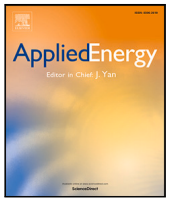
Applied Energy 309 (2022) 118346

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



Reinforced model predictive control (RL-MPC) for building energy management

Javier Arroyo [a,b,c,*], Carlo Manna [b,c], Fred Spiessens [b,c], Lieve Helsen [a,b]

[a] Department of Mechanical Engineering, KU Leuven, Heverlee, Belgium
[b] EnergyVille, Thor Park, Waterschei, Belgium
[c] Flemish Institute for Technological Research (VITO), Mol, Belgium

ARTICLE INFO

Keywords:
Model predictive control
Reinforcement learning
Reinforced model predictive control
Building automation
BOPTEST

ABSTRACT

Buildings need advanced control for the efficient and climate-neutral use of their energy systems. Model predictive control (MPC) and reinforcement learning (RL) arise as two powerful control techniques that have been extensively investigated in the literature for their application to building energy management. These methods show complementary qualities in terms of constraint satisfaction, computational demand, adaptability, and intelligibility, but usually a choice is made between both approaches. This paper compares both control approaches and proposes a novel algorithm called reinforced predictive control (RL-MPC) that merges their relative merits. First, the complementarity between RL and MPC is emphasized on a conceptual level by commenting on the main aspects of each method. Second, the RL-MPC algorithm is described that effectively combines features from each approach, namely state estimation, dynamic optimization, and learning. Finally, MPC, RL, and RL-MPC are implemented and evaluated in BOPTEST, a standardized simulation framework for the assessment of advanced control algorithms in buildings. The results indicate that pure RL cannot provide constraint satisfaction when using a control formulation equivalent to MPC and the same controller model for learning. The new RL-MPC algorithm can meet constraints and provide similar performance to MPC while enabling continuous learning and the possibility to deal with uncertain environments.

1. Introduction

To achieve a net-zero carbon building stock by 2050, the International Energy Agency estimates that direct building CO_2 emissions need to decrease by 50% and indirect building sector emissions need to decline 60% by 2030 [1]. Optimal control in buildings has gained popularity in recent years because of the substantial energy savings potential in the building sector. Building energy automation systems that use optimal predictive controllers can improve indoor comfort while lowering energy use locally and enabling demand response strategies to offer flexibility from the buildings to the electric grid [2–4]. Model predictive control (MPC) and reinforcement learning (RL) are two approaches for optimal control that pursue the same goal but follow different strategies. Both approaches show promising potential for their implementation in buildings but come along with relevant challenges.

The MPC approach is mainly studied within the field of *control theory*, and it is well known for its robustness and sample efficiency, but falls short in terms of adaptability. It usually requires a significant engineering effort to implement and configure a controller model suitable for optimization. Particularly, large and complex systems, like buildings, can easily lead to models comprising several variables and equations that give rise to non-convex optimization problems that are challenging to solve. Even if a solution to the optimization problem can be found, the controller model suffers from unavoidable model mismatch, and the system is usually exposed to forecast uncertainty. The latter can be addressed using stochastic MPC like robust or chance-constrained MPC, but these approaches require the description of the process uncertainties, which is usually hard to provide [5,6].

RL is mainly studied within the *machine learning* community. Contrarily to MPC, this advanced control approach is well known for its adaptability, but it has difficulties dealing with constraints, it needs extensive training, and it lacks intelligibility. The curse of dimensionality has always threatened the usability of RL. Nevertheless, the recent developments in deep learning have opened up the possibilities of these algorithms to applications as complex as building automation systems. Still, constraint satisfaction remains a relevant challenge in RL and is a research topic intensively investigated [7,8]. In the building sector,

* Corresponding author at: Department of Mechanical Engineering, KU Leuven, Heverlee, Belgium.
E-mail addresses: javier.arroyo@kuleuven.be (J. Arroyo), carlo.manna@vito.be (C. Manna), fred.spiessens@vito.be (F. Spiessens), lieve.helsen@kuleuven.be (L. Helsen).

https://doi.org/10.1016/j.apenergy.2021.118346
Received 29 September 2021; Received in revised form 19 November 2021; Accepted 5 December 2021
Available online 7 January 2022

RL has been mainly used at a supervisory level with a few set-points as controllable actions and with low dimensional observation spaces.

Usually, authors use either one approach or another based on their expertise and justify their choice by highlighting the strengths of their approach and stressing the weaknesses of the other. Both control families have advantages and disadvantages, and the complementarity between them is clear. On the one hand, MPC struggles with uncertainties, system complexity and long term prediction horizons while deep RL can naturally deal with the uncertainty of complex systems and tackle infinite prediction horizons. On the other hand, RL can difficultly satisfy constraints and lacks interpretability, while MPC can provide safety guarantees and intelligibility.

Although there is a clear potential for synergy between both families of methods, minimal efforts have been made to merge them and combine their relative merits. This research gap is not only seen in the application of building energy management. The control and machine learning communities keep evolving independently with a radically different notation natively adopted to formulate the same problem. Despite the parallel developments, some authors have pointed out that potential benefits may arise from their collaboration [9–12]. Combining these methodologies is a powerful way to integrate robust methods from the control theory with machine learning approaches that can exploit further information from real-time data [12,13].

The motivation for the work presented in this paper revolves around the question how RL and MPC can work together for the application of building energy management. While there appears to be universal agreement that benefits may arise from their combination, little has been done to develop methods that involve both algorithms working together. Moreover, the works that have investigated how these controllers can collaborate either have the algorithms working at different control levels e.g. in [14], or tested their collaborative methods with a low-dimensional state–action space [15]. There is no prior work that compares and merges MPC and RL for the same optimal control problem formulation for building energy management.

The aim of this paper is twofold: First, it is of interest to compare RL and MPC for their application to building energy management. This comparison begins on a conceptual level to underline the main similitudes and differences between the optimal control approaches. Then, the algorithms are tested when implemented separately. In the comparison, the MPC uses a gray-box model and the RL agent uses a value-based algorithm implemented by an equivalent control formulation and for the same building test case taken from the Building Optimization Testing (BOPTEST) framework [16]. The BOPTEST framework is a new environment meant to evaluate the performance of advanced controllers for building energy management systems in simulation. BOPTEST provides a menu of detailed building emulators and standardizes the way controllers are evaluated. Moreover, a BOPTEST-Gym wrapper [17] has been developed that facilitates the implementation of RL algorithms. The second aim of this paper is to propose a novel method named Reinforced Model Predictive Control (RL-MPC) that synergistically combines RL and MPC. The conceptual and practical comparison of MPC and RL motivates the development and formulation of this new RL-MPC algorithm. Differently to similar approaches proposed in the literature like [15], our method combines the MPC objective function with the RL agent value function while using a nonlinear controller model encoded from domain knowledge. This practice ensures the interoperability between both methods and enables truncation of the MPC optimization problem, which can become very complex even for fairly simple buildings. Finally, this new algorithm is explained and tested in the same BOPTEST building case.

Therefore, the main *novelty* of this paper is the introduction of RL-MPC, a control algorithm that combines methods from the control theory and the machine learning communities. This algorithm is tested in BOPTEST, a new standardized framework for advanced control of building energy management systems. Furthermore, the proposed RL-MPC algorithm can be used in different applications and domains e.g. complex industrial processes or energy markets. This study also brings control theory and machine learning closer by comparing and disentangling the principal differences between MPC and RL.

The main *limitations* of this paper relate to the lack of theoretical guarantees of RL-MPC, which has only been tested empirically. Additionally, the tests are performed in a deterministic setting since the BOPTEST framework does not yet provide the functionality to emulate uncertainties. Although RL-MPC is expected to excel in stochastic settings, this study shows that it can achieve similar performance levels to MPC in the deterministic case.

This paper uses the following *notation*: scalar-valued variables, scalar-valued functions, and scalar-valued sets are denoted by regular italic symbols as in x. Vector-valued variables, vector-valued functions, and vector-valued sets are denoted by bold italic symbols as in $\boldsymbol{x}$. Deterministic variables in the abstract domain are denoted by lowercase letters as in r, whereas stochastic variables in the abstract domain are denoted by capital letters as in R. The latter does not apply to the physical domain where a variable like the temperature or electric power may be denoted by capital letters e.g. by T, or P.

The *outline* of the paper is as follows: Section 2 summarizes related work; Section 3 performs a conceptual comparison between MPC and RL; Section 4 describes the RL-MPC algorithm. Section 5 describes the implementation details of MPC, RL, and RL-MPC for their assessment in simulation. Section 6 presents the simulation results obtained from the implementation of each control algorithm in the same simulation test case building. Finally, Section 8 draws the main conclusions.

2. Related work

Extensive scientific literature exists that explores the implementation of either MPC or RL in buildings, e.g. [18] or [19]. The increased attention to MPC for this application during the last years is remarkable. An elaborate review on the application of MPC in buildings is offered by [20]. This advanced control approach has been proven not only at the building supervisory level as a reference governor for temperature set-point control [21], but also at the local loop level, where MPC directly decides on the actuator signals that drive the heating, ventilation, and air conditioning (HVAC) systems in buildings [22].

It is natural that MPC has dominated the control of individual buildings because buildings are usually large continuous systems with well-known dynamics. Hence, analytical models can be derived from domain knowledge to optimize the system in a receding horizon approach. Multiple libraries and tools exist that facilitate the modeling task, e.g. [23–27]. These models can also help practitioners in making decisions for building energy commissioning. However, the complexity of the optimization problems subject to these models (representing system constraints) remains a major bottleneck for the widespread adoption of MPC in practice. That is why the configuration of control-oriented models is a central research topic in the field [28,29].

On the other hand, the rapid emergence and sophistication of general function approximation techniques like neural networks or random forests have leveraged the development of powerful RL algorithms that can be implemented for various applications [9,30]. The latter has gained the attention of the building sector, and recent studies arise as well where this optimal control technique is implemented in buildings, usually to harness the flexibility of buildings in a demand response setting [4]. A review of previous implementations of RL for demand response can be found in [31]. Despite the substantial progress in RL, this control approach is still generally implemented at a supervisory level, which allows only indirect control of the building HVAC systems. Examples of this indirect control can be found in [32] or in [33]. Even when direct control is implemented, the RL agent usually controls low-dimensional action spaces like on/off operation [34,35].

