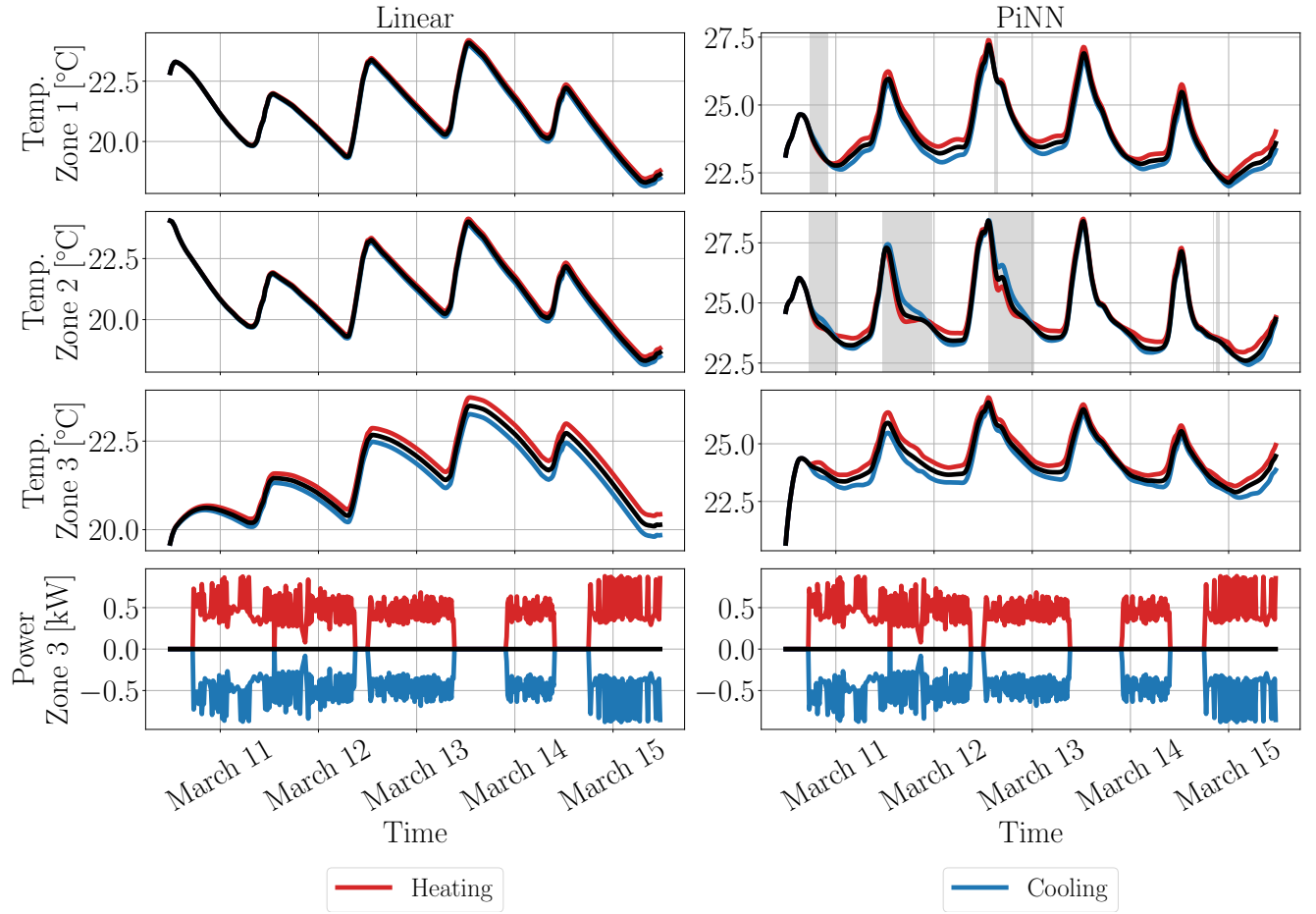


Figure A.4: Visualization of heat propagation for the linear baseline and a PiNN. The bottom plots show the heating (red) and cooling (blue) power inputs applied to Zone 3 while heating and cooling are turned off in Zone 1 and 2, compared to the situation when no power is applied (black). The other plots depict the corresponding temperature predictions of each model in the three zones. Gray-shaded areas mark physical inconsistencies in the PiNN predictions.

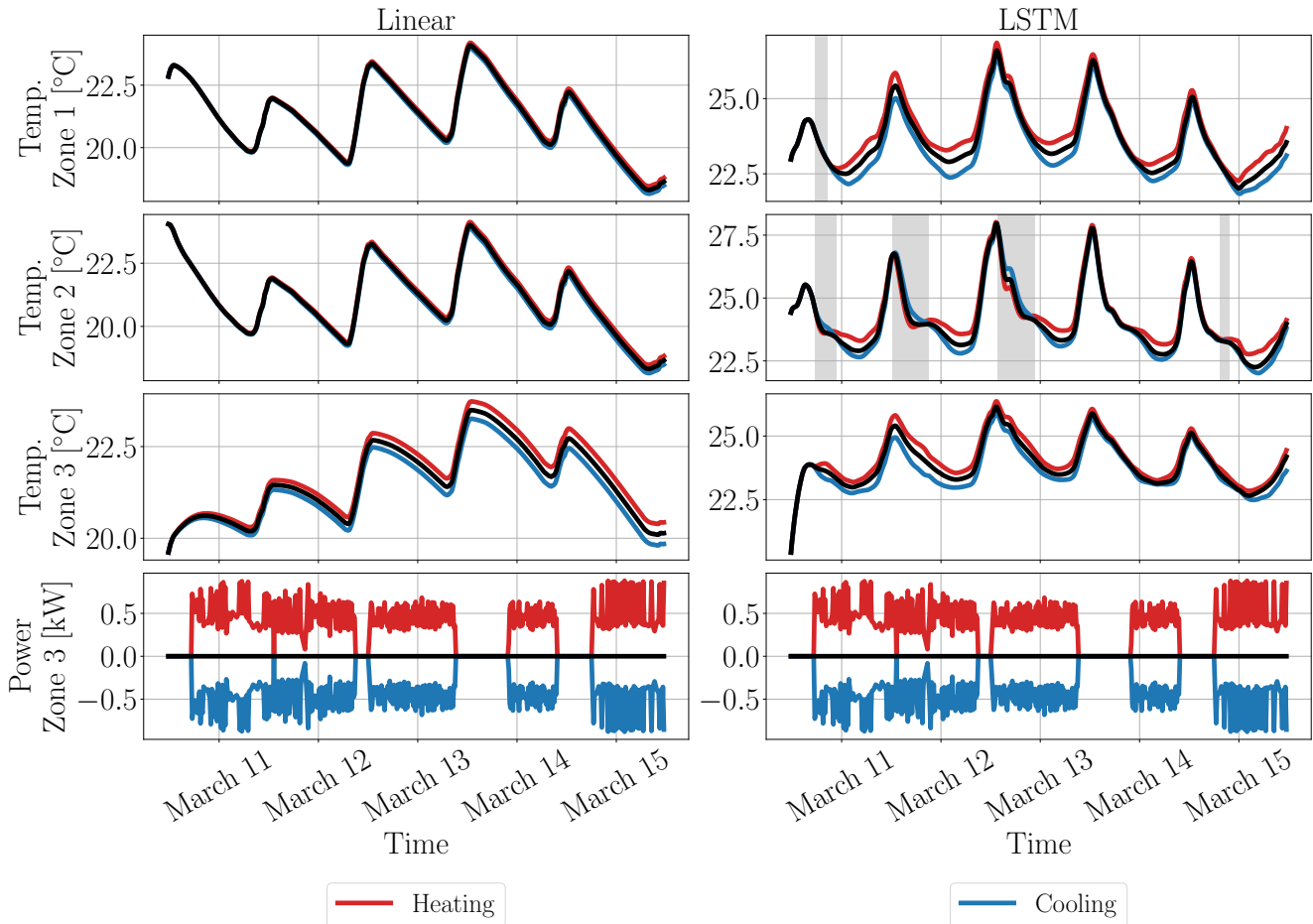


Figure A.5: Visualization of heat propagation for the linear model and an LSTM. The bottom plots show the heating (red) and cooling (blue) power inputs applied to Zone 3 while heating and cooling are turned off in Zone 1 and 2, compared to the situation when no power is applied (black). The other plots depict the corresponding temperature predictions of each model in the three zones. Gray-shaded areas mark physical inconsistencies in the LSTM predictions.

overwriting the data at each step breaks the automatic backpropagation of PyTorch, and we cannot automatically compute the gradient of the temperature in zone z with respect to power inputs or temperatures in another zone y without implementation overhead. We can only retrieve gradients with respect to each single-zone PCNN's power inputs u^z and the ambient temperature.

Intuitively, these available gradients are expected to be larger in magnitude than the gradients with respect to power inputs in other zones since they have a direct impact on the zone of interest. This explains the absence of low gradient values ($< 10^{-3}$) in Figure 2.16 for X-PCNNs compared to M- and S-PCNNs. Even if we can only compute parts of the gradients automatically, we still show them in Figure 2.16 for reference. Note that as we already know X-PCNNs are physically consistent since they satisfy the criteria of Corollary 1, these implementation considerations do not put the physical consistency of this architecture in jeopardy.

A.9 Number of numerical gradient values

The numerical investigation of NN-based model gradients in Section 2.5.2 is carried out on the validation data set of more than 750 three-day long sequences (288 steps). Following Remark 11, for each of the three zones, we compute the gradients of its last temperature predictions with respect to power inputs in all the zones (3 values) and the ambient temperature (1 value) at each step, giving rise to more than $750 \times 3 \times 288 \times (3 + 1) = 2'592'000$ values. In the case of X-PCNNs, we only have access to half of these values since we do not compute gradients with respect to power inputs in other zones (Appendix A.8), which still leaves us with more than 1 million values.

B Appendices - Chapter 4

B.1 Proof of Proposition 3

We want to ensure that the magnitude of all the eigenvalues of A is bounded by γ . From [351, Theorem 2.2], we know this is the case if and only if the following LMI holds for some symmetric $Q = Q^\top > 0$:

$$\begin{bmatrix} \gamma Q & AQ \\ \star & \gamma Q \end{bmatrix} > 0.$$

Taking its Schur complement, this is equivalent to

$$\gamma Q - A \frac{Q^\top}{\gamma} A^\top > 0. \quad (\text{B.1})$$

Defining the transformation $T = [I, -A]$ and introducing a free parameter $G \in \mathbb{R}^{n \times n}$, this can be rewritten as

$$\begin{aligned} & \gamma Q - AGA^\top - A^\top G^\top A \\ & \quad + AGA^\top + A^\top G^\top A - A \frac{Q^\top}{\gamma} A^\top > 0 \\ \Leftrightarrow & T \begin{bmatrix} \gamma Q & AG \\ G^\top A^\top & G^\top + G - \frac{Q^\top}{\gamma} \end{bmatrix} T^\top > 0 \\ \Leftrightarrow & \begin{bmatrix} \gamma Q & AG \\ G^\top A^\top & G^\top + G - \frac{Q^\top}{\gamma} \end{bmatrix} > 0. \end{aligned} \quad (\text{B.2})$$

In words, A is Schur with eigenvalues bounded by γ if and only if there exist $Q > 0$ and G such that (B.2) holds.

Let us now parametrize the left-hand side of the above LMI by the matrix S in (4.2). Remarkably, since S is positive definite by construction for any choice of W , (B.2) will always be satisfied, ensuring the stability of A .