Many simulation case studies have claimed that model-free RL agents can learn policies for building climate control from only a few

days of operation, but the agents are usually tested using reduced-order models [4,36–38]. This practice directly conflicts the findings of Picard et al. [39], who emphasized the relevance of using a detailed and reliable plant model when evaluating the building controllers in order not to overestimate their performance. Real implementations of RL for building HVAC control like [15,40,41] never use model-free RL techniques directly and rely on simulation-based off-line learning or model-based RL. To the best of the authors' knowledge, only Peirelinck et al. [34] has evaluated a model-free RL agent in a simulation environment using a detailed emulator building model. However, they used indirect control and a low-dimensional action space. By using indirect control the agent does not need to deal with constraint satisfaction. Contrarily, it only suggests actions that can be overwritten by the backup controller of the emulator, which is ultimately responsible for satisfying constraints. Note that satisfying constraints is of utmost importance for every controlled system because it avoids unwanted behavior, performance degradation, and the violation safe-critical conditions.

This paper uses a detailed and high-fidelity building model from the BOPTEST framework to evaluate the controllers. Similar setups to BOPTEST are the EnerGym [42] environment or the CityLearn challenge [43], although these use reduced-order building models and focus on algorithms coordinating demand response strategies in districts rather than control strategies for the energy management of individual buildings. The CityLearn setup was extended in [32] with a Q-learning agent based on TensorFlow [44] to compare a rule-based controller with a single- and a multi-agent RL controller. Despite using function approximations, the state and action spaces were limited to only four states and one action. Moreover, the indoor temperature was not included in the state space, making it impossible to exploit the thermal flexibility of the building's indoor air. We use a high-order model from the BOPTEST framework to test the advanced control algorithms, which contrasts with previous related studies that used first or second order emulator models, e.g. in [4,36–38,42].

Most of the studies mentioned above consider either MPC or RL independently, and only a few of them compare both methods for the same building test cases. One example is described in [36], where a model-based and a model-free RL algorithm was implemented to control building space heating. They also compared this algorithm in simulations against an idealized MPC and a rule-based controller. However, the MPC served as an upper performance bound since it assumed perfect knowledge of the system dynamics. Mbuwir et al. [4] also compared two RL techniques to MPC, where again the MPC was assuming full knowledge of the system parameters. Cost savings of a simulation-based RL agent were shown in [41] during an experimental study. This approach was then compared to other approaches using a calibrated model for the evaluation, and it was indicated that a model-based optimization approach could significantly outperform the RL agent. The agent was shown to be highly affected by the training model accuracy used in the off-line learning phase. Ernst et al. [9] directly compared both control paradigms, showing that RL may be competitive with MPC. However, they implemented such comparison in a simple case without any disturbances.

Even less common are the studies that investigate the possibilities of merging MPC and RL approaches to exploit the merits of each method. Learning-based MPC is a research field that automates control design and tuning from monitoring data to enhance performance. Although the field of learning-based and data-driven MPC approaches is vast, only a few studies investigate the possibilities of merging MPC and RL to exploit the strengths of each method. Hewing et al. [45] present a general review about learning-based MPC and classifies these methods into three main categories: learning the system dynamics, learning the controller design, and MPC for safe learning. An overview of learning-based MPC with a major focus on RL can be found in [11].

For the application of collaborative methods in buildings, we highlight the work of Ojand et al. [14] who present a two-level MPC integrating Q-learning for the control of a residential community using an energy management system aggregator. The MPC computes optimal planning for day-ahead operation at the aggregator level and Q-learning is used for the real-time control of each thermostatically controlled load in the community. Although MPC and RL collaborate in this case study, they work at two different levels, with each agent being independent of the other. It is also worth noting the work of Liu et al. [46]. They laid the foundations of a theoretical framework for implementing RL in an actual building and classified the solution of sequential decision making problems into four main categories: pure planning (MPC), classical RL, simulation-based RL and model-based RL. They present simulation- and model-based RL as hybrid approaches between both optimal control families. While it is true that model- and simulation-based approaches combine learning and planning, they do not combine the machinery of MPC and RL. On the contrary, these approaches learn a model that serves an agent to update its policy from simulated experience.

The recently developed Differentiable MPC [47] provides a methodology to combine planning and control. This algorithm solves the linear quadratic regulator problem formulated as a classical MPC problem in a forward pass. In a backward pass, the parameters of the model are estimated. Differentiable MPC has been adopted by Chen et al. [15] to develop a new algorithm named Gnu-RL that allows end-to-end planning and control of HVAC systems. A quadratic cost function with linear constraints is formulated as in MPC, and the parameters of the system dynamics are estimated from historical data. Remarkably, the objective function parameters are also estimated to obtain the advantage function of their gradient-based RL algorithm, which constitutes their core novelty and bridges the gap between MPC and RL. On a similar note, Drgoňa et al. [48] propose to use differentiable predictive control combined with neural networks for optimal control without requiring the supervision of an expert controller. Their approach is successfully tested in a MIMO building system.

Although Differentiable MPC and Gnu-RL suppose a significant step forward in merging both optimal control families, the implementation of Chen et al. only used a low-dimensional state–action space and a prediction horizon of only 1.5 h. Moreover, they formulate an objective function that balances thermal discomfort and energy use, but do not encode any building system dynamics in their optimization problem. Contrarily, the system dynamics are introduced as state-space matrices randomly initialized with parameters that lack any physical meaning.

More generally, RL has been proposed to collaborate with MPC for applications other than building energy management. In [49–51], and in [52] it is proposed to use the MPC as a function approximation for RL, in the same spirit as Differentiable MPC. Formal theory for this approach is presented in [53]. Kamthe et al. [54] use dynamic programming combined with Pontryagin's maximum principle to solve the constrained open-loop optimization of an MPC. In the latter scheme, Gaussian process models are learned instead of a parametrized policy or value function.

Negenborn et al. [10] propose to use an explicit representation of the system dynamics to optimize in the short term (e.g., for one step ahead) and to lean on a value function to account for the long-term value of an action. The idea is to use value functions to help the MPC in dealing with suboptimality, finite horizons, and computational requirements. In this approach, the MPC provides robustness and decision making over the short term, and RL provides adaptation and decision making over the long term.

The proposal of Negenborn et al. is probably the prior work that resembles the RL-MPC algorithm of this paper the most, although they never implemented their method in practice. On the contrary, Zhang et al. [55] implemented a similar concept in the Q-learning-based model predictive control using the Lyapunov technique (Q-LMPC) for continuous nonlinear systems with complicated dynamics. They analyzed the convergence of such a scheme and showed that the control policy approximated by the algorithm ensures the stability of the

closed-loop system. The algorithm was successfully tested in simulations for a well-mixed, non-isothermal continuous stirred tank reactor with two states and two inputs. The simulation results showed that when using an inaccurate system model to pretrain Q-LMPC can improve the performance of the closed-loop system and that a well-trained Q-LMPC can achieve performances similar to those of LMPC.

This paper elaborates on the approach proposed by Negenborn et al. [10], which is also similar to Q-LMPC. The approach is extended by encoding domain knowledge with a pretrained physics-based controller model and using a state observer to estimate the initial states of the model every control step. Additionally, the algorithm is empirically tested in a standardized framework for the assessment of building controls and compared against classical MPC and RL approaches.

3. Disentangling the differences between MPC and RL

Optimal control solves a sequential decision making problem to determine the actions that optimize a performance objective. The previous section has underlined the need of comparing and the potential of merging the two main approaches for optimal control applied to building energy management: MPC and RL. Both methods share common components, while other components are more controller-specific. These differences in formulation make it difficult to compare and merge both approaches, and motivates an analysis at a conceptual level. Fig. 1 classifies the solution methods for optimal control. The arrows indicate the dependencies of each element on a lower abstraction level, such that a single algorithm can be characterized by tracing a downward path. Ramifications may arise, especially for Markov Decision Processes (MDP), where the solution methods for learning an optimal policy are characterized as the combination of several features, rather than exclusive attributes leading to a single thread. It is important to note that the taxonomy presented in Fig. 1 is not exhaustive because of the wide variety of existing approaches. However, it helps in locating the main methods for optimal control and sets a common ground for a complete classification. The following subsections report the main aspects of these control methods.

3.1. Approach

Starting from the highest level of abstraction at the top of Fig. 1, there exist mainly two ways of approaching an optimal control problem: using the receding horizon principle inherent to MPC, or formalizing the problem as a MDP.

In MPC, at every time step k a vector of measurements $\boldsymbol{m}_k$ is taken from the plant, and an observer estimates the state vector $\hat{\boldsymbol{x}}_k$ that fully characterizes the controller model at current time. Then, the future state $\boldsymbol{x}$ and input $\boldsymbol{u}$ trajectories are optimized for a finite prediction horizon Δt_h according to the explicit representation of an objective function J and a controller model $\boldsymbol{F}$. The set of constraints $\boldsymbol{H}$ are also explicitly introduced in the optimization problem. The objective function, model and constraints may also depend on the model outputs $\boldsymbol{y}$, algebraic variables $\boldsymbol{z}$, disturbances $\boldsymbol{d}$, and time-invariant parameters $\boldsymbol{p}$. The forecast of the disturbances along the prediction horizon $\boldsymbol{d}(t_k, t_k+\Delta t_h)$ are provided as external data to the optimization. The objective function is usually defined as the integral of a function l of the model variables along the prediction horizon, as indicated by Eq. (1). Only the first control input from the optimized trajectory is implemented. The complete MPC process is shown in Fig. 2(a).

$$J_k = \int_{t=t_k}^{t_k+\Delta t_h} l(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{d}(t), \boldsymbol{p}) dt \tag{1}$$

In the application of MPC to building energy management, the state vector $\boldsymbol{x}$ typically represents building temperatures like the zone operative, floor, or wall temperatures, and the model outputs $\boldsymbol{y}$ are usually a subset of these, e.g. the zone operative temperature only. The vector of algebraic variables $\boldsymbol{z}$ represents the variables dependent on the states, e.g. the heat flows through the building envelope, and the vector of disturbances $\boldsymbol{d}$ comprises the main uncontrollable variables affecting the building thermal behavior. Typically, the ambient temperature, solar irradiation, and the occupancy are included in $\boldsymbol{d}$. The set of time invariant parameters $\boldsymbol{p}$ may or may not represent physical properties of the building depending on whether the controller model uses or does not use physical insights, respectively.

In an MDP, the decision making model is defined by the state-space $\mathcal{S}$, the action space $\mathcal{A}$, the rewards $\mathcal{R} \subset \mathbb{R}$, and the transition function of the environment $\boldsymbol{f}$. Notice that $\boldsymbol{f}$ represents the ground truth of the system dynamics, contrarily to the controller model $\boldsymbol{F}$ used in MPC, which is a simplified representation of the system. In RL, the agent interacts with the environment during a sequence of discrete-time steps. Every step k, the RL agent receives an observation of the state-space $\boldsymbol{S}_k \in \mathcal{S}$ and a reward $R_k \in \mathcal{R}$ that reflects the goodness of the action taken. In turn, the agent computes its control logic and sends a new action $\boldsymbol{A}_k \in \mathcal{A}$ to the environment. The environment is defined by $\mathcal{E} : \mathcal{S}\times\mathcal{A} \rightarrow \mathcal{S}\times\mathcal{R}$, and the objective of RL is to infer an optimal control policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected cumulative return G when the agent acts according to it. The cumulative return is defined as some function of the rewards sequence, and a typical definition is to discount the rewards with a factor $\gamma \in [0, 1]$ as shown in Eq. (2). The process is summarized in Fig. 2(b).

$$G_k = R_{k+1} + \gamma R_{k+2} + \gamma^2 R_{k+3} + \cdots = \sum_{i=0}^{\infty} \gamma^i R_{k+i+1} \tag{2}$$

3.2. Terminology

By analyzing the control processes from Figs. 2(a) and 2(b) it is possible to identify several expressions with a total or partial equivalence between both approaches. Controller and agent, plant and environment, control input $\boldsymbol{u}$ and action $\boldsymbol{A}$, can all be considered as equivalent elements. Notice, however, that slight differences arise from these elements. For instance, the plant is limited to the representation of the system process, while the environment is extended to provide the states and rewards as perceived by the agent. Others, like the objective and return, or the state's definition, can be seen as analogous elements only. The control community usually minimizes an objective function J whereas the machine learning community maximizes a cumulative return G. The relation between both can be formalized through the immediate reward. In the deterministic setting, the scalar immediate reward r_{k+1} can be related to the objective function as follows:

$$r_{k+1} = -(J_{k+1} - J_k) \tag{3}$$

Special attention should be paid to the definition of the state. The control community tends to use the state $\boldsymbol{x}$ to designate only internal properties of the system. On the contrary, the machine learning community tends to use the state $\boldsymbol{S}$ to refer to the environment's condition, which may include internal and external system variables like the forecast of the disturbances, the previous measurements, or some notion of time. Therefore, the state in machine learning may acquire a different meaning than in control theory, especially for controllers using a physics-based model where the state is a strict representation of a system variable.

This terminology is conventionally used by each community separately. However, it is common to find studies that maintain their native notation while borrowing concepts from the other community, or the other way around, e.g. in [12] or in [56]. While this is a valid practice as far as consistency is maintained, it makes the cooperation of both approaches difficult because it obscures the precise meaning of each term. For the sake of clarity, this work strictly respects the notation typically adopted by each community as presented above. Therefore, a state $\boldsymbol{x}$ has a different meaning than $\boldsymbol{S}$, and the controller model $\boldsymbol{F}$ is also different from the ground truth dynamics represented with $\boldsymbol{f}$.

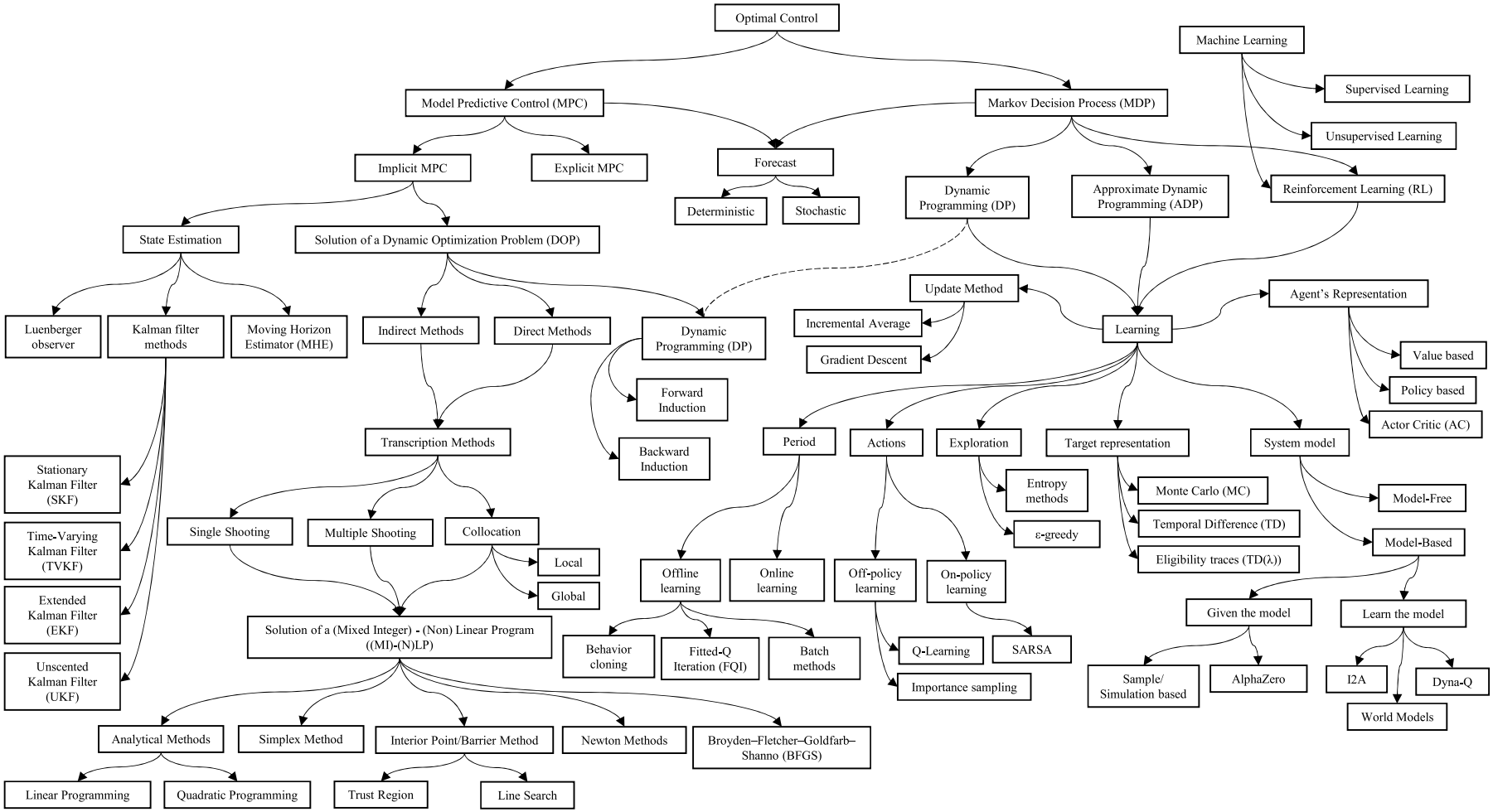
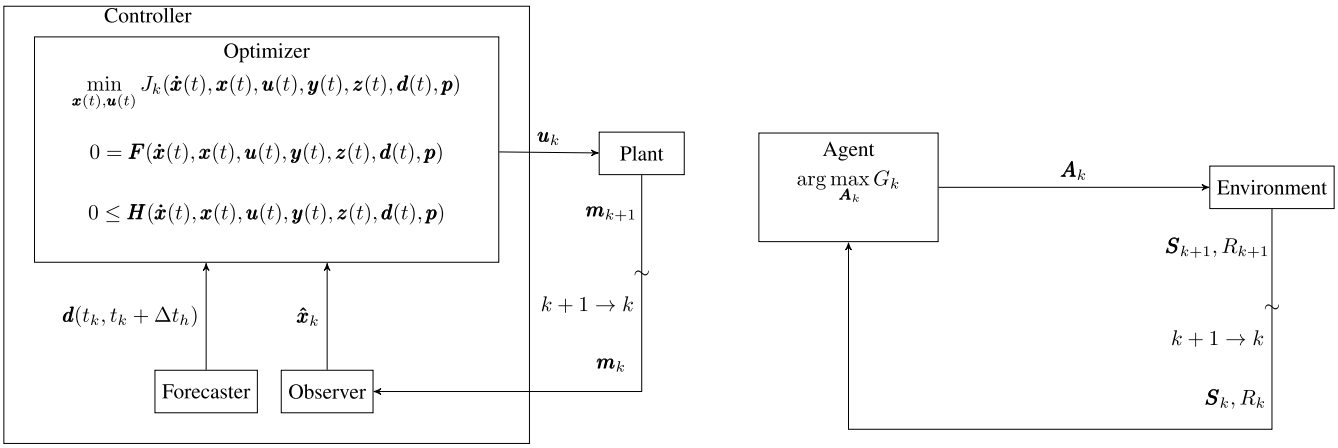


Fig. 1. The (non exhaustive) taxonomy of optimal control. An algorithm shall be characterized by tracing a path from the highest abstraction level to the lowest. Ramifications may arise for the same algorithm.



(a) MPC (b) MDP

Fig. 2. Block diagrams of MPC (left) and MDP (right). Note: $(\cdot)_k \triangleq (\cdot)(t_k)$.