Finally, define

$$S_{11} := \gamma Q, \quad S_{12} = S_{21}^\top := AG \quad S_{22} := G^\top + G - \frac{Q^\top}{\gamma}.$$

This allows us to recover $Q = \frac{S_{11}}{\gamma}$, which is positive definite and symmetric by construction. We then note that

$$G^\top + G = \frac{Q}{\gamma} + S_{22} = \frac{S_{11}}{\gamma^2} + S_{22} \quad (\text{B.3})$$

needs to hold. Since S_{11} and S_{22} are symmetric, (B.3) holds for any $V \in \mathbb{R}^{n \times n}$ if we set

$$G = \frac{1}{2} \left(\frac{S_{11}}{\gamma^2} + S_{22} \right) + V - V^\top.$$

Remembering that $A = S_{12}G^{-1}$ then concludes the proof. \square

B.2 Proof of Corollary 2

Let A be a Schur matrix and $\epsilon > 0$. Setting $\gamma = 1$ in the proof of Proposition B.1, we know there exists $Q = Q^\top > 0$ and G such that (B.2) holds, i.e.,

$$\begin{bmatrix} Q & AG \\ G^\top A^\top & G^\top + G - Q \end{bmatrix} =: \Gamma > 0.$$

Then, for any $\alpha > 0$, αQ and αG are valid alternative choices of Lyapunov function and free parameter because $\alpha Q = \alpha Q^\top > 0$ and

$$\begin{bmatrix} \alpha Q & A(\alpha G) \\ \alpha G^\top A^\top & \alpha G^\top + \alpha G - \alpha Q \end{bmatrix} = \alpha \Gamma > 0.$$

Since Γ is positive definite, $\lambda_{\min}(\Gamma) > 0$, and we can set

$$\alpha := \frac{2\epsilon}{\lambda_{\min}(\Gamma)} > 0, \quad \Delta := \alpha \Gamma - \epsilon \mathbb{I}_{2n}.$$

Then, according to Weyl's inequality [352], we have

$$\lambda_{\min}(\Delta) \geq \lambda_{\min}(\alpha \Gamma) - \lambda_{\max}(\epsilon \mathbb{I}_{2n}) = \alpha \lambda_{\min}(\Gamma) - \epsilon = 2\epsilon - \epsilon = \epsilon > 0,$$

so that $\Delta = \Delta^\top > 0$. We can then define $W = \Delta^{\frac{1}{2}}$, set $V = 0$ and construct A as in (4.3), with S as in (4.2). \square

B.3 Proof of Proposition 4

We need to show that A as defined in (4.3) is Schur, i.e., the autonomous system

$$x_{k+1} = Ax_k \quad (\text{B.4})$$

is stable. This is equivalent to finding a matrix $Q = Q^\top > 0$ that solves the following Lyapunov inequality [308]:

$$Q - A^\top QA > 0. \quad (\text{B.5})$$

By definition of A , we have

$$\begin{aligned} Q - A^\top QA &> 0 \\ \iff Q - (\mathbb{I}_n + \delta \bar{A})^\top Q (\mathbb{I}_n + \delta \bar{A}) &> 0 \\ \iff Q - Q - \delta Q \bar{A} - \delta \bar{A}^\top Q - \delta^2 \bar{A}^\top Q \bar{A} &> 0 \\ \iff -\delta Q \bar{A} - \delta \bar{A}^\top Q - \delta^2 \bar{A}^\top Q \bar{A} &> 0. \end{aligned}$$

Let us decompose \bar{A} as $\bar{A} = E^{-1}F$ for suitable matrices E and F . We can then rewrite the last inequality as

$$-\delta QE^{-1}F - \delta F^\top E^{-\top}Q - \delta^2 F^\top E^{-\top}QE^{-1}F > 0.$$

Defining $P = E^{-\top}QE^{-1} > 0$ and dividing by δ , this can be rewritten as

$$-QE^{-1}F - F^\top E^{-\top}Q - \delta F^\top PF > 0. \quad (\text{B.6})$$

Since $Q = E^\top PE$, (B.6) is equivalent to

$$-E^\top PF - F^\top PE - \delta F^\top PF > 0.$$

Using Schur's complement, this can be rewritten as

$$\begin{bmatrix} -E^\top PF - F^\top PE & F^\top \\ F & \frac{1}{\delta}P^{-1} \end{bmatrix} > 0. \quad (\text{B.7})$$

We then parametrize the left-hand side of the above LMI with S from (4.6). Critically, (B.7) will always be satisfied since S is positive definite by construction for any choice of W , ensuring the stability of A . Since $S_{22} = S_{22}^\top$, we can then recover

$$P = \frac{S_{22}^{-1}}{\delta}, \quad F = S_{21} = S_{12}^\top.$$

Knowing that $S_{11}^\top = S_{11}$ by definition and that

$$-E^\top PF - F^\top PE = S_{11}$$

needs to hold, we can set

$$F^\top P E = -\frac{S_{11}}{2} + V - V^\top \implies E^{-1} = -2(S_{11} + V - V^\top)^{-1} F^\top P,$$

for any $V \in \mathbb{R}^{n \times n}$. Since $\bar{A} = E^{-1} F$, this leads to

$$\begin{aligned} \bar{A} &= -2(S_{11} + V - V^\top)^{-1} F^\top P F = -\frac{2}{\delta} (S_{11} + V - V^\top)^{-1} S_{12} S_{22}^{-1} S_{21} \\ \implies A &= I + \delta \bar{A} = I - 2(S_{11} + V - V^\top)^{-1} S_{12} S_{22}^{-1} S_{21}. \end{aligned}$$

□

B.4 Proof of Corollary 3

Let A be a Schur matrix and $\epsilon > 0$. Setting $E = \mathbb{I}_n$ and $F = \bar{A}$ in the proof of Proposition 4, there exists a Lyapunov function $P = P^\top > 0$ such that (B.7) holds. Similarly to the proof of Corollary 2, for any $\alpha > 0$, αE , αF , and $\frac{P}{\alpha}$ are valid alternative choices since

$$\begin{bmatrix} -\alpha E^\top \frac{P}{\alpha} \alpha F - \alpha F^\top \frac{P}{\alpha} \alpha E & \alpha F^\top \\ \alpha F & \frac{1}{\delta} \alpha P^{-1} \end{bmatrix} \succ 0$$

and $(\alpha E)^{-1} \alpha F = \bar{A}$. One can then set $V = 0$ and follow the proof of Corollary 2 to find suitable values for α and W such that (4.7) holds for S as in (4.6). □

B.5 Proof of Proposition 5

B.5.1 Preliminaries

Throughout the proof below, we will use the following properties of the Hadamard product, for any matrices $K, L, M \in \mathbb{R}^{n \times n}$ and diagonal matrix $\Lambda \in \mathbb{R}^{n \times n}$ [352]:

$$(P1) \quad (K \odot L) + (K \odot M) = K \odot (L + M),$$

$$(P2) \quad (L \odot M)^\top = (L^\top \odot M^\top),$$

$$(P3) \quad (K \odot L)(\Lambda \odot M) = K \odot (L(\Lambda \odot M)),$$

where the last property holds since Λ is diagonal.

B.5.2 Proof of Proposition 5

As in the proof of Proposition 4, we need to find a symmetric matrix $Q = Q^\top > 0$ such that $Q - A^\top Q A > 0$. Following the proof of Proposition 3 for $\gamma = 1$, one can show that this is

equivalent to finding $Q = Q^\top > 0$ such that

$$\begin{bmatrix} Q & AG \\ G^\top A^\top & G^\top + G - Q \end{bmatrix} > 0, \quad (\text{B.8})$$

where $G \in \mathbb{R}^{n \times n}$ is a free parameter. In this sparse case, we consider diagonal Q and G matrices of the form

$$Q = N \odot P, \quad G = N \odot H,$$

for some $P, H \in \mathbb{R}^{n \times n}$ and with N defined in (4.9). Note here that this also implies $Q = Q^\top$, as required in (B.8). Recalling that we want to identify a matrix A of the form $A = \mathcal{M} \odot \tilde{A}$ for some \tilde{A} , (B.8) can be written as

$$\begin{bmatrix} N \odot P & (\mathcal{M} \odot \tilde{A})(N \odot H) \\ (*)^\top & (N \odot H)^\top + (N \odot H) - (N \odot P) \end{bmatrix} \stackrel{(\text{P1})}{=} \begin{bmatrix} N \odot P & (\mathcal{M} \odot \tilde{A})(N \odot H) \\ (*)^\top & N \odot (H^\top + H - P) \end{bmatrix} > 0,$$

where $(*)^\top$ represents the transpose of the upper right block. Since N is a diagonal matrix, using (P3), we can rewrite the above LMI as

$$\begin{aligned} & \begin{bmatrix} N \odot P & \mathcal{M} \odot (\tilde{A}(N \odot H)) \\ (\mathcal{M} \odot (\tilde{A}(N \odot H)))^\top & N \odot (H^\top + H - P) \end{bmatrix} > 0 \\ \stackrel{(\text{P2})}{\iff} & \begin{bmatrix} N \odot P & \mathcal{M} \odot (\tilde{A}(N \odot H)) \\ \mathcal{M}^\top \odot (\tilde{A}(N \odot H))^\top & N \odot (H^\top + H - P) \end{bmatrix} > 0 \\ \iff & \begin{bmatrix} N & \mathcal{M} \\ (*)^\top & N \end{bmatrix} \odot \begin{bmatrix} P & \tilde{A}(N \odot H) \\ (*)^\top & H^\top + H - P \end{bmatrix} > 0. \end{aligned} \quad (\text{B.9})$$

A sufficient condition for the above Hadamard product to be positive semi-definite is to ensure that both factors are individually positive semi-definite [353], i.e.,

$$\begin{bmatrix} N & \mathcal{M} \\ (*)^\top & N \end{bmatrix} > 0 \quad (\text{B.10})$$

$$\begin{bmatrix} P & \tilde{A}(N \odot H) \\ (*)^\top & H^\top + H - P \end{bmatrix} > 0. \quad (\text{B.11})$$

Since \mathcal{M} is fixed and known, (B.10) is satisfied by construction of N in (4.9) according to the Levy–Desplanques theorem [352]. To satisfy (B.11), as in Proposition 4, we parametrize its left-hand side with S in (4.8), which allows us to recover

$$P = S_{11}, \quad H + H^\top = S_{22} + P = S_{11} + S_{22}.$$

Appendix B. Appendices - Chapter 4

As before, by symmetry of S_{11} and S_{22} , for any $V \in \mathbb{R}^{n \times n}$, the right equation is satisfied for

$$H = \frac{1}{2}(S_{11} + S_{22}) + V - V^\top.$$

Finally,

$$\begin{aligned} \tilde{A} &= S_{12}(N \odot H)^{-1} = S_{12} \left[N \odot \left(\frac{1}{2}(S_{11} + S_{22}) + V - V^\top \right) \right]^{-1} \\ \Rightarrow A &= \mathcal{M} \odot \tilde{A} = \mathcal{M} \odot \left(S_{12} \left[N \odot \left(\frac{1}{2}(S_{11} + S_{22}) + V - V^\top \right) \right]^{-1} \right). \end{aligned}$$

□

B.6 Proof of Proposition 6

First, the desired sparsity pattern is achieved because

$$\frac{\sigma(\eta)\gamma}{|\lambda(\mathcal{M} \odot V)|_{\max}} (\mathcal{M} \odot V) = \mathcal{M} \odot \left(\frac{\sigma(\eta)\gamma}{|\lambda(\mathcal{M} \odot V)|_{\max}} V \right)$$

by definition of the Hadamard product since the maximum eigenvalue is a scalar.