3.3. Solution method

MPC can either be solved implicitly by performing state estimation, forecasting and resolving a dynamic optimization problem at every time step, or it can be solved explicitly by learning a control policy from data produced from an implicit MPC with any kind of function approximation. Therefore, implicit MPC has larger online computational cost since it requires to estimate the states and perform dynamic optimization every control step. Contrarily, explicit MPC can deliver the optimal input just at the cost of a function evaluation, but it has a larger offline computational cost. The reason is that explicit MPC requires the configuration and implementation of an implicit MPC to learn the policy function using any supervised learning method. The main advantage of an explicit MPC formulation stems from its easy implementation even on low-level hardware [57]. The outcome of an explicit MPC is similar to the outcome of any algorithm solving an MDP since the function approximation of explicit MPC can be interpreted as a policy that decides actions from system observations.

The solution of the dynamic optimization problem is the core of the implicit MPC approach. Three main solution methods can be identified to resolve dynamic optimization problems: dynamic programming (DP), direct methods, and indirect methods [58]. Direct and indirect methods differ in when discretization takes place, but both require discretization. The process of transforming the infinite-dimensional optimization problem to the finite-dimensional optimization problem through discretization is called transcription. Single-shooting, multiple-shooting, and collocation are the most widely used transcription methods. These methods differ on how the system variables are discretized and how the system dynamic are imposed. After transcription, the solution boils down to solving a (Mixed Integer) — (Non) Linear Program ((MI)-(N)LP) with methods that should be selected based on the properties of the resulting problem. Examples are quadratic programming, line search, or the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [59]. The solution method used for an MDP depends on the information accessible to the agent and the approximations needed to derive a control policy. When perfect knowledge of the MDP is available, DP algorithms can obtain an exact solution. If the MDP contains infinite elements, as in continue state–action spaces, approximations are required, and Approximate Dynamic Programming (ADP) is used. When the agent cannot reproduce the actual environment and needs to learn from samples of experience, RL methods are needed instead. In the latter case, the agent learns from empirical experience, either from historical data or direct interaction with the environment. A trade-off should be found between off-line and on-line learning. Fig. 1 shows the main aspects that may characterize the learning process of the solution method of an MDP, i.e.: the period when learning takes place, the approach used to decide actions, the exploration method, the use of a system model, the way of representing the targets for training, the method used for policy updates, and the agent's architecture.

3.4. Optimality

In MPC, the quality of the optimization solution is subject to the accuracy of the controller model, which is often simplified for computational reasons. Moreover, direct optimization methods are based on Karush–Kuhn–Tucker conditions, and indirect methods are based on Pontryagin's maximum principle. Both can only provide necessary conditions for optimality unless specific convexity properties are met for the objective function and the constraints. Stability and feasibility are inherently guaranteed for MPC, while there is only immature theory for these matters in RL [12]. The lack of safety guarantees in RL stems from the constraints not being directly imposed in the formulation of the solution method.

On the other hand, DP-based methods rely on Bellman's principle and can provide sufficient conditions for global optimality. As a downside, these methods are hampered by the curse of dimensionality, i.e., an exponential complexity growth with the size of the state–action space. Pure DP can be used only when the state space is countable and the MDP is perfectly known. Algorithms approximating any element of the MDP, like ADP or RL, can only guarantee optimality under certain conditions, namely when using linear function approximations or Monte Carlo learning. Despite these challenges, RL can target real complex problems where exact methods may become infeasible [13], though optimality is not guaranteed in these cases. Additionally, the same RL algorithms may extrapolate to very different types of environments.

3.5. Computational effort

A main drawback of implicit MPC is the burden of solving an optimization problem on-line that can be complex and involve a large number of optimization variables. This is why controller models for MPC are commonly simplified at the cost of optimality loss and why efficiency gains in optimization solvers are highly wanted. Additionally, state estimation and forecasting need to be performed at every control step. Explicit MPC aims to overcome this burden by developing a control policy through behavior cloning, but an implicit MPC needs to be developed beforehand. The off-line complexity of MPC relates to the engineering effort required to develop and identify a controller model that is accurate and suitable for optimization. While the latter is a challenging task, its computational cost is limited.

DP-based algorithms, on the other hand, typically work mostly off-line, producing a policy that is then used to control the process [56]. Large computational efforts are expected to train policies even for fairly simple systems. In RL, a model of the environment may be used for off-line simulation-based learning. This approach provides a separate training environment that does not limit exploration to obtain a high variance data-set. Other off-line methods to learn a policy are behavior cloning or Fitted-Q Iteration (FQI) [60]. Once the policy is developed, it directly provides the estimated optimal action given the state and at the cost of a policy evaluation only, which may be expressed by a neural network or other mathematical functions. While learning may continue on-line, the agent shall be more limited when exploring new actions to improve its policy. This is because deviations from the learned policy may result in performance degradation or unexpected behavior [13].

The *sampling efficiency* is defined as the capacity of an RL agent to learn a reliable policy from the least possible amount of interactions with the environment, and it is an actively studied topic within the machine learning community, especially in the context of model-based RL approaches [13,61].

3.6. Prediction horizon

Both MPC and RL are predictive controllers, independently of whether they integrate disturbances forecast in their control logic. MPC uses explicit optimization along a finite prediction horizon, and RL learns actions to optimize the sum of the immediate *and* the discounted future rewards. A shortcoming of MPC is the finite horizon, which is especially pronounced in tasks with sparse rewards where a short horizon can make the agent extremely myopic [62]. Additionally, MPC struggles with longer prediction horizons because these increase the number of state and input variables in the optimization. Only single shooting methods are independent of the number of states, but they are known for their instability, especially for stiff systems. On the contrary, the machine learning community usually works with infinite discounted returns, although finite returns are sometimes considered as well. The discount factor represents the extent of credit given to future rewards and bounds the infinite sum. Notice that policies built with finite returns require time as an argument to use the policy function associated with the passed time.

An *effective horizon* can be defined as the number of steps after which rewards are negligible for policies with infinite prediction horizons. It is determined by the discount factor using the convergence of a geometric series, as shown in Eq. (4).

$$1 + \gamma + \gamma^2 + \cdots = \frac{1}{1-\gamma} \tag{4}$$

The above equation indicates that the rewards for time steps after $1/(1-\gamma)$ are ignored in the total return. Hence, it is possible to relate the discount factor of an RL agent with the finite prediction time horizon of an MPC as follows.

$$\Delta t_h = \Delta t_s \frac{1}{1-\gamma} \tag{5}$$

Where Δt_h is the finite prediction horizon of the MPC and Δt_s is the control step period.

Although infinite prediction horizons may benefit from improved behavior in the long term, it should be noted that RL algorithms make less efficient use of the forecast information. The reason is that all forecast data is flattened and brought to the agent's observations with the time dependency removed from it. The agent needs to figure out the time dependency that is not explicitly described in the problem definition during the learning process. On the contrary, MPC naturally preserves the time dependency (and thus the chronology) by taking into account the system dynamics computed in the controller model. A potential synergetic approach may use short prediction horizons to solve an optimization problem for MPC while benefitting from the infinite horizons that characterize RL.

3.7. Use of models

Models and function approximations are used differently in MPC and RL. In MPC, the model used to represent the system is called the *controller model*. These models are obtained through domain knowledge, system identification or supervised learning from historical monitoring data. Commonly, the controller models are qualified as white-, gray-, or black-box depending on whether physical insights and/or monitoring data are used for their configuration. The optimization problem in MPC imposes severe restrictions on the controller models, which are often simplified to guarantee convergence at the cost of performance loss. The same controller model (or a derivation) is commonly used for state estimation as well.

In RL, models approximate the policy or the value function, which does not directly represent the system dynamics. A particularity of training policies and value functions is that the training data targets are continuously obtained from experience samples, in contrast to those used for supervised learning, which are fixed before learning begins [63]. This characteristic poses a significant challenge to many machine learning modeling approaches that are based on independent and identically distributed data. RL may also use a system model to pretrain a policy in simulation-based RL [41], or to alternate real and modeled experience in model-based RL [64,65]. An important advantage of using system models to train RL agents is that their analytical form is not required, such that they are not restricted in complexity.

3.8. Partial observability

Since MPC uses a system model to optimize actions for every control step, it needs to know the present value of the state vector, which may not be entirely determined from the system measurements. Hence, state estimation is required to estimate the hidden states of the controller model, i.e., those states that are not measured during operation. A few observers used for state estimation are listed in the scheme of Fig. 1. A Luenberger observer is probably the most straightforward and intuitive type. Instead, moving horizon estimation is a complex and computationally expensive approach since it requires optimization for each state estimation, but allows nonlinear dynamics. Extended Kalman filter methods are the non-linear version of the conventional stationary Kalman filter and the time-varying Kalman filter. Finally, the unscented Kalman filter uses a minimal set of carefully chosen sample points to represent the state distribution. It uses the unscented transform to propagate through the true non-linear system empirically. Comparative studies of different state estimators for their application to building systems can be found in [66,67].

Similarly, an MDP is partially observable when some of the elements of the state-space remain hidden to the agent's observation. The Markov property is violated in these scenarios since the future behavior cannot be entirely determined from the present state. To amend this, an arbitrary selection of past features is introduced in the observation space, e.g. in [34], to allow the agent to extract the hidden features.

4. Reinforced model predictive control (RL-MPC)

This section introduces the details of the novel proposed RL-MPC algorithm. The goal is to learn from the environment while ensuring constraint satisfaction. With that aim, elements from the control and machine learning communities are effectively combined, namely state estimation, dynamic optimization, and learning. First, Section 4.1 provides a high-level introduction on how MPC and RL are logically merged. Then, Section 4.2 provides a formal description of the RL-MPC algorithm.