To show that A is Schur with eigenvalues in a circle of radius γ centered at the origin, suppose β is an eigenvalue of $(\mathcal{M} \odot V)$ corresponding to the eigenvector e , i.e., $(\mathcal{M} \odot V)e = \beta e$. Then,

$$Ae = \frac{\sigma(\eta)\gamma}{|\lambda(\mathcal{M} \odot V)|_{\max}} (\mathcal{M} \odot V)e = \frac{\sigma(\eta)\gamma}{|\lambda(\mathcal{M} \odot V)|_{\max}} \beta e =: \alpha e,$$

so that e is still an eigenvector of A , with eigenvalue α . By definition of $|\lambda(\mathcal{M} \odot V)|_{\max}$ and since $0 < \sigma(\eta) < 1$, we obtain

$$|\alpha| = \frac{\sigma(\eta)\gamma}{|\lambda(\mathcal{M} \odot V)|_{\max}} |\beta| < \frac{|\beta|}{|\lambda(\mathcal{M} \odot V)|_{\max}} \gamma \leq \gamma,$$

hence, concluding the proof. □

B.7 Proof of Corollary 4

One can set $\mathcal{M} := \mathbb{1}_{n \times n}$, $V := A$, γ such that $|\lambda(A)|_{\max} < \gamma \leq 1$, which exists since the matrix A is Schur, and $\eta := \sigma^{-1}(|\lambda(A)|_{\max}/\gamma)$, which is well-defined since $|\lambda(A)|_{\max}/\gamma < 1$ by definition of γ .

B.8 Temperature computation

By definition, the energy U of a mass m of air can be described as $U(T) = mc(T)T$, where c is the specific heat capacity of air and T is temperature. Assuming we deal with an ideal gas, we also know that $PV = nRT$, for a given absolute gas pressure P , volume V , n moles of substance, and where $R \approx 8.31 \text{ J K}^{-1} \text{ mol}^{-1}$ is the universal gas constant. This allows us to rewrite the time derivative of entropy, by definition satisfying $T\dot{S} = \dot{U} + P\dot{V}$ as [354]

$$\frac{dS}{dt} = \frac{1}{T} \frac{dU}{dt} + \frac{P}{T} \frac{dV}{dt} = \frac{1}{T} \frac{d}{dt}(mc(T)T) + \frac{nRT}{VT} \frac{d}{dt}V.$$

Since the temperature does not change abruptly, we can assume a constant heat capacity $c(T) \approx c$ and obtain

$$\frac{dS}{dt} \approx mc \frac{\frac{dT}{dt}}{T} + nR \frac{\frac{dV}{dt}}{V} = mc \frac{d[\ln T]}{dt} + nR \frac{d[\ln V]}{dt}. \quad (\text{B.12})$$

Integrating (B.12) on both sides from an initial time t_i to a final time t_f , we get

$$\int_{t_i}^{t_f} \frac{dS}{dt} dt = mc \int_{t_i}^{t_f} \frac{d \ln T}{dt} dt + nR \int_{t_i}^{t_f} \frac{d \ln V}{dt} dt,$$

leading to

$$S(t_f) - S(t_i) = mc \ln \frac{T(t_f)}{T(t_i)} + nR \ln \frac{V(t_f)}{V(t_i)} = \ln \left[\left(\frac{T(t_f)}{T(t_i)} \right)^{mc} \left(\frac{V(t_f)}{V(t_i)} \right)^{nR} \right].$$

We can thus compute the final temperatures as

$$T(t_f) = \left[\exp \left(\frac{S(t_f) - S(t_i)}{mc} \right) \left(\frac{V(t_f)}{V(t_i)} \right)^{\frac{-nR}{mc}} \right] T(t_i).$$

B.9 Proof of Proposition 7

Let us define $\mathcal{E} = \{(i, j) \mid \text{Zones } i \text{ and } j \text{ are adjacent}\}$, the set of connections between the thermal zones, and consider the following decomposition of $\tilde{J}(T)$, for $R_k : \mathbb{R}^n \mapsto \mathbb{R}$:

$$\tilde{J}(T) = \sum_{k \in \mathcal{E}} R_k(T) \mathcal{J}_k \quad R_k(T) = \lambda_{ij} \frac{(T_j - T_i)}{(T_i T_j)},$$

where \mathcal{J}_k is an $N \times N$ constant skew-symmetric matrix with zeroes everywhere, except $(J_k)_{ij} = -(J_k)_{ji} = 1$, for $k = (i, j)$.

Appendix B. Appendices - Chapter 4

Then, for $T_e, Q_s, Q_h, Q_c \equiv 0$, we have:

$$\frac{dH}{dt} = \frac{\partial H(S)}{\partial S}^\top \dot{S} = \frac{\partial H(S)}{\partial S}^\top \left[\sum_{k \in \mathcal{E}} R_k(T) \mathcal{J}_k \right] \frac{\partial H(S)}{\partial S} = \sum_{k \in \mathcal{E}} \left[\frac{\partial H(S)}{\partial S}^\top R_k(T) \mathcal{J}_k \frac{\partial H(S)}{\partial S} \right] = 0,$$

since each term of the sum is zero, as in equation (4.21), because each \mathcal{J}_k is now constant and each $R_k(T)$ satisfies condition P2. This proves the required conservation of energy of the system.

In order to verify the irreversible creation of total entropy S^t , we note that for $T_e, Q_s, Q_h, Q_c \equiv 0$,

$$R_k(T) = \left(\frac{\lambda_{ij}}{T_i T_j} \right) \{S^t, H\}_{\mathcal{J}_k},$$

as in the case of two heat exchangers [343]. Since the total entropy S^t is the sum of the entropy in each zone d , we get

$$\begin{aligned} \dot{S}_t &= \sum_{d=1}^z (\dot{S})_d = \sum_{d=1}^z \left(\left[\sum_{k \in \mathcal{E}} R_k \mathcal{J}_k \right] \frac{\partial H(S)}{\partial S} \right)_d = \sum_{k \in \mathcal{E}} R_k(T) \sum_{d=1}^z \left(\mathcal{J}_k \frac{\partial H}{\partial S} \right)_d \\ &= \sum_{k \in \mathcal{E}} R_k(T) \left(\mathbb{1}_n^\top \mathcal{J}_k \frac{\partial H}{\partial S} \right) = \sum_{k \in \mathcal{E}} R_k(T) \left(\frac{\partial S^t}{\partial S}^\top \mathcal{J}_k \frac{\partial H}{\partial S} \right) = \sum_{k \in \mathcal{E}} \frac{\lambda_{ij}}{T_i T_j} \{S^t, H\}_{\mathcal{J}_k}^2 \geq 0, \end{aligned}$$

since $R_k(T) \in \mathbb{R}$ and $\frac{\partial S^t}{\partial S} = \mathbb{1}_z$ by definition, and the inequality holds if all $\{\lambda_{ij}\}_{(i,j) \in \mathcal{E}}$ are positive since temperatures (defined in Kelvin) are positive.

Finally, if all the input matrices B_e, B_s, B_h , and B_c are positive definite, monotonicity follows from the fact that PC-NODE (4.24) is affine in input by construction. \square

Bibliography

- [1] International Energy Agency (IEA). Buildings. <https://www.iea.org/energy-system/buildings#tracking>, 2023. [Accessed: 2023.08.10].
- [2] International Energy Agency (IEA). Heating. <https://www.iea.org/energy-system/buildings/heating>, 2023. [Accessed: 2023.08.10].
- [3] International Energy Agency (IEA). The future of cooling. <https://www.iea.org/reports/the-future-of-cooling>, 2018. [Accessed: 2023.08.10].
- [4] Benjamin D Leibowicz, Christopher M Lanham, Max T Brozynski, José R Vázquez-Canteli, Nicolás Castillo Castejón, and Zoltan Nagy. Optimal decarbonization pathways for urban residential building energy services. *Applied energy*, 230:1311–1325, 2018.
- [5] Paul Westermann and Ralph Evins. Surrogate modelling for sustainable building design—a review. *Energy and Buildings*, 198:170–186, 2019.
- [6] Chirag Deb and Arno Schlueter. Review of data-driven energy modelling techniques for building retrofit. *Renewable and Sustainable Energy Reviews*, 144:110990, 2021.
- [7] Ján Drgoňa, Javier Arroyo, Iago Cupeiro Figueroa, David Blum, Krzysztof Arendt, Donghun Kim, Enric Perarnau Ollé, Juraj Oravec, Michael Wetter, Draguna L Vrabie, et al. All you need to know about model predictive control for buildings. *Annual Reviews in Control*, 50:190–232, 2020.
- [8] Tyler Hoyt, Edward Arens, and Hui Zhang. Extending air temperature setpoints: Simulated energy savings and design considerations for new and retrofit buildings. *Building and Environment*, 88:89–96, 2015.
- [9] Sama Aghniaey and Thomas M Lawrence. The impact of increased cooling setpoint temperature during demand response events on occupant thermal comfort in commercial buildings: A review. *Energy and Buildings*, 173:19–27, 2018.
- [10] Yue Lei, Sicheng Zhan, Eikichi Ono, Yuzhen Peng, Zhiang Zhang, Takamasa Hasama, and Adrian Chong. A practical deep reinforcement learning framework for multivariate occupant-centric control in buildings. *Applied Energy*, 324:119742, 2022.
- [11] Mahsa Farjadnia, Angela Fontan, Alessio Russo, Karl Henrik Johansson, and Marco Molinari. What influences occupants' behavior in residential buildings: An experimental study on window operation in the KTH Live-In Lab. *arXiv preprint arXiv:2307.08090*, 2023.
- [12] Sophie Naylor, Mark Gillott, and Tom Lau. A review of occupant-centric building control strategies to reduce building energy use. *Renewable and Sustainable Energy Reviews*, 96:1–10, 2018.
- [13] Javier Arroyo, Fred Spiessens, and Lieve Helsen. Comparison of optimal control techniques for building energy management. *Frontiers in Built Environment*, 8:849754, 2022.
- [14] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.
- [15] Bratislav Svetozarevic, Christian Baumann, Simon Muntwiler, Loris Di Natale, Melanie N Zeilinger, and Philipp Heer. Data-driven control of room temperature and bidirectional EV charging using deep reinforcement learning: Simulations and experiments. *Applied Energy*, 307:118127, 2022.
- [16] Timothy I Salisbury. A survey of control technologies in the building automation industry. *IFAC Proceedings Volumes*, 38(1):90–100, 2005.

Bibliography

- [17] Felix Bünning, Benjamin Huber, Philipp Heer, Ahmed Aboudonia, and John Lygeros. Experimental demonstration of data predictive control for energy optimization and thermal comfort in buildings. *Energy and Buildings*, 211:109792, 2020.
- [18] Maximilian Mork, Florian Redder, André Xhonneux, and Dirk Müller. Real-world implementation and evaluation of a Model Predictive Control framework in an office space. *Journal of Building Engineering*, page 107619, 2023.
- [19] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [20] Davide Coraci, Silvio Brandi, and Alfonso Capozzoli. Effective pre-training of a deep reinforcement learning agent by means of long short-term memory models for thermal energy management in buildings. *Energy Conversion and Management*, 291:117303, 2023.
- [21] Xianzhong Ding, Alberto Cerpa, and Wan Du. Exploring deep reinforcement learning for holistic smart building control. *arXiv preprint arXiv:2301.11510*, 2023.
- [22] Usman Ali, Mohammad Haris Shamsi, Cathal Hoare, Eleni Mangina, and James O'Donnell. Review of urban building energy modeling (UBEM) approaches, methods and tools using qualitative and quantitative analysis. *Energy and Buildings*, 246:111073, <https://doi.org/10.1016/j.enbuild.2021.111073>, 2021.
- [23] Mohammad Haris Shamsi, Usman Ali, Eleni Mangina, and James O'Donnell. Feature assessment frameworks to evaluate reduced-order grey-box building energy models. *Applied Energy*, 298:117174, <https://doi.org/10.1016/j.apenergy.2021.117174>, 2021.
- [24] Mohammad Haris Shamsi, Usman Ali, and James O'Donnell. A generalization approach for reduced order modelling of commercial buildings. *Journal of Building Performance Simulation*, 12(6):729–744, <https://doi.org/10.1080/19401493.2019.1641554>, 2019.
- [25] Paige Wenbin Tien, Shuangyu Wei, Jo Darkwa, Christopher Wood, and John Kaiser Calautit. Machine Learning and Deep Learning Methods for Enhancing Building Energy Efficiency and Indoor Environmental Quality—A Review. *Energy and AI*, page 100198, 2022.
- [26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [27] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
- [28] Emilio T Maddalena, Yingzhao Lian, and Colin N Jones. Data-driven methods for building control—A review and promising future directions. *Control Engineering Practice*, 95:104211, 2020.
- [29] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.
- [30] Francesco M Solinas, Alberto Macii, Edoardo Patti, and Lorenzo Bottaccioli. An online reinforcement learning approach for HVAC control. *Expert Systems with Applications*, page 121749, 2023.
- [31] Qiming Fu, Zhicong Han, Jianping Chen, You Lu, Hongjie Wu, and Yunzhe Wang. Applications of reinforcement learning for building energy efficiency control: A review. *Journal of Building Engineering*, 50:104165, 2022.
- [32] Samir Touzani, Anand Krishnan Prakash, Zhe Wang, Shreya Agarwal, Marco Pritoni, Mariam Kiran, Richard Brown, and Jessica Granderson. Controlling distributed energy resources via deep reinforcement learning for load flexibility and energy efficiency. *Applied Energy*, 304:117733, <https://doi.org/10.1016/j.apenergy.2021.117733>, 2021.
- [33] Phillip Stoffel, Patrick Henkel, Martin Rätz, Alexander Kümpel, and Dirk Müller. Safe operation of online learning data driven model predictive control of building energy systems. *Energy and AI*, page 100296, 2023.
- [34] Truong X Nghiem, Ján Drgoňa, Colin Jones, Zoltan Nagy, Roland Schwan, Biswadip Dey, Ankush Chakrabarty, Stefano Di Cairano, Joel A Paulson, Andrea Carron, et al. Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems. *arXiv preprint arXiv:2306.13867*, 2023.
- [35] Priya L Donti and J Zico Kolter. Machine learning for sustainable energy systems. *Annual Review of Environment and Resources*, 46:719–747, 2021.
- [36] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

- [37] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.
- [38] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.
- [39] Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and Colin N Jones. Physically consistent neural networks for building thermal modeling: theory and analysis. *Applied Energy*, 325:119806, 2022.
- [40] Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and Colin Neil Jones. Towards scalable physically consistent neural networks: An application to data-driven multi-zone thermal building models. *Applied Energy*, 340:121071, 2023.
- [41] José R Vázquez-Canteli and Zoltán Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied energy*, 235:1072–1089, 2019.
- [42] Zachary E Lee and K Max Zhang. Generalized reinforcement learning for building control using Behavioral Cloning. *Applied Energy*, 304:117602, 2021.
- [43] Silvio Brandi, Massimo Fiorentini, and Alfonso Capozzoli. Comparison of online and offline deep reinforcement learning with model predictive control for thermal energy management. *Automation in Construction*, 135:104128, 2022.
- [44] Dan Wang, Wanfu Zheng, Zhe Wang, Yaran Wang, Xiufeng Pang, and Wei Wang. Comparison of reinforcement learning and model predictive control for building energy system optimization. *Applied Thermal Engineering*, 228:120430, 2023.
- [45] Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and Colin N Jones. Near-optimal deep reinforcement learning policies from data for zone temperature control. In *2022 IEEE 17th International Conference on Control & Automation (ICCA)*, pages 698–703. IEEE, 2022.
- [46] Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and Colin N Jones. Computationally efficient reinforcement learning: Targeted exploration leveraging simple rules. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 2334–2339. IEEE, 2023.
- [47] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.
- [48] Ian R Manchester, Max Revay, and Ruigang Wang. Contraction-based methods for stable identification and robust machine learning: a tutorial. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2955–2962. IEEE, 2021.
- [49] Jan M Maciejowski. Guaranteed stability with subspace methods. *Systems & Control Letters*, 26(2):153–156, 1995.
- [50] Giuseppe Armenise, Marco Vaccari, Riccardo Bacci Di Capaci, and Gabriele Pannocchia. An open-source system identification package for multivariable processes. In *2018 UKACC 12th International Conference on Control (CONTROL)*, pages 152–157. IEEE, 2018.
- [51] Giorgos Mamakoukas, Ian Abraham, and Todd D Murphey. Learning stable models for prediction and control. *IEEE Transactions on Robotics*, 2023.
- [52] Chengpu Yu and Michel Verhaegen. Subspace identification of distributed clusters of homogeneous systems. *IEEE Transactions on Automatic Control*, 62(1):463–468, 2016.
- [53] Chengpu Yu, Lennart Ljung, Adrian Wills, and Michel Verhaegen. Constrained subspace method for the identification of structured state-space models (COSMOS). *IEEE Transactions on Automatic Control*, 65(10):4201–4214, 2019.
- [54] S Joe Qin. An overview of subspace identification. *Computers & chemical engineering*, 30(10-12):1502–1513, 2006.
- [55] Oliver Nelles and Oliver Nelles. *Nonlinear dynamic system identification*. Springer, 2020.
- [56] Loris Di Natale, Muhammad Zakwan, Bratislav Svetozarevic, Philipp Heer, Giancarlo Ferrari Trecate, and Colin N Jones. Linear Stable Subspace Identification: A Machine Learning Approach. *Manuscript submitted to the European Control Conference (ECC)*, 2024.
- [57] Loris Di Natale, Muhammad Zakwan, Bratislav Svetozarevic, Philipp Heer, Giancarlo Ferrari Trecate, and Colin N Jones. SIMBa: System Identification Methods leveraging Backpropagation. *arXiv preprint arXiv:-*, 2023.

Bibliography

- [58] Muhammad Zakwan, Loris Di Natale, Bratislav Svetožarevic, Philipp Heer, Colin N Jones, and Giancarlo Ferrari Trecate. Physically consistent neural ODEs for learning multi-physics systems. *arXiv preprint arXiv:2211.06130*, 2023.
- [59] Zongcai Du, Ding Liu, Jie Liu, Jie Tang, Gangshan Wu, and Lean Fu. Fast and memory-efficient network towards efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 853–862, 2022.
- [60] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [61] Mirco Fabbri and Gianluca Moro. Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks. In *Data*, pages 142–153, 2018.
- [62] Yoav Goldberg. *Neural network methods for natural language processing*. Springer Nature, 2022.
- [63] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.
- [64] Jonas Degraeve, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.
- [65] Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pages 263–277. PMLR, 2022.
- [66] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [67] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.
- [68] Amir Rosenfeld, Richard Zemel, and John K Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.
- [69] Rey Reza Wiyatno, Anqi Xu, Ousmane Dia, and Archy de Berker. Adversarial examples in modern machine learning: A review. *arXiv preprint arXiv:1911.05268*, 2019.
- [70] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.
- [71] Hendrik Heuer, Christof Monz, and Arnold WM Smeulders. Generating captions without looking beyond objects. *arXiv preprint arXiv:1610.03708*, 2016.
- [72] John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric Karl Oermann. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. *PLoS medicine*, 15(11):e1002683, 2018.
- [73] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.
- [74] Yun Xu and Royston Goodacre. On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*, 2(3):249–262, <https://doi.org/10.1007/s41664--018--0068--2>, 2018.
- [75] Hossein Abbasimehr and Reza Paki. Improving time series forecasting using LSTM and attention models. *Journal of Ambient Intelligence and Humanized Computing*, 13(1):673–691, 2022.
- [76] Alexander D’Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *The Journal of Machine Learning Research*, 23(1):10237–10297, 2022.

- [77] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020.
- [78] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7):1235–1270, https://doi.org/10.1162/neco_a_01199, 2019.
- [79] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
- [80] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.
- [81] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, <https://doi.org/10.1109/TKDE.2017.2720168>, 2017.
- [82] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.
- [83] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part II): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.
- [84] Yibo Yang and Paris Perdikaris. Physics-informed deep generative models. *arXiv preprint arXiv:1812.03511*, 2018.
- [85] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
- [86] Chengping Rao, Pu Ren, Qi Wang, Oral Buyukozturk, Hao Sun, and Yang Liu. Encoding physics to learn reaction–diffusion processes. *Nature Machine Intelligence*, pages 1–15, 2023.
- [87] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Michal Walczak, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, <https://doi.org/10.1109/TKDE.2021.3079836>, 2021.
- [88] Arka Daw, Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.
- [89] Xinbin Liang, Xu Zhu, Siliang Chen, Xinqiao Jin, Fu Xiao, and Zhimin Du. Physics-constrained cooperative learning-based reference models for smart management of chillers considering extrapolation scenarios. *Applied Energy*, 349:121642, 2023.
- [90] Johannes Hendriks, Carl Jidling, Adrian Wills, and Thomas Schön. Linearly constrained neural networks. *arXiv preprint arXiv:2002.01600*, 2020.
- [91] Franz M Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C Geiger. On the Apparent Pareto Front of Physics-informed Neural Networks. *IEEE Access*, 2023.
- [92] Jingyi Yuan and Yang Weng. Physics interpretable shallow-deep neural networks for physical system identification with unobservability. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 847–856. IEEE, 2021.
- [93] Xinyue Hu, Haoji Hu, Saurabh Verma, and Zhi-Li Zhang. Physics-guided deep neural networks for power flow analysis. *IEEE Transactions on Power Systems*, 36(3):2082–2092, <https://doi.org/10.1109/TPWRS.2020.3029557>, 2020.
- [94] Raad Z Homod. Review on the HVAC system modeling types and the shortcomings of their application. *Journal of Energy*, 2013:, <https://doi.org/10.1155/2013/768632>, 2013.
- [95] Aurélie Fouquier, Sylvain Robert, Frédéric Suard, Louis Stéphan, and Arnaud Jay. State of the art in building modelling and energy performances prediction: A review. *Renewable and Sustainable Energy Reviews*, 23: 272–288, <https://doi.org/10.1016/j.rser.2013.03.004>, 2013.
- [96] Xiwang Li and Jin Wen. Review of building energy modeling for control and operation. *Renewable and Sustainable Energy Reviews*, 37:517–537, <https://doi.org/10.1016/j.rser.2014.05.056>, 2014.