4.1. Intuitive description

To understand the intuition behind RL-MPC it is first required to realize the differences between the non-linear program used to solve the online optimization problems in MPC and the two main learning methods of RL: value-based and policy-based methods. Fig. 3(a) shows a schematic representation of the non-linear program obtained from an MPC. The solution method of MPC takes as an input the union of the estimated states $\hat{x}$ and the forecast of the disturbances d. In this approach, the objective function J is inferred every control step based on the controller model. Fig. 3 also depicts the value-based approach (Fig. 3(b)) and the policy-based approach (Fig. 3(c)) for uni-dimensional state and action spaces in an equivalent optimal control problem. These RL approaches take as an input the agent's observation $\boldsymbol{s}$. Value-based methods characterize an action-value function $q(\boldsymbol{s},\boldsymbol{a})$ as the expected total cumulative return from state $\boldsymbol{s}$ when taking action $\boldsymbol{a}$. Policy-based methods directly map a state $\boldsymbol{s}$ to an action $\boldsymbol{a}$ with a policy function π parametrized with the parameter set $\boldsymbol{\theta}$. In RL-MPC, the MPC non-linear program is truncated with the expected value of the state one step ahead $\boldsymbol{s}'$ as estimated by the value-based RL approach. Hence, RL-MPC uses value-based RL to estimate the value of being in a specific state $\boldsymbol{s}'$ as obtained by the MPC when using a prediction horizon of only one control step. Therefore, the main components of the MPC remain active in RL-MPC, namely the state estimator, forecaster and optimizer, but the value function is used to shorten the non-linear program and to enable learning. Fig. 4 shows a high-level diagram with the interaction of the main components of RL-MPC. Fig. 4 intuitively illustrates how MPC and RL are merged in the RL-MPC algorithm.

4.2. Formal description

In the deterministic setting of an MDP, the *action-value* function $q_\pi(\boldsymbol{s},\boldsymbol{a})$ is defined as the expected return when taking action $\boldsymbol{a}$ from state $\boldsymbol{s}$ and following policy π hereafter. The objective is to find a policy π_* that maximizes the sum of the expected cumulative returns. Implementing Bellman's principle of optimality, the following expression is obtained:

$$q_*(\boldsymbol{s},\boldsymbol{a}) = \max_\pi q_\pi(\boldsymbol{s},\boldsymbol{a}) = r_{k+1} + \gamma \max_{\boldsymbol{a}'} q_*(\boldsymbol{s}',\boldsymbol{a}') \tag{6}$$

Where q_* represents the optimal q function and $\boldsymbol{s}'$ is the reached state when applying action $\boldsymbol{a}$ from state $\boldsymbol{s}$. Using the equivalence from Eq. (3), the previous expression can be rewritten as Eq. (7).

$$q_*(\boldsymbol{s},\boldsymbol{a}) = -(J_{k+1} - J_k) + \gamma \max_{\boldsymbol{a}'} q_*(\boldsymbol{s}',\boldsymbol{a}') \tag{7}$$

This can be simplified by defining the *state-value* function $v_\pi(\boldsymbol{s})$, which is the expected return from state $\boldsymbol{s}$ and following policy π hereafter. The state-value function can be related to the action-value function as shown in Eq. (8).

$$v_*(\boldsymbol{s}) = \max_{\boldsymbol{a}\in\mathcal{A}} q_{\pi_*}(\boldsymbol{s},\boldsymbol{a}) \tag{8}$$

This allows reformulating the action-value function as follows.

$$q_*(\boldsymbol{s},\boldsymbol{a}) = -(J_{k+1} - J_k) + \gamma v_*(\boldsymbol{s}') \tag{9}$$

Notice that Eq. (7) exposes the objective function from a classical MPC formulation along the first prediction step. Hence, given a state $\boldsymbol{s}$, the optimal action $\boldsymbol{a}$ can be obtained by optimizing $q_*(\boldsymbol{s},\boldsymbol{a})$ while explicitly imposing system constraints, as typically done in the MPC approach. Assuming that a controller model $\boldsymbol{F}$ is available, the policy followed by RL-MPC is defined in the set of Eqs. (10).

$$\pi_*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} q_*(\boldsymbol{s},\boldsymbol{a}) = \tag{10a}$$

$$\arg\max_{\boldsymbol{a}} \left(-\int_{t=t_k}^{t_{k+1}} l(\dot{\boldsymbol{x}}(t),\boldsymbol{x}(t),\boldsymbol{a}(t),\boldsymbol{y}(t),\boldsymbol{z}(t),\boldsymbol{d}(t),\boldsymbol{p})dt + \gamma v_*(\boldsymbol{s}') \right) \tag{10b}$$

$$s.t. \begin{cases} 0 = \boldsymbol{F}(\dot{\boldsymbol{x}}(t),\boldsymbol{x}(t),\boldsymbol{a}(t),\boldsymbol{y}(t),\boldsymbol{z}(t),\boldsymbol{d}(t),\boldsymbol{p}) & \text{(c)} \\ 0 \le \boldsymbol{H}(\dot{\boldsymbol{x}}(t),\boldsymbol{x}(t),\boldsymbol{a}(t),\boldsymbol{y}(t),\boldsymbol{z}(t),\boldsymbol{d}(t),\boldsymbol{p}) & \text{(d)} \end{cases} \tag{10}$$

The main advantage of utilizing the formulation presented by Eqs. (10) is that it imposes safety constraints in the short term while enabling continuous learning from empirical experience. Moreover, shortening the prediction horizon of the dynamic optimization problem considerably eases the complexity of the resulting non-linear program. The long-term behavior is still accounted for with the state-value function v. It is important to note that both terms from Eq. (10b) should be jointly optimized, such that state $\boldsymbol{s}'$ needs to be related to the expected optimization variables in t_{k+1}. This leads to a lower overhead than optimizing with longer prediction horizons that need discretization over time. Additionally, the reward needs to be shaped in alignment with the design of the objective function and the constraints to encourage cooperation between both terms in the action-value function. This means that the first and second elements of the objective function as shown in Eq. (10b) should not compete. Contrarily, they should be defined to be complementary.

It is also worth noting that domain knowledge is encoded in the controller model $\boldsymbol{F}$ for optimization and state estimation, providing intelligibility to the algorithm. The controller model $\boldsymbol{F}$ can also be used to configure a separate simulation environment $\mathcal{E}_{\boldsymbol{F}}$ to pretrain $q(\boldsymbol{s},\boldsymbol{a})$ from simulated experience and expedite learning. The process is summarized in Algorithm 1, where α is known as the *learning rate*. Note that lines 1–3 of Algorithm 1 constitute the off-line learning part, while lines 4–10 constitute the deployment of the algorithm in the actual building environment.

Algorithm 1 Reinforced model predictive control (RL-MPC)

1: Identify $\boldsymbol{F}(\dot{\boldsymbol{x}}(t),\boldsymbol{x}(t),\boldsymbol{a}(t),\boldsymbol{y}(t),\boldsymbol{z}(t),\boldsymbol{d}(t),\boldsymbol{p}) = 0$
2: Configure $\mathcal{E}_{\boldsymbol{F}}$
3: Pretrain $q(\boldsymbol{s},\boldsymbol{a})$ using $\mathcal{E}_{\boldsymbol{F}}$
4: **while** true **do**
5: $\quad\boldsymbol{s} \leftarrow$ current state
6: $\quad\hat{\boldsymbol{x}} \leftarrow$ estimate from measurements
7: $\quad\boldsymbol{a} \leftarrow$ Solve Eqs. (10)
8: $\quad\boldsymbol{s}', r_{k+1} \leftarrow$ Apply $\boldsymbol{a}$ to $\mathcal{E}_{\boldsymbol{f}}$
9: $\quad q(\boldsymbol{s},\boldsymbol{a}) \leftarrow q(\boldsymbol{s},\boldsymbol{a}) + \alpha[r_{k+1} + \gamma \max_{\boldsymbol{a}} q(\boldsymbol{s}',\boldsymbol{a}) - q(\boldsymbol{s},\boldsymbol{a})]$
10: **end while**

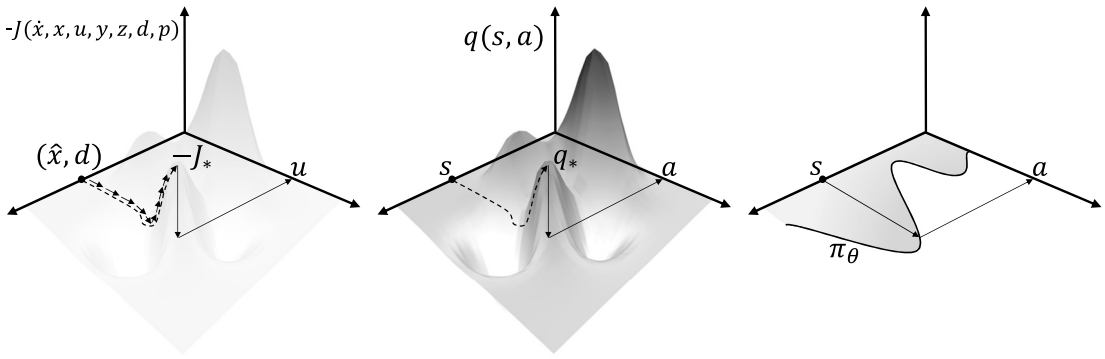
5. Implementation

This section describes the implementation details of MPC, RL, and RL-MPC to the same test building included in the BOPTEST framework. Each controller uses equivalent features to enable a fair comparison. An explanatory diagram is shown in Fig. 5 that summarizes the implementation methodology.

5.1. The control problem in BOPTEST

A standardized simulation environment is required to ensure a fair evaluation of the algorithms. BOPTEST is an open-source framework that provides a menu of high-fidelity emulator building models and the standardization of test cases to assess control algorithms. In this framework, the building models are containerized with the Docker technology and considered as the ground truth plant to be controlled. The functionality is enabled through a clear API to select a test scenario, advance a simulation, and get data like measurements, forecast, or Key Performance Indicators (KPIs) at every control step. The framework is freely accessible in https://github.com/ibpsa/project1-boptest.

The building of interest in this study is the single-zone residential hydronic case as in version `v0.1.0` of the framework. The building



(a) Non-linear program (MPC) (b) Value-based approach (RL) (c) Policy-based approach (RL)

Fig. 3. Schematic overview of solution methods for optimal control. In the MPC approach, the objective function is inferred every control step based on the controller model. In the value-based approach, a value function is constructed and cached to derive the most performant action from a specific state. Finally, the policy-based approach parametrizes a policy to map states to actions directly.