Bibliography

- [97] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, <https://doi.org/10.1016/j.rser.2017.02.085>, 2017.
- [98] Abhinandana Boodi, Karim Beddiar, Malek Benamour, Yassine Amirat, and Mohamed Benbouzid. Intelligent systems for building energy and occupant comfort optimization: A state of the art review and recommendations. *Energies*, 11(10):2604, <https://doi.org/10.3390/en11102604>, 2018.
- [99] Zakia Afroz, GM Shafiullah, Tania Urmee, and Gary Higgins. Modeling techniques used in building HVAC control systems: A review. *Renewable and sustainable energy reviews*, 83:64–84, <https://doi.org/10.1016/j.rser.2017.10.044>, 2018.
- [100] Mathieu Bourdeau, Xiao qiang Zhai, Elyes Nefzaoui, Xiaofeng Guo, and Patrice Chatellier. Modeling and forecasting building energy consumption: A review of data-driven techniques. *Sustainable Cities and Society*, 48:101533, <https://doi.org/10.1016/j.scs.2019.101533>, 2019.
- [101] Ting Yang, Liyuan Zhao, Wei Li, Jianzhong Wu, and Albert Y Zomaya. Towards healthy and cost-effective indoor environment management in smart homes: A deep reinforcement learning approach. *Applied Energy*, 300:117335, 2021.
- [102] Suroor M Dawood, Alireza Hatami, and Raad Z Homod. Trade-off decisions in a novel deep reinforcement learning for energy savings in HVAC systems. *Journal of Building Performance Simulation*, 15(6):809–831, 2022.
- [103] Cheng Fan, Fu Xiao, and Yang Zhao. A short-term building cooling load prediction method using deep learning algorithms. *Applied energy*, 195:222–233, <https://doi.org/10.1016/j.apenergy.2017.03.064>, 2017.
- [104] Tianshu Wei, Shaolei Ren, and Qi Zhu. Deep reinforcement learning for joint datacenter and HVAC load control in distributed mixed-use buildings. *IEEE Transactions on Sustainable Computing*, pages , <https://doi.org/10.1109/TSUSC.2019.2910533>, 2019.
- [105] Wei Tian, Yeonsook Heo, Pieter De Wilde, Zhanyong Li, Da Yan, Cheol Soo Park, Xiaohang Feng, and Godfried Augenbroe. A review of uncertainty analysis in building energy assessment. *Renewable and Sustainable Energy Reviews*, 93:285–301, <https://doi.org/10.1016/j.rser.2018.05.029>, 2018.
- [106] Drury B Crawley, Linda K Lawrie, Frederick C Winkelmann, Walter F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, Richard J Liesen, Daniel E Fisher, Michael J Witte, et al. EnergyPlus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, [https://doi.org/10.1016/S0378-7788\(00\)00114-6](https://doi.org/10.1016/S0378-7788(00)00114-6), 2001.
- [107] Michael Wetter and Christoph Haugstetter. Modelica versus TRNSYS—A comparison between an equation-based and a procedural modeling language for building energy simulation. *Proceedings of SimBuild*, 2(1), 2006.
- [108] Domenico Mazzeo, Nicoletta Matera, Cristina Cornaro, Giuseppe Oliveti, Piercarlo Romagnoni, and Livio De Santoli. EnergyPlus, IDA ICE and TRNSYS predictive simulation accuracy for building thermal behaviour evaluation by using an experimental campaign in solar test boxes with and without a PCM module. *Energy and Buildings*, 212:109812, <https://doi.org/10.1016/j.enbuild.2020.109812>, 2020.
- [109] Hassan Harb, Neven Boyanov, Luis Hernandez, Rita Streblow, and Dirk Müller. Development and validation of grey-box models for forecasting the thermal response of occupied buildings. *Energy and Buildings*, 117:199–207, 2016.
- [110] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, and Khoo Poh Lam. Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning. *Energy and Buildings*, 199:472–490, 2019.
- [111] Ankush Chakrabarty, Emilio Maddalena, Hongtao Qiao, and Christopher Laughman. Scalable Bayesian Optimization for Model Calibration: Case Study on Coupled Building and HVAC Dynamics. *Energy and Buildings*, pages 111460, <https://doi.org/10.1016/j.enbuild.2021.111460>, 2021.
- [112] Fabrizio Ascione, Nicola Bianco, Claudio De Stasio, Gerardo Maria Mauro, and Giuseppe Peter Vanoli. Artificial neural networks to predict energy performance and retrofit scenarios for any member of a building category: A novel approach. *Energy*, 118:999–1017, 2017.
- [113] Sullivan Royer, Stéphane Thil, and Thierry Talbert. Towards a generic procedure for modeling buildings and their thermal zones. In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1–6, <https://doi.org/10.1109/EEEIC.2016.7555567>. IEEE, 2016.

- [114] Karim Boubouh, Robert Basmadjian, Omid Ardakanian, Alexandre Maurer, and Rachid Guerraoui. Efficacy of temporal and spatial abstraction for training accurate machine learning models: A case study in smart thermostats. *Energy and Buildings*, page 113377, 2023.
- [115] Chaobo Zhang, Junyang Li, Yang Zhao, Tingting Li, Qi Chen, Xuejun Zhang, and Weikang Qiu. Problem of data imbalance in building energy load prediction: Concept, influence, and solution. *Applied Energy*, 297: 117139, <https://doi.org/10.1016/j.apenergy.2021.117139>, 2021.
- [116] Zhengbo Zou, Xinran Yu, and Semiha Ergan. Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network. *Building and Environment*, 168:106535, 2020.
- [117] Benoit Delcroix, Jérôme Le Ny, Michel Bernier, Muhammad Azam, Bingrui Qu, and Jean-Simon Venne. Autoregressive neural networks with exogenous variables for indoor temperature prediction in buildings. In *Building Simulation*, volume 14, pages 165–178. Springer, 2021.
- [118] Felix Bünnig, Adrian Schalbetter, Ahmed Aboudonia, Mathias Hudoba de Badyn, Philipp Heer, and John Lygeros. Input Convex Neural Networks for Building MPC. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 251–262. PMLR, 07 – 08 June 2021. URL <https://proceedings.mlr.press/v144/bunning21a.html>.
- [119] Xinyi Li and Runming Yao. Modelling heating and cooling energy demand for building stock using a hybrid approach. *Energy and Buildings*, 235:110740, <https://doi.org/10.1016/j.enbuild.2021.110740>, 2021.
- [120] Mehdi Maasoumy, M Razmara, M Shahbakhti, and A Sangiovanni Vincentelli. Handling model uncertainty in model predictive control for energy efficient buildings. *Energy and Buildings*, 77:377–392, 2014.
- [121] Mehdi Maasoumy, Meysam Razmara, Mahdi Shahbakhti, and Alberto Sangiovanni Vincentelli. Selecting building predictive control based on model uncertainty. In *2014 American Control Conference*, pages 404–411. IEEE, 2014.
- [122] Yanfei Li, Zheng O’Neill, Liang Zhang, Jianli Chen, Piljae Im, and Jason DeGraw. Grey-box modeling and application for building energy simulations-A critical review. *Renewable and Sustainable Energy Reviews*, 146:111174, 2021.
- [123] Javier Arroyo, Fred Spiessens, and Lieve Helsen. Identification of multi-zone grey-box building models for use in model predictive control. *Journal of Building Performance Simulation*, 13(4):472–486, 2020.
- [124] Mohammad Haris Shamsi, Usman Ali, Eleni Mangina, and James O’Donnell. A framework for uncertainty quantification in building heat demand simulations using reduced-order grey-box energy models. *Applied Energy*, 275:115141, <https://doi.org/10.1016/j.apenergy.2020.115141>, 2020.
- [125] Kenneth R Muske and James B Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2): 262–287, 1993.
- [126] Yashen Lin, Timothy Middelkoop, and Prabir Barooah. Issues in identification of control-oriented thermal models of zones in multi-zone buildings. In *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pages 6932–6937. IEEE, 2012.
- [127] Thomas Berthou, Pascal Stabat, Raphael Salvazet, and Dominique Marchio. Development and validation of a gray box model to predict thermal behavior of occupied office buildings. *Energy and Buildings*, 74:91–100, <https://doi.org/10.1016/j.enbuild.2014.01.038>, 2014.
- [128] Samuel F Fux, Araz Ashouri, Michael J Benz, and Lino Guzzella. EKF based self-adaptive thermal model for a passive house. *Energy and Buildings*, 68:811–817, 2014.
- [129] Felix Bünnig, Benjamin Huber, Adrian Schalbetter, Ahmed Aboudonia, Mathias Hudoba de Badyn, Philipp Heer, Roy S Smith, and John Lygeros. Physics-informed linear regression is competitive with two Machine Learning methods in residential building MPC. *Applied Energy*, 310:118491, 2022.
- [130] Mehdi Maasoumy, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. Model-based hierarchical optimal control design for HVAC systems. In *Dynamic Systems and Control Conference*, volume 54754, pages 271–278, 2011.
- [131] Feng Sheng and Li Jia. Short-term load forecasting based on SARIMAX-LSTM. In *2020 5th International Conference on Power and Renewable Energy (ICPRE)*, pages 90–94. IEEE, 2020.
- [132] Francesco Massa Gray and Michael Schmidt. A hybrid approach to thermal building modelling using a combination of Gaussian processes and grey-box models. *Energy and Buildings*, 165:56–63, <https://doi.org/10.1016/j.enbuild.2018.01.039>, 2018.

Bibliography

- [133] Gargya Gokhale, Bert Claessens, and Chris Develder. Physics informed neural networks for control oriented thermal modeling of buildings. *Applied Energy*, 314:118852, 2022.
- [134] Yongbao Chen, Qiguo Yang, Zhe Chen, Chengchu Yan, Shu Zeng, and Mingkun Dai. Physics-informed neural networks for building thermal modeling and demand response control. *Building and Environment*, page 110149, 2023.
- [135] Srinarayana Nagarathinam, Yashovardhan S Chati, Malini Pooni Venkat, and Arunchandar Vasan. PACMAN: physics-aware control MANager for HVAC. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 11–20, 2022.
- [136] Xuezheng Wang and Bing Dong. Physics-informed Hierarchical Data-driven Predictive Control for Building HVAC Systems to Achieve Energy and Health Nexus. *Energy and Buildings*, page 113088, 2023.
- [137] Ján Drgoňa, Aaron R Tuor, Vikas Chandan, and Draguna L Vrabie. Physics-constrained deep learning of multi-zone building thermal dynamics. *Energy and Buildings*, 243:110992, <https://doi.org/10.1016/j.enbuild.2021.110992>, 2021.
- [138] Ahmad Mohammadshirazi, Aida Nadafian, Amin Karimi Monsefi, Mohammad H Rafiei, and Rajiv Ramnath. Novel Physics-Based Machine-Learning Models for Indoor Air Quality Approximations. *arXiv preprint arXiv:2308.01438*, 2023.
- [139] Tianqi Xiao and Fengqi You. Building thermal modeling and model predictive control with physically consistent deep learning for decarbonization and energy optimization. *Applied Energy*, 342:121165, 2023.
- [140] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [141] Empa. NEST. <https://www.empa.ch/web/nest/overview>, 2021. Accessed: 21.08.2023.
- [142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [143] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.
- [144] Arnab Chatterjee and Dolaana Khovalyg. Dynamic indoor thermal environment using reinforcement learning-based controls: Opportunities and challenges. *Building and Environment*, page 110766, 2023.
- [145] Charalampos Vallianos, Andreas Athienitis, and Benoît Delcroix. Automatic generation of multi-zone RC models using smart thermostat data from homes. *Energy and Buildings*, 277:112571, 2022.
- [146] Maarten Schoukens. Improved initialization of state-space artificial neural networks. In *2021 European Control Conference (ECC)*, pages 1913–1918. IEEE, 2021.
- [147] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.
- [148] Giuseppe Pinto, Zhe Wang, Abhishek Roy, Tianzhen Hong, and Alfonso Capozzoli. Transfer learning for smart buildings: A critical review of algorithms, applications, and future perspectives. *Advances in Applied Energy*, 5:100084, 2022.
- [149] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.
- [150] Feiyu Li, Zhibo Wan, Thomas Koch, Guokuan Zan, Mengjiao Li, Zhonghai Zheng, and Bo Liang. Improving the accuracy of multi-step prediction of building energy consumption based on EEMD-PSO-Informer and long-time series. *Computers and Electrical Engineering*, 110:108845, 2023.
- [151] Edward Curry, James O'Donnell, Edward Corry, Souleiman Hasan, Marcus Keane, and Seán O'Riain. Linking building data in the cloud: Integrating cross-domain building data using linked data. *Advanced Engineering Informatics*, 27(2):206–219, 2013.
- [152] Wooyoung Jung and Farrokh Jazizadeh. Comparative assessment of HVAC control strategies using personal thermal comfort and sensitivity models. *Building and Environment*, 158:104–119, 2019.

- [153] Kingsley Nweye, Bo Liu, Peter Stone, and Zoltan Nagy. Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings. *Energy and AI*, 10:100202, 2022.
- [154] Loris Di Natale, Yingzhao Lian, Emilio T Maddalena, Jicheng Shi, and Colin N Jones. Lessons Learned from Data-Driven Building Control Experiments: Contrasting Gaussian Process-based MPC, Bilevel DeePC, and Deep Reinforcement Learning. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1111–1117. IEEE, 2022.
- [155] Joachim Verhelst, Geert Van Ham, Dirk Saelens, and Lieve Helsen. Model selection for continuous commissioning of HVAC-systems in office buildings: A review. *Renewable and Sustainable Energy Reviews*, 76: 673–686, 2017.
- [156] Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- [157] Jicheng Shi and Colin N Jones. Efficient Recursive Data-enabled Predictive Control: an Application to Consistent Predictors. *arXiv preprint arXiv:2309.13755*, 2023.
- [158] Petr Stluka, Girija Parthasarathy, Steve Gabel, and Tariq Samad. Architectures and algorithms for building automation—An industry view. *Intelligent Building Control Systems: A Survey of Modern Building Control and Sensing Strategies*, pages 11–43, 2018.
- [159] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.
- [160] Marcello Fiducioso, Sebastian Curi, Benedikt Schumacher, Markus Gwerder, and Andreas Krause. Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. *arXiv preprint arXiv:1906.12086*, 2019.
- [161] Wenjie Xu, Bratislav Svetožarević, Loris Di Natale, Philipp Heer, and Colin N Jones. Data-driven adaptive building thermal controller tuning with constraints: A primal-dual contextual Bayesian optimization approach. *arXiv preprint arXiv:2310.00758*, 2023.
- [162] Wenjie Xu, Yuning Jiang, Bratislav Svetožarević, and Colin N Jones. Primal-Dual Contextual Bayesian Optimization for Control System Online Optimization with Time-Average Constraints. *arXiv preprint arXiv:2304.06104*, 2023.
- [163] Andreas Krause and Cheng Ong. Contextual gaussian process bandit optimization. *Advances in neural information processing systems*, 24, 2011.
- [164] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, pages 1–35, 2021.
- [165] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.
- [166] Elias Goldsztejn, Tal Feiner, and Ronen Brafman. PTDL: Parameter Tuning using Deep Reinforcement Learning. *arXiv preprint arXiv:2306.10833*, 2023.
- [167] Myisha A Chowdhury and Qiugang Lu. A Novel Entropy-Maximizing TD3-based Reinforcement Learning for Automatic PID Tuning. In *2023 American Control Conference (ACC)*, pages 2763–2768. IEEE, 2023.
- [168] David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S Smith. Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis. *IEEE Transactions on Control Systems Technology*, 24(1):1–12, 2015.
- [169] Georgios D Kontes, Georgios I Giannakis, Víctor Sánchez, Pablo de Agustin-Camacho, Ander Romero-Amorrortu, Natalia Panagiotidou, Dimitrios V Rovas, Simone Steiger, Christopher Mutschler, and Gunnar Gruen. Simulation-based evaluation and optimization of control strategies in buildings. *Energies*, 11(12): 3376, 2018.
- [170] Samuel Privara, Jiří Cigler, Zdeněk Váňa, Frauke Oldewurtel, Carina Sagerschnig, and Eva Žáčková. Building modeling as a crucial part for building predictive control. *Energy and Buildings*, 56:8–22, 2013.
- [171] Gianluca Serale, Massimo Fiorentini, Alfonso Capozzoli, Daniele Bernardini, and Alberto Bemporad. Model predictive control (MPC) for enhancing building and HVAC system energy efficiency: Problem formulation, applications and opportunities. *Energies*, 11(3):631, 2018.
- [172] Samuel Privara, Jiří Cigler, Zdeněk Váňa, Frauke Oldewurtel, and Eva Žáčková. Use of partial least squares within the control relevant identification for buildings. *Control Engineering Practice*, 21(1):113–121, 2013.

Bibliography

- [173] Emilio T Maddalena, Silvio A Mueller, Rafael M dos Santos, Christophe Salzmänn, and Colin N Jones. Experimental data-driven model predictive control of a hospital HVAC system during regular use. *Energy and Buildings*, 271:112316, 2022.
- [174] Hao Huang, Lei Chen, and Eric Hu. Model predictive control for energy-efficient buildings: An airport terminal building study. In *11th IEEE International Conference on Control & Automation (ICCA)*, pages 1025–1030. IEEE, 2014.
- [175] Damien Picard, Ján Drgoňa, Michal Kvasnica, and Lieve Helsén. Impact of the controller model complexity on model predictive control performance for buildings. *Energy and Buildings*, 152:739–751, 2017.
- [176] Frauke Oldewurtel, Alessandra Parisio, Colin N Jones, Dimitrios Gyalistras, Markus Gwerder, Vanessa Stauch, Beat Lehmann, and Manfred Morari. Use of model predictive control and weather forecasts for energy efficient building climate control. *Energy and buildings*, 45:15–27, 2012.
- [177] Yan Zhang, Xuemei Bai, Franklin P Mills, and John CV Pezzey. Rethinking the role of occupant behavior in building energy performance: A review. *Energy and Buildings*, 172:279–294, 2018.
- [178] Salvatore Carlucci, Marilena De Simone, Steven K Firth, Mikkel B Kjærgaard, Romana Markovic, Mohammad Saiedur Rahaman, Masab Khalid Annaqeeb, Silvia Biandrate, Anooshmita Das, Jakub Wladyslaw Dziedzic, et al. Modeling occupant behavior in buildings. *Building and Environment*, 174:106768, 2020.
- [179] Da Yan, William O’Brien, Tianzhen Hong, Xiaohang Feng, H Burak Gunay, Farhang Tahmasebi, and Ardeshtir Mahdavi. Occupant behavior modeling for building performance simulation: Current state and future challenges. *Energy and buildings*, 107:264–278, 2015.
- [180] Jan Drgono, Karol Kis, Aaron Tuor, Draguna Vrabie, and Martin Klauco. Deep learning alternative to explicit model predictive control for unknown nonlinear systems. *arXiv preprint arXiv:2011.03699*, 2020.
- [181] Emilio Tanowe Maddalena, CG da S Moraes, Gerri Waltrich, and Colin N Jones. A neural network architecture to learn explicit MPC controllers from data. *IFAC-PapersOnLine*, 53(2):11362–11367, 2020.
- [182] Steven Chen, Kelsey Saulnier, Nikolay Atanasov, Daniel D Lee, Vijay Kumar, George J Pappas, and Manfred Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American control conference (ACC)*, pages 1520–1527. IEEE, 2018.
- [183] Ján Drgoňa, Aaron Tuor, Elliott Skomski, Soumya Vasisht, and Draguna Vrabie. Deep learning explicit differentiable predictive control laws for buildings. *IFAC-PapersOnLine*, 54(6):14–19, 2021.
- [184] Jan C Willems, Paolo Rapisarda, Ivan Markovsky, and Bart LM De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.
- [185] Ivan Markovsky and Florian Dörfler. Behavioral systems theory in data-driven analysis, signal processing, and control. *Annual Reviews in Control*, 52:42–64, 2021.
- [186] Venkatesh Chinde, Yashen Lin, and Matthew J Ellis. Data-enabled predictive control for building HVAC systems. *Journal of Dynamic Systems, Measurement, and Control*, 144(8):081001, 2022.
- [187] Edward O’Dwyer, Eric C Kerrigan, Paola Falugi, Marta Zagorowska, and Nilay Shah. Data-driven predictive control with improved performance using segmented trajectories. *IEEE Transactions on Control Systems Technology*, 2022.
- [188] Yingzhao Lian, Jicheng Shi, Manuel Koch, and Colin Neil Jones. Adaptive robust data-driven building control via bilevel reformulation: An experimental result. *IEEE Transactions on Control Systems Technology*, 2023.
- [189] Jeremy Coulson, John Lygeros, and Florian Dörfler. Distributionally robust chance constrained data-enabled predictive control. *IEEE Transactions on Automatic Control*, 67(7):3289–3304, 2021.
- [190] Manuel Koch and Colin N Jones. Comparison of behavioral systems theory and conventional linear models for predicting building zone temperature in long-term in situ measurements. *arXiv preprint arXiv:2302.04063*, 2023.
- [191] Manuel Koch and Colin N Jones. A comparison of methods to eliminate regularization weight tuning from data-enabled predictive control. *arXiv preprint arXiv:2305.00807*, 2023.
- [192] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3: 269–296, 2020.
- [193] Shiyu Yang, Man Pun Wan, Wanyu Chen, Bing Feng Ng, and Deqing Zhai. An adaptive robust model predictive control for indoor climate optimization and uncertainties handling in buildings. *Building and Environment*, 163:106326, 2019.

- [194] Tingting Zeng and Prabir Barooah. An autonomous MPC scheme for energy-efficient control of building HVAC systems. In *2020 American control conference (ACC)*, pages 4213–4218. IEEE, 2020.
- [195] Ruixin Lv, Zhongyuan Yuan, Bo Lei, Jiacheng Zheng, and Xiuqing Luo. Model predictive control with adaptive building model for heating using the hybrid air-conditioning system in a railway station. *Energies*, 14(7):1996, 2021.
- [196] Shiyu Yang, Man Pun Wan, Wanyu Chen, Bing Feng Ng, and Swapnil Dubey. Model predictive control with adaptive machine-learning-based model for building energy efficiency and comfort optimization. *Applied Energy*, 271:115147, 2020.
- [197] Marko Tanaskovic, David Sturzenegger, Roy Smith, and Manfred Morari. Robust adaptive model predictive building climate control. *Ifac-Papersonline*, 50(1):1871–1876, 2017.
- [198] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [199] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.
- [200] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [201] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.
- [202] Hao Ju, Rongshun Juan, Randy Gomez, Keisuke Nakamura, and Guangliang Li. Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, 4(12):1077–1087, 2022.
- [203] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.
- [204] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.
- [205] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.
- [206] Mengjie Han, Ross May, Xingxing Zhang, Xinru Wang, Song Pan, Da Yan, Yuan Jin, and Liguoxu Xu. A review of reinforcement learning methodologies for controlling occupant comfort in buildings. *Sustainable Cities and Society*, 51:101748, 2019.
- [207] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2 : Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.
- [208] Zhiang Zhang and Khee Poh Lam. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In *Proceedings of the 5th Conference on Systems for Built Environments*, pages 148–157, 2018.
- [209] Chao Huang, Hongcai Zhang, Long Wang, Xiong Luo, and Yonghua Song. Mixed deep reinforcement learning considering discrete-continuous hybrid action space for smart home energy management. *Journal of Modern Power Systems and Clean Energy*, 10(3):743–754, 2022.
- [210] Adam Nagy, Hussain Kazmi, Farah Cheaib, and Johan Driesen. Deep reinforcement learning for optimal control of space heating. *arXiv preprint arXiv:1805.03777*, 2018.
- [211] Giuseppe Tommaso Costanzo, Sandro Iacovella, Frederik Ruelens, Tim Leurs, and Bert J Claessens. Experimental analysis of data-driven control for a building heating system. *Sustainable Energy, Grids and Networks*, 6:81–90, 2016.