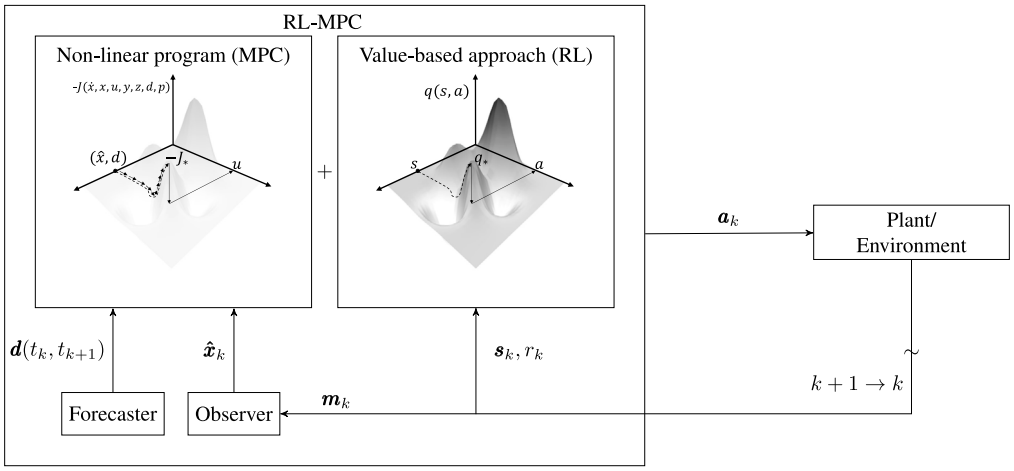


Fig. 4. Block diagram showing a high-level introduction of RL-MPC. Every control step, RL-MPC decides an action $\boldsymbol{a}_k$ by optimizing the sum of the one step ahead non-linear program of the MPC and the value function from the estimated following state. Note: $(\cdot)_k \triangleq (\cdot)(t_k)$.

represents a residential dwelling of 192 m^2 for a family of five members located in Belgium. The family inhabits the building before 7:00 h and after 20:00 h every weekday and full-time during weekends. The HVAC system consists of an air-to-water modulating heat pump of 15 kW thermal power coupled to a floor heating system. The control signal is the heat pump modulation signal for compressor frequency $u_{hp} \in [0, 1]$ that controls the heat delivered to the floor heating. In turn, this heat is released to the building zone by a water emission circuit with a hydronic pump that is activated when the heat pump is working. The controlled signal is the zone operative temperature T_z that should remain within the temperature range 21–24 °C during occupied hours and within 15–30 °C otherwise. No cooling is considered in this test case.

All BOPTEST cases come with one year of boundary condition data and predefined testing periods of two weeks. The cases can be initialized at any arbitrary time of the year, although special attention should be paid to not use any of the predefined testing periods for training. The envisaged building offers a total of six testing scenarios from the combination of two heating periods: peak and typical, and three electricity price profiles: constant, dynamic, and highly dynamic. This work considers the two heating periods with the highly dynamic electricity pricing for testing. This highly dynamic price profile is obtained from the sum of (1) historical day-ahead energy prices as determined by the Belpex wholesale electricity market in 2019, and (2) constant transmission fees and taxes representative for the same location.

The *final objective* of the controller is to guarantee thermal comfort while minimizing operational cost. From the control perspective, the presented problem is particularly challenging for different reasons. First, the controlled variable, T_z, has a significant delay with respect to the control variable, u_{hp}, because of the considerable thermal inertia of the system. Second, the data used for system identification is limited and has low variance because it is generated with the baseline controller that comes along with the test case and that uses a regular thermostatic control. Long training periods or extra-excitations are not considered when identifying the controller model $\boldsymbol{F}$ to emulate a realistic scenario where the controller model works with typical operational data. Third, the controller can only access a few of the signals that influence the plant dynamics, as is the case in reality. Precisely, only the operative zone temperature, out of the 63 continuous-time states present in the emulator model, is measured. Moreover, only 6 out of the 31 disturbances are provided through the forecast. These are weather variables like ambient temperature and solar irradiation or others like energy pricing or comfort setpoint bounds. All control algorithms in this work assume perfect deterministic forecast provided by BOPTEST.

5.2. System identification

A controller model $\boldsymbol{F}$ is required by the MPC, and to configure the environment $\mathcal{E}_{\boldsymbol{F}}$ used to pretrain the policies in the RL and RL-MPC algorithms. The same gray-box model is utilized in all cases, consisting of a thermal resistance–capacitance (RC) architecture constructed from

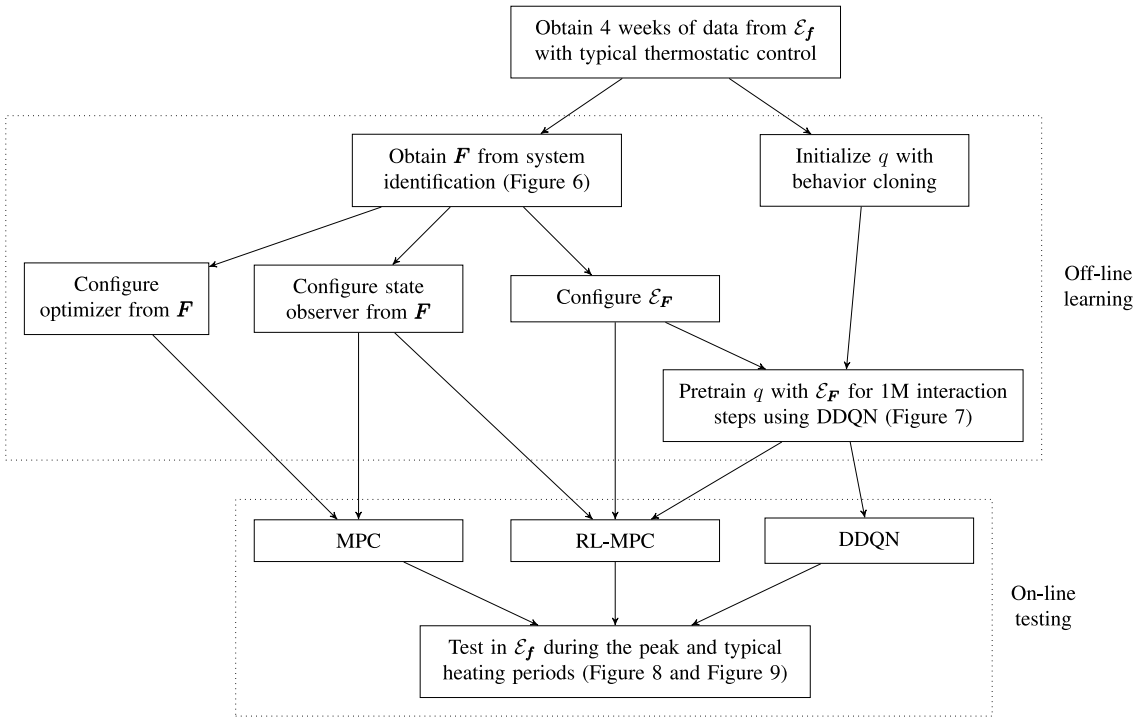


Fig. 5. Implementation methodology.

basic physical principles and without using any system metadata like building geometry or material properties. A model of order five is decided from a forward-selection procedure.

The model has a total of 20 lumped parameters that need to be estimated. Two weeks of data are generated by the emulator with the baseline controller for training, and two weeks are used for validation of the model. The fitting variables are the zone operative temperature T_z, the heat pump condenser thermal power $\dot{Q}_c$ and the heat pump electrical power use P_{hp}. The last two variables determine the coefficient of performance of the heat pump. The model further consists of three inputs: ambient temperature T_a, solar irradiation $\dot{Q}_{rad}$, and occupancy gains $\dot{Q}_{occ}$. Fig. 6 shows an overview of the training and validation periods that have been selected not to overlap any of the testing periods.

5.3. MPC implementation

The MPC of this study heavily relies on JModelica [68], a framework for dynamic simulation and optimization of Modelica models. Particularly, the non-linear JModelica MPC module developed in [69] is used here and has been extended to enable mutable external data. The JModelica MPC module utilizes the direct collocation scheme and relies on CasADi [70] for algorithmic differentiation. This approach is well known for its versatility and robustness.

The optimization problem that needs to be solved at every control step is formulated in Eqs. (11).

$$\min_{u_{HP}} J_k = \min_{u_{HP}} \int_{t=t_k}^{t_k+\Delta t_h} (p^e(P_{hp} + P_{fan} + P_{pum}) + w\delta^{T_z})dt \tag{11a}$$

$$\dot{T}_z, P_{hp}, P_{fan}, P_{pum} = \boldsymbol{F}(u_{hp}, \dot{Q}_{rad}, \dot{Q}_{occ}, T_a, T_z, \boldsymbol{T}, \boldsymbol{p}) \tag{11b}$$

$$\underline{T}_z - \delta^{T_z} \leq T_z \leq \overline{T}_z + \delta^{T_z} \tag{11c}$$

$$\delta^{T_z} \geq 0 \tag{11d}$$

$$0 \leq u_{hp} \leq 1. \tag{11e}$$

In Eqs. (11), the time dependency has been omitted for clarity because all variables are time-dependent except the vector of model parameters $\boldsymbol{p}$, and the weighting factor w, the latter being used to account for the different orders of magnitude between energy cost and discomfort δ^{T_z}. The energy cost is the first term of the objective function where all elements accounting for electrical power are summed and multiplied by the electricity price p^e. These are the heat pump power P_{hp}, the evaporator fan power P_{fan}, and the circulation pump power P_{pum}. $\boldsymbol{T}$ represents the vector of non-measured temperature states of the controller model. The discomfort δ^{T_z} is defined as the deviations of T_z out of the comfort range bounded by $\underline{T}_z$ and $\overline{T}_z$. The weighting factor w is tuned to penalize more heavily the instantaneous discomfort than the instantaneous energy cost, such that the thermal discomfort is treated as a soft constraint in the optimization.

The MPC module is combined with the unscented Kalman filter of the JModelica toolbox for state estimation. The specific state estimation algorithm is the non-augmented version described in [71], and the sigma points are chosen according to [72]. The controller model $\boldsymbol{F}$ described in the previous section is directly used for optimization and state estimation with a control step of $\Delta t_s = 15$ min and a prediction horizon of $\Delta t_h = 24$ h.