Bibliography

- [212] Roberto Rocchetta, Lorenzo Nespoli, Vasco Medici, Saverio Basso, Marco Derboni, and Matteo Salani. Rule-based deep reinforcement learning for optimal control of electrical batteries in an energy community. In *Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023)*, 2023.
- [213] Xinlei Zhou, Shan Xue, Han Du, and Zhenjun Ma. Optimization of building demand flexibility using reinforcement learning and rule-based expert systems. *Applied Energy*, 350:121792, 2023.
- [214] Frederik Ruelens, Bert J Claessens, Stijn Vandael, Bart De Schutter, Robert Babuška, and Ronnie Belmans. Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Transactions on Smart Grid*, 8(5):2149–2159, 2016.
- [215] Minghao Chen, Zhiyuan Xie, Yi Sun, and Shunlin Zheng. The predictive management in campus heating system based on deep reinforcement learning and probabilistic heat demands forecasting. *Applied Energy*, 350:121710, 2023.
- [216] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.
- [217] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.
- [218] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.
- [219] Thiago D Simão, Nils Jansen, and Matthijs TJ Spaan. AlwaysSafe: Reinforcement learning without safety constraint violations during training. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2021.
- [220] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.
- [221] Oscar De Somer, Ana Soares, Koen Vanthournout, Fred Spiessens, Tristan Kuijpers, and Koen Vossen. Using reinforcement learning for demand response of domestic hot water buffers: A real-life demonstration. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–7. IEEE, 2017.
- [222] Qisong Yang, Thiago D Simão, Nils Jansen, Simon H Tindemans, and Matthijs TJ Spaan. Reinforcement Learning by Guided Safe Exploration. *arXiv preprint arXiv:2307.14316*, 2023.
- [223] Hongzi Mao, Malte Schwarzkopf, Hao He, and Mohammad Alizadeh. Towards safe online reinforcement learning in computer systems. In *NeurIPS Machine Learning for Systems Workshop*, 2019.
- [224] Rolando Bautista-Montesano, Renato Galluzzi, Kangrui Ruan, Yongjie Fu, and Xuan Di. Autonomous navigation at unsignalized intersections: A coupled reinforcement learning and model predictive control approach. *Transportation Research Part C: Emerging Technologies*, 139:103662, 2022.
- [225] Hui Hwang Goh, Yifeng Huang, Chee Shen Lim, Dongdong Zhang, Hui Liu, Wei Dai, Tonni Agustiono Kurniawan, and Saifur Rahman. An Assessment of Multi-Stage Reward Function Design for Deep Reinforcement Learning-Based Microgrid Energy Management. *IEEE Transactions on Smart Grid*, 2022.
- [226] Ruihang Wang, Xinyi Zhang, Xin Zhou, Yonggang Wen, and Rui Tan. Toward physics-guided safe deep reinforcement learning for green data center cooling control. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, pages 159–169. IEEE, 2022.
- [227] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [228] Alexander W Goodall and Francesco Belardinelli. Approximate Shielding of Atari Agents for Safe Exploration. *arXiv preprint arXiv:2304.11104*, 2023.
- [229] Linhai Xie, Sen Wang, Stefano Rosa, Andrew Markham, and Niki Trigoni. Learning with training wheels: speeding up training with a simple controller for deep reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6276–6283. IEEE, 2018.
- [230] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *International conference on machine learning*, pages 7953–7963. PMLR, 2020.

- [231] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.
- [232] Ramij R Hossain, Tianzhixi Yin, Yan Du, Renke Huang, Jie Tan, Wenhao Yu, Yuan Liu, and Qiuhua Huang. Efficient Learning of Voltage Control Strategies via Model-based Deep Reinforcement Learning. *arXiv preprint arXiv:2212.02715*, 2022.
- [233] Scott Jeen, Alessandro Abate, and Jonathan M Cullen. Low Emission Building Control with Zero-Shot Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14259–14267, 2023.
- [234] Xianzhong Ding, Alberto Cerpa, and Wan Du. Multi-zone HVAC Control with Model-Based Deep Reinforcement Learning. *arXiv preprint arXiv:2302.00725*, 2023.
- [235] Alekh Agarwal, Sham Kakade, and Lin F Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, pages 67–83. PMLR, 2020.
- [236] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.
- [237] Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.
- [238] Bingqing Chen, Zicheng Cai, and Mario Bergés. Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy. In *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 316–325, 2019.
- [239] Javier Arroyo, Carlo Manna, Fred Spiessens, and Lieve Helsen. Reinforced model predictive control (RL-MPC) for building energy management. *Applied Energy*, 309:118346, <https://doi.org/10.1016/j.apenergy.2021.118346>, 2022.
- [240] Simeng Liu and Gregor P Henze. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis. *Energy and buildings*, 38(2):148–161, 2006.
- [241] Tyler Westenbroek, Jacob Levy, and David Fridovich-Keil. Feedback is All You Need: Real-World Reinforcement Learning with Approximate Physics-Based Models. *arXiv preprint arXiv:2307.08168*, 2023.
- [242] Arash Bahari Kordabad, Rafal Wisniewski, and Sebastien Gros. Safe Reinforcement Learning Using Wasserstein Distributionally Robust MPC and Chance Constraint. *IEEE Access*, 2022.
- [243] Peixiao Fan, Jun Yang, Song Ke, Yuxin Wen, Yonghui Li, and Lilong Xie. Load frequency control strategy for islanded multimicrogrids with V2G dependent on learning-based model predictive control. *IET Generation, Transmission & Distribution*, 2023.
- [244] Daniel Mayfrank, Alexander Mitsos, and Manuel Dahmen. End-to-End Reinforcement Learning of Koopman Models for Economic Nonlinear MPC. *arXiv preprint arXiv:2308.01674*, 2023.
- [245] Bingqing Chen, Priya L Donti, Kyri Baker, J Zico Kolter, and Mario Bergés. Enforcing policy feasibility constraints through differentiable projection for energy optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pages 199–210, 2021.
- [246] Qingang Zhang, Muhammad Haiqal Bin Mahbod, Chin-Boon Chng, Poh-Seng Lee, and Chee-Kong Chui. Residual Physics and Post-Posed Shielding for Safe Deep Reinforcement Learning Method. *IEEE Transactions on Cybernetics*, 2022.
- [247] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.
- [248] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.
- [249] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [250] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Bibliography

- [251] Ruoxi Jia, Ming Jin, Kaiyu Sun, Tianzhen Hong, and Costas Spanos. Advanced building control via deep reinforcement learning. *Energy Procedia*, 158:6158–6163, 2019.
- [252] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.
- [253] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.
- [254] Debaprasad Dutta and Simant R Upreti. A reinforcement learning-based transformed inverse model strategy for nonlinear process control. *Computers & Chemical Engineering*, 178:108386, 2023.
- [255] Jinxiong Lu, Gokhan Alcan, and Ville Kyrki. Integrating Expert Guidance for Efficient Learning of Safe Overtaking in Autonomous Driving Using Deep Reinforcement Learning. *arXiv preprint arXiv:2308.09456*, 2023.
- [256] Loris Di Natale, Bratislav Svetozarevic, Philipp Heer, and CN Jones. Deep Reinforcement Learning for room temperature control: a black-box pipeline from data to policies. In *Journal of Physics: Conference Series*, volume 2042, page 012004. IOP Publishing, 2021.
- [257] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Sim2Real in robotics and automation: Applications and challenges. *IEEE transactions on automation science and engineering*, 18(2):398–400, 2021.
- [258] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.
- [259] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.
- [260] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.
- [261] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.
- [262] Lei Yang, Zoltan Nagy, Philippe Goffin, and Arno Schlueter. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 156:577–586, 2015.
- [263] OpenAI. Part 2: Kinds of RL Algorithms, 2018. URL https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#a-taxonomy-of-rl-algorithms. Accessed: 25.11.2022.
- [264] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- [265] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [266] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.
- [267] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.
- [268] Hado Hasselt. Double Q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.
- [269] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the Variance of the Adaptive Learning Rate and Beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020.
- [270] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A Highly Modularized Deep Reinforcement Learning Library. *Journal of Machine Learning Research*, 23(267):1–6, 2022. URL <http://jmlr.org/papers/v23/21-1127.html>.
- [271] José Vázquez-Canteli, Jérôme Kömpf, and Zoltán Nagy. Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted Q-iteration. *Energy Procedia*, 122:415–420, 2017.

- [272] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [273] William Valladares, Marco Galindo, Jorge Gutierrez, Wu-Chieh Wu, Kuo-Kai Liao, Jen-Chung Liao, Kuang-Chin Lu, and Chi-Chuan Wang. Energy optimization associated with thermal comfort and indoor air control via a deep reinforcement learning algorithm. *Building and Environment*, 155:105–117, 2019.
- [274] Moritz Frahm, Thomas Dengiz, Philipp Zwickel, Heiko Maaß, Jörg Matthes, and Veit Hagenmeyer. Occupant-oriented demand response with multi-zone thermal building control. *Applied Energy*, 347:121454, 2023.
- [275] Yan Du, Fangxing Li, Kuldeep Kurte, Jeffrey Munk, and Helia Zandi. Demonstration of Intelligent HVAC Load Management With Deep Reinforcement Learning: Real-World Experience of Machine Learning in Demand Control. *IEEE Power and Energy Magazine*, 20(3):42–53, 2022.
- [276] Bram De Cooman, Johan Suykens, and Andreas Ortseifen. Enforcing Hard State-Dependent Action Bounds on Deep Reinforcement Learning Policies. In *Machine Learning, Optimization, and Data Science: 8th International Workshop, LOD 2022, Certosa di Pontignano, Italy, September 19–22, 2022, Revised Selected Papers, Part II*, pages 193–218. Springer, 2023.
- [277] Peng Zhang, Jianye Hao, Weixun Wang, Hongyao Tang, Yi Ma, Yihai Duan, and Yan Zheng. KoGuN: accelerating deep reinforcement learning via integrating human suboptimal knowledge. *arXiv preprint arXiv:2002.07418*, 2020.
- [278] Ramij R Hossain, Kaveri Mahapatra, Qiuhua Huang, and Renke Huang. Physics-informed Deep Reinforcement Learning-based Adaptive Generator Out-of-step Protection for Power Systems. In *2023 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2023.
- [279] Sebastien Gros, Mario Zanon, and Alberto Bemporad. Safe reinforcement learning via projection on a safe set: How to achieve optimality? *IFAC-PapersOnLine*, 53(2):8076–8081, 2020.
- [280] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.
- [281] Ruihang Wang, Xinyi Zhang, Xin Zhou, Yonggang Wen, and Rui Tan. Toward Physics-Guided Safe Deep Reinforcement Learning for Green Data Center Cooling Control. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPs)*, pages 159–169. IEEE, 2022.
- [282] Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.
- [283] Nikos Komodakis and Jean-Christophe Pesquet. Playing with duality: An overview of recent primal? dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015.
- [284] Xuyang Zhong, Zhiang Zhang, Ruijun Zhang, and Chenlu Zhang. End-to-End Deep Reinforcement Learning Control for HVAC Systems in Office Buildings. *Designs*, 6(3):52, 2022.
- [285] Xuebo Liu, Yingying Wu, Bo Liu, and Hongyu Wu. A Multi-Agent Deep Deterministic Policy Gradient Method for Multi-Zone HVAC Control. In *2023 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2023.
- [286] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2022.
- [287] Annie Wong, Thomas Bäck, Anna V Kononova, and Aske Plaat. Deep multiagent reinforcement learning: Challenges and directions. *Artificial Intelligence Review*, 56(6):5023–5056, 2023.
- [288] Jiechao Gao, Wenpeng Wang, Fateme Nikseresht, Viswajith Govinda Rajan, and Bradford Campbell. PFDRL: Personalized Federated Deep Reinforcement Learning for Residential Energy Management. In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 402–411, 2023.
- [289] Zixin Jiang and Bing Dong. OCCUPIED: Long-term Field Experiment Results from an Occupant-Centric Control in an Office Building. *Energy and Buildings*, page 113435, 2023.
- [290] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Bibliography