5.4. RL implementation

The BOPTEST-Gym wrapper developed in [17] accommodates the BOPTEST API to the Gym standard [73] to assemble the testing environment $\mathcal{E}_f$, which we call the *actual environment* henceforth. Similarly, the $\mathcal{E}_{\boldsymbol{F}}$ is developed by replacing the ground truth building model $\boldsymbol{f}$ by the simpler lower-order controller model $\boldsymbol{F}$ explained in the previous sections. We call $\mathcal{E}_{\boldsymbol{F}}$ the *simulation environment*. Both environments are wrapped following the Gym standard, which provides a convenient setting for implementing any RL algorithm. Additionally, both environments expose the same observations and actions to the RL agent, but their underlying dynamics are different: the actual environment uses a detailed emulator model representing reality, while the simulation environment uses a simple RC model. The motivation for the development of $\mathcal{E}_{\boldsymbol{F}}$ is that it represents an environment that could be obtained in practice to train an agent with simulation-based RL and without any restriction on the actions taken. Contrarily, $\mathcal{E}_f$ represents

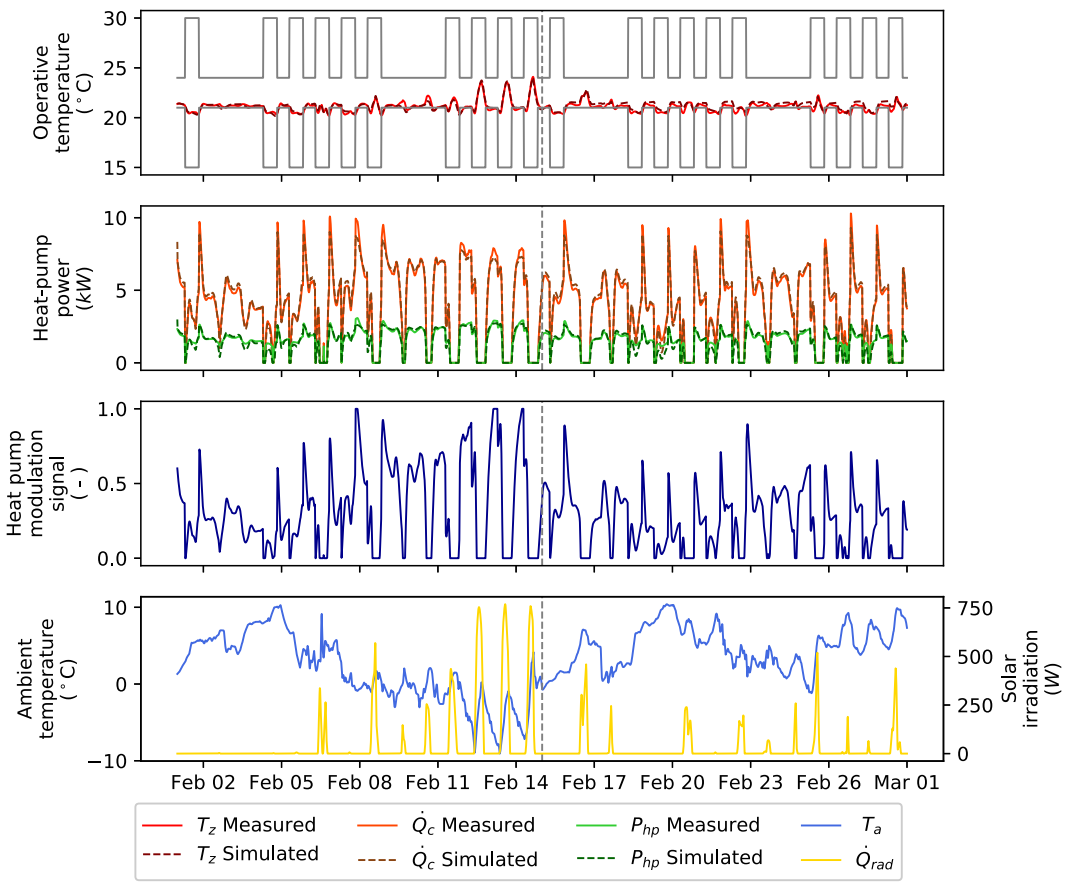


Fig. 6. Training and validation periods for estimating the controller model $\boldsymbol{F}$. The training period is at the left of the vertical gray dashed line and the validation period at the right. The inputs are presented in the bottom two plots, and the simulated outputs are presented in the top two plots and compared with the measured data. The gray lines in the first plot represent the comfort constraints.

an actual building and thus, only a limited set of interactions may be permitted. In both environments, the state–action space is designed to be analogous to the MPC formulation described in the previous section. Specifically, the control step is also set to $\Delta t_s = 15$ min, and the agent's observations include forecast over a prediction horizon of $\Delta t_h = 24$ h with the same interval period. In this configuration, a discount factor of $\gamma = 0.99$ accounts for an effective horizon of approximately one day according to Eq. (5). The reward function is defined as the negated increment of the MPC objective integrand cost as shown in Eq. (3). The state observer is also replicated by including past measurements of the zone operative temperature and the time of the week in the agent's observation. The agent shall use these observations to infer the current building state and to deal with the partial observability of the environment. As a result, the state space observed by the agent has a total dimension of $|\boldsymbol{S}| = 608$ with all continuous variables that are normalized between $[-1, 1]$ to facilitate learning. The heat pump modulation signal is discretized in 10 uniform intervals such that the action space is $\boldsymbol{\mathcal{A}} = \{u_{hp}\}$, with $u_{HP} \in \{0, 0.1, 0.2, \dots, 1\}$, and $|\boldsymbol{\mathcal{A}}| = 11$.

The large state–action space requires the use of function approximations and a suitable RL algorithm for training. We use the Double Deep Q-Network (DDQN) algorithm as proposed in [63] and in [74], which enables a neural network as a function approximation and offers a natural extension towards the implementation of RL-MPC, as explained in the following section. Specifically, a double network is implemented to avoid the overoptimism inherent to Q-learning for large-scale problems [74]. DDQN is an off-policy algorithm that updates network weights following a stochastic gradient descent scheme. Tuples of the form $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ are stored in a *replay memory* $\boldsymbol{D}$ and served in random batches during training in order to alleviate the problem of correlated data. We set the batches to sample one week of transition data. A multi-layer perceptron (MLP) neural network is configured with TensorFlow for the state–action value function. The network contains two hidden layers of 64 neurons each and a rectified linear activation function for the nodes.

In a first learning step, the agent is pretrained with behavior cloning during the month of data used for system identification. Then, the agent interacts with the simulation environment $\mathcal{E}_{\boldsymbol{F}}$ for one million steps. Notice that this is still off-line learning since $\mathcal{E}_{\boldsymbol{F}}$ is a simulation environment. During this learning process, $\mathcal{E}_{\boldsymbol{F}}$ is configured to launch episodes of experience that last for one week and that are randomly initialized throughout the year. The episodes never overlap the periods that will be used for testing later on with $\mathcal{E}_f$, such that the boundary condition data used for testing is not available during training. The agent is configured to follow an ϵ-greedy exploration scheme with a linear schedule that goes from 10% to 1% of random exploration probability along the learning process. The reason to limit the initial exploration to 10% is that the prior behavior cloning forces the agent to explore the most interesting region of the state–action space already from the beginning.

5.5. RL-MPC implementation

The RL-MPC algorithm inherits all attributes and hyperparameters described in the MPC and the RL implementations. This means that the same control step, prediction horizon, and state estimator than the MPC is used, as well as the same pretrained q function than the RL agent. Since both RL and MPC have been designed with analogous hyperparameters, they are expected to be complementary in the RL-MPC implementation. The method jointly optimizes the sum of the immediate rewards (negated objective function increase) obtained from

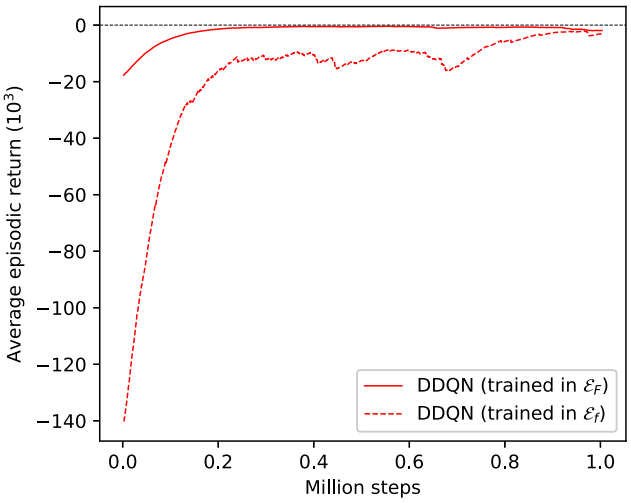


Fig. 7. Average episodic return for the DDQN agents trained with the simulation $\mathcal{E}_F$ and the actual $\mathcal{E}_f$ environments. Both environments expose the same observations and control actions to the agent, but the simulation environment is configured with the RC model, while the actual environment is configured with the BOPTEST emulator.

the MPC and the value function obtained from the RL agent. At every time step t_k, the same state observer as the one used by the MPC provides an estimate of the vector of states $\hat{\boldsymbol{x}}_k$ to initialize the $\mathcal{E}_F$ environment. Then, this environment evaluates the immediate reward of taking every possible action. The advantage of using exhaustive search for the optimization of the first step in the horizon is that the environment $\mathcal{E}_F$ also delivers the expected next state $\boldsymbol{s}'$, such that it can directly evaluate $v_*(\boldsymbol{s}')$ and sum it to the immediate reward. This approach would be prohibitive for larger action spaces that would require a function mapping between $\boldsymbol{x}$ and $\boldsymbol{s}$. We prefer the exhaustive search approach from the convenience of having $\mathcal{E}_F$ available.