- [291] Seongkwon Cho and Cheol Soo Park. Rule reduction for control of a building cooling system using explainable AI. *Journal of Building Performance Simulation*, 15(6):832–847, 2022.
- [292] Thomas Hickling, Abdelhafid Zenati, Nabil Aouf, and Phillippa Spencer. Explainability in Deep Reinforcement Learning, a Review into Current Methods and Applications. *arXiv preprint arXiv:2207.01911*, 2022.
- [293] Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, pages 1–66, 2022.
- [294] Daniel Gedon, Niklas Wahlström, Thomas B Schön, and Lennart Ljung. Deep state space models for nonlinear system identification. *IFAC-PapersOnLine*, 54(7):481–486, 2021.
- [295] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [296] Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*, 1974.
- [297] Lennart Ljung, Carl Andersson, Koen Tiels, and Thomas B Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020.
- [298] Marco Forgone and Dario Piga. Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, 59:69–81, 2021.
- [299] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [300] Gerben I Beintema, Maarten Schoukens, and Roland Tóth. Deep subspace encoders for nonlinear system identification. *Automatica*, 156:111210, 2023.
- [301] Daniele Masti and Alberto Bemporad. Learning nonlinear state–space models using autoencoders. *Automatica*, 129:109666, 2021.
- [302] Wilson J Rugh. *Linear system theory*. Prentice-Hall, Inc., 1996.
- [303] IS Khalil, JC Doyle, and K Glover. *Robust and optimal control*. Prentice hall, 1996.
- [304] Jan C Schulze and Alexander Mitsos. Data-Driven Nonlinear Model Reduction Using Koopman Theory: Integrated Control Form and NMPC Case Study. *IEEE Control Systems Letters*, 6:2978–2983, 2022.
- [305] Jan C Schulze, Danimir T Doncevic, and Alexander Mitsos. Identification of MIMO Wiener-type Koopman models for data-driven model reduction using deep learning. *Computers & Chemical Engineering*, 161:107781, 2022.
- [306] Hyungjin Choi, Valerio De Angelis, and Yuliya Preger. Data-driven Battery Modeling based on Koopman Operator Approximation using Neural Network. In *2023 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2023.
- [307] Michael A Lones. How to avoid machine learning pitfalls: a guide for academic researchers. *arXiv preprint arXiv:2108.02497*, 2021.
- [308] Maurício C De Oliveira, Jacques Bernussou, and José C Geromel. A new discrete-time robust stability condition. *Systems & control letters*, 37(4):261–265, 1999.
- [309] Lennart Ljung. *System identification toolbox: User's guide*. Citeseer, 1995.
- [310] Peter Van Overschee and Bart De Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.
- [311] Michel Verhaegen and Patrick Dewilde. Subspace model identification part 2. Analysis of the elementary output-error state-space model identification algorithm. *International journal of control*, 56(5):1211–1241, 1992.
- [312] Wallace E Larimore. Canonical variate analysis in identification, filtering, and adaptive control. In *29th IEEE Conference on Decision and control*, pages 596–604. IEEE, 1990.

- [313] Peter Van Overschee and Bart De Moor. A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1853–1864, 1995.
- [314] S Joe Qin, Weilu Lin, and Lennart Ljung. A novel subspace identification approach with enforced causal models. *Automatica*, 41(12):2043–2053, 2005.
- [315] S Joe Qin and Lennart Ljung. Parallel QR implementation of subspace identification with parsimonious models. *IFAC Proceedings Volumes*, 36(16):1591–1596, 2003.
- [316] Gabriele Pannocchia and Mirco Calosi. A predictor form PARSIMonious algorithm for closed-loop subspace identification. *Journal of Process Control*, 20(4):517–524, 2010.
- [317] Wen-bing Huang, Le le Cao, Fuchun Sun, Deli Zhao, Huaping Liu, and Shanshan Yu. Learning Stable Linear Dynamical Systems with the Weighted Least Square Method. In *IJCAI*, volume 1599, page 1605, 2016.
- [318] Byron Boots, Geoffrey J Gordon, and Sajid Siddiqi. A constraint generation approach to learning stable linear dynamical systems. *Advances in neural information processing systems*, 20, 2007.
- [319] Seth L Lacy and Dennis S Bernstein. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on automatic control*, 48(7):1259–1263, 2003.
- [320] Nicolas Gillis, Michael Karow, and Punit Sharma. A note on approximating the nearest stable discrete-time descriptor systems with fixed rank. *Applied Numerical Mathematics*, 148:131–139, 2020.
- [321] Wouter Jongeneel, Tobias Sutter, and Daniel Kuhn. Efficient learning of a linear dynamical system with stability guarantees. *IEEE Transactions on Automatic Control*, 2022.
- [322] Ján Drgoňa, Aaron R Tuor, Vikas Chandan, and Draguna L Vrabie. Physics-constrained deep learning of multi-zone building thermal dynamics. *Energy and Buildings*, 243:110992, 2021.
- [323] J Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. *Advances in neural information processing systems*, 32, 2019.
- [324] Max Revay, Ruigang Wang, and Ian R Manchester. Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness. *IEEE Transactions on Automatic Control*, 2023.
- [325] Aleksandar Haber and Michel Verhaegen. Subspace identification of large-scale interconnected systems. *IEEE Transactions on Automatic Control*, 59(10):2754–2759, 2014.
- [326] Paolo Massioni and Michel Verhaegen. Subspace identification of circulant systems. *Automatica*, 44(11):2825–2833, 2008.
- [327] Paolo Massioni and Michel Verhaegen. Subspace identification of distributed, decomposable systems. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3364–3369. IEEE, 2009.
- [328] Fabian Zapf and Thomas Wallek. Gray-box surrogate models for flash, distillation and compression units of chemical processes. *Computers & Chemical Engineering*, 155:107510, 2021.
- [329] Ololade O Obadina, Mohamed A Thaha, Zaharuddin Mohamed, and M Hasan Shaheed. Grey-box modelling and fuzzy logic control of a Leader-Follower robot manipulator system: A hybrid Grey Wolf-Whale Optimisation approach. *ISA transactions*, 129:572–593, 2022.
- [330] Yann N Dauphin and Samuel Schoenholz. Metainit: Initializing learning by learning to initialize. *Advances in Neural Information Processing Systems*, 32, 2019.
- [331] Kartik Loya, Jake Buzhardt, and Phanindra Tallapragada. Koopman operator based predictive control with a data archive of observables. *ASME Letters in Dynamic Systems and Control*, pages 1–7, 2023.
- [332] Daniele Martinelli, Clara Lucía Galimberti, Ian R Manchester, Luca Furieri, and Giancarlo Ferrari-Trecate. Unconstrained Parametrization of Dissipative and Contracting Neural Ordinary Differential Equations. *arXiv preprint arXiv:2304.02976*, 2023.
- [333] Giorgos Mamakoukas, Orest Xherija, and Todd Murphey. Memory-efficient learning of stable linear dynamical systems for prediction and control. *Advances in Neural Information Processing Systems*, 33:13527–13538, 2020.
- [334] Bart De Moor, Peter De Gersem, Bart De Schutter, and Wouter Favoreel. DAISY: A database for identification of systems. *JOURNAL A*, 38(4):5, 1997.
- [335] Chengpu Yu, Lennart Ljung, and Michel Verhaegen. Identification of structured state-space models. *Automatica*, 90:54–61, 2018.

Bibliography

- [336] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.
- [337] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.
- [338] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.
- [339] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.
- [340] Muhammad Zakwan, Liang Xu, and Giancarlo Ferrari Trecate. Robust Classification Using Contractive Hamiltonian Neural ODEs. *IEEE Control Systems Letters*, 7:145–150, 2022.
- [341] Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming. *IEEE Transactions on Automatic Control*, 67(1):1–15, January 2022.
- [342] Clara Lucía Galimberti, Luca Furieri, Liang Xu, and Giancarlo Ferrari-Trecate. Hamiltonian Deep Neural Networks Guaranteeing Nonvanishing Gradients by Design. *IEEE Transactions on Automatic Control*, 68(5):3155–3162, 2023.
- [343] Hector Ramirez, Bernhard Maschke, and Daniel Sbarbaro. Irreversible port-Hamiltonian systems: A general formulation of irreversible processes with application to the CSTR. *Chemical Engineering Science*, 89:223–234, 2013.
- [344] Arjan Van der Schaft and Dimitri Jeltsema. On Energy Conversion in Port-Hamiltonian Systems. *arXiv preprint arXiv:2103.09116*, 2021.
- [345] Hector Ramirez, Bernhard Maschke, and Daniel Sbarbaro. Modelling and control of multi-energy systems: An irreversible port-Hamiltonian approach. *European journal of control*, 19(6):513–520, 2013.
- [346] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop*, 2017.
- [347] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [348] Bart Merema, Dirk Saelens, and Hilde Breesch. Demonstration of an MPC framework for all-air systems in non-residential buildings. *Building and Environment*, 217:109053, 2022.
- [349] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.
- [350] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL <https://github.com/fmfn/BayesianOptimization>.
- [351] Mahmoud Chilali and Pascal Gahinet. H/sub/spl infin//design with pole placement constraints: an lmi approach. *IEEE Transactions on automatic control*, 41(3):358–367, 1996.
- [352] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.
- [353] Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.
- [354] Josiah Willard Gibbs. A method of geometrical representation of the thermodynamic properties by means of surfaces. *The Collected Works of J. Willard Gibbs, Ph. D., LL. D.*, pages 33–54, 1957.



LORIS DI NATALE

Swiss

Objective: Using advanced Machine Learning tools, I aim to support the energy transition and tackle climate change. I am convinced that Artificial Intelligence, if and when applied carefully, has to play a major role in this fight.



+41 (0)79 900 29 05



loris.dinatale@netplus.ch



Planereuses 3
1943 Praz-de-Fort,
VS, Switzerland



[github](#) / [scholar](#)

SKILL HIGHLIGHTS

- Everyday Python user
- Artificial Intelligence
- Analytical mindset
- Problem solver
- Not afraid of challenges

LANGUAGES

- **English** – Bilingual
- **French** – Native
- **German** – Excellent listening and reading skills
- **Swiss German** – Good understanding

HOBBIES

- Drums & percussions
- Biking
- Skiing
- Football

▶ EDUCATION

Ph.D. in Electrical Engineering

Empa/EPFL (Ecole Polytechnique Fédérale de Lausanne)

Title: Fusing Pre-existing Knowledge and Machine Learning for Enhanced Building Thermal Modeling and Control

Feb. 2020 – Feb. 2024

*Lausanne,
Switzerland*

M.Sc. in Energy Management and Sustainability

EPFL (Ecole Polytechnique Fédérale de Lausanne)

Topics: Machine & Deep Learning, data analysis, statistical Machine Learning, mathematics of data

Sep. 2017 – Jul. 2019

*Lausanne,
Switzerland*

B.Sc. in Mathematics

EPFL (Ecole Polytechnique Fédérale de Lausanne)

Topics: Applied mathematics, probability & statistics, numerical analysis

Sep. 2014 – Jul. 2017

*Lausanne,
Switzerland*

▶ SELECTED ACADEMIC PROJECTS

Physically Consistent Neural Networks (PCNNs)

Showed how PCNNs simultaneously achieved state-of-the-art accuracy and compliance with the laws of thermodynamics when applied to building thermal modeling. Created an open-source toolbox of PCNNs.

This led to the publication of two first-author journal papers.

Deep Reinforcement Learning (DRL) for building control

In simulation, demonstrated how DRL agents can achieve near-optimality for building control and investigated how to accelerate their convergence.

Subsequently deployed these controllers in the physical building for validation.

This led to four conference papers (first author or equal contribution).

SIMBa: System Identification Methods leveraging Backpropagation

Examined how to use Machine Learning tools to solve complex system identification problems while maintaining stability and introducing prior system knowledge. Created an open-source toolbox for SIMBa.

This led to one journal and two conference papers (equal contribution).

▶ EXPERIENCE

Engineering intern

Ras Al Khaimah Waste Management Agency (RAKWMA)

Computed the carbon footprint of the company. Worked on waste trucks routing, co-processing of waste in cement factory kilns, assessment of the energy value of various types of waste, and food waste treatment.

Jul. – Sep. 2018

*Ras Al Khaimah,
UAE*