6. Simulation results

Fig. 7 shows the evolution of the average episodic return during the off-line learning process, i.e., during the one million steps of interaction with the simulation environment $\mathcal{E}_F$. The steady increase of the expected episodic return indicates that DDQN can properly learn from experience. For the sake of benchmarking, the same algorithm is separately trained with the actual environment, $\mathcal{E}_f$, and using the same random seed for exploration. Although unrealistic, the latter scenario provides an upper performance bound for testing. The comparison of both learning curves in Fig. 7 reveals an overestimation of the rewards when the simulation environment is used for training. The reason is the less dynamic behavior of the RC model that leads to a more gentle response of the zone operative temperature to the random actions of the agent. The agent is thus overoptimistic when trained with simulated experience and thinks that it can easily maintain the indoor temperature within the comfort bounds. This effect has a catastrophic consequence during testing, as is clear from Fig. 8 which shows that the agent trained in the simulation environment cannot maintain the controlled variable within bounds during testing. Contrarily, the agent trained in the actual environment can successfully keep the temperature within bounds. It is concluded that the false cues received from $\mathcal{E}_F$ mislead the agent leading to poor behavior during testing. A similar observation of simulation-based RL in buildings was already pointed out in [41].

Interestingly, the MPC results of Fig. 8 reveal that MPC uses the controller model more effectively than the DDQN algorithm because it respects the constraints when using the same controller model and an equivalent control configuration. This performance difference stems from the different machinery of each algorithm: MPC performs state estimation and dynamic optimization every control step, which leads to a higher on-line computational intensity, but allows to decide on best actions based on an accurate representation of the current system state at all times; conversely, RL only requires a function evaluation to decide best actions on-line (which is computationally very efficient), but relies on a parametrized value function that has been trained off-line and that can hardly infer the current system state.

The poor constraint satisfaction of the DDQN algorithm is eliminated by the implementation of RL-MPC, which incorporates a state estimator to accurately track the state of the environment and an optimizer to evaluate actions during the first horizon step. The long-term behavior is accounted for with the same value function trained with the DDQN algorithm in the simulation environment. To emphasize the effect of the value function in RL-MPC, the same algorithm using $\gamma = 0$ has been implemented. This represents a controller with a one-step-ahead prediction horizon that disregards posterior behavior since the value function is disabled in the objective.

It is apparent from Fig. 8 that the state estimator and the one-step-ahead optimizer substantially aid the RL-MPC agent to satisfy constraints when compared to the DDQN agent. Moreover, the value function positively contributes to the controller performance, which can be seen from the comparison with RL-MPC when $\gamma = 0$. The inability to predict beyond one time step period leads to constraint violations with the myopic agent at occupancy setbacks. This effect is most evident during the peak heating test because of the increased heat demand.

The KPIs obtained with each controller at the end of the two-week testing periods are summarized in Fig. 9. These KPIs, among others, are directly obtained from the BOPTEST framework and have been selected based on the designed control objective. The horizontal lines serve as a reference of reasonable thermal discomfort levels when following the recommended criteria for acceptable deviations of standard EN16798-2 [75]. Particularly, these horizontal lines are obtained when using weighting factors of 1, 2, and 3 K during the short time deviations allowed in this standard. In our case, the temperature limits are established by the predefined BOPTEST temperature comfort range. Note that there exist other environmental factors that should be addressed when providing comfort for occupants, like thermal radiation, humidity, and air speed, or personal factors like activity and clothing. For the sake of clarity, our comfort assessment is based on temperature only since it is the main factor influencing the thermal comfort perceived by the occupants in a building [76].

The results of the DDQN agent trained in $\mathcal{E}_F$ are located at higher values and thus omitted in Fig. 9 to preserve the scale. The DDQN agent trained in the actual environment maintains a low thermal discomfort but at a higher operational cost than the MPC strategy. The DDQN algorithm trained in $\mathcal{E}_f$ achieves slightly lower discomfort during the peak heating period than MPC, that penalizes thermal discomfort more heavily than the energy cost. However, this does not hold for the typical heating period, where the MPC clearly outperforms DDQN, even when the latter has been trained in the actual environment and the former only uses a simplified controller model. The use of RL-MPC substantially improves the results and leads the agent to achieve similar performance levels as the MPC even for the deterministic setting of this example. The importance of the value function in the objective is underlined by the comparison with the myopic DDQN agent with $\gamma = 0$ that incurs substantially higher discomfort, especially during the peak testing period.

7. Discussion

There are two main ways to merge MPC and RL: truncating the objective function or using the MPC as the function approximator of the learning agent. The former is the approach adopted by RL-MPC and described in this work, and the latter is the approach proposed by other recently developed algorithms, namely Differentiable MPC and its associated Gnu-RL. RL-MPC constrains the actions explicitly (using

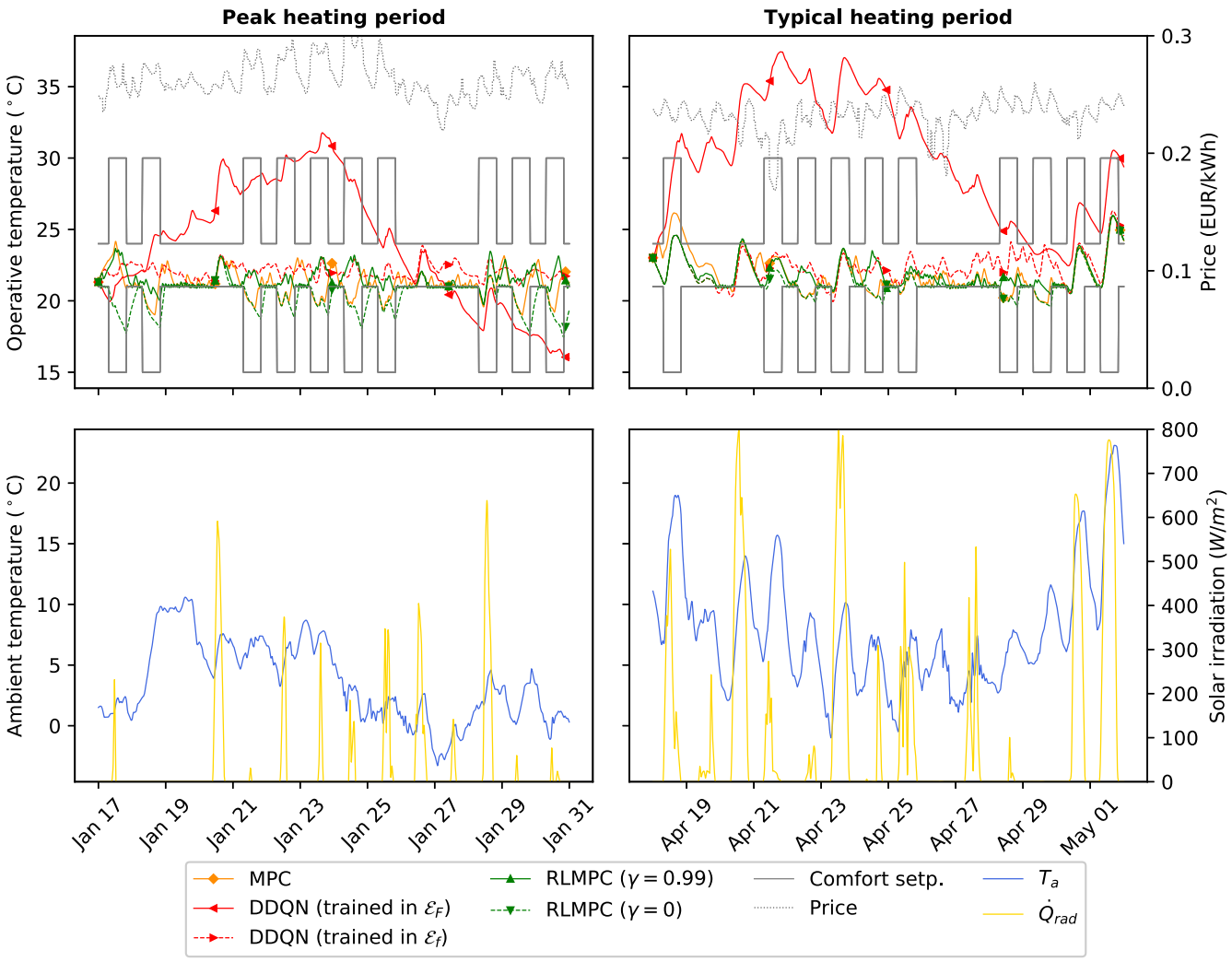


Fig. 8. Simulation results for all controllers in each of the BOPTEST testing periods for the envisaged building. The plots at the top show the pricing signal, the comfort constraints, and the evolution of the zone operative temperature T_z for each control strategy. The plots at the bottom show the main system disturbances, namely the ambient temperature and the direct solar irradiation.

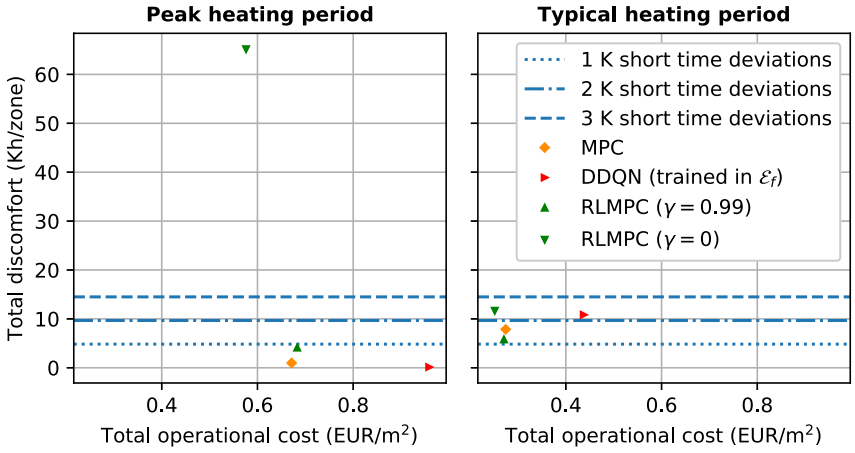


Fig. 9. KPIs for each controller and the two testing periods as obtained from the BOPTEST framework. Each of the testing periods lasts for two weeks. The horizontal lines serve as a reference of reasonable thermal discomfort levels when following the recommended criteria for acceptable deviations of standard EN16798-2 [75].

the controller model) during the first control step, and implicitly (using the value function) along the rest of the prediction horizon. Differentiable MPC constrains the actions explicitly throughout the whole prediction horizon and uses the rewards to tune the weights of the objective function. These are just two ways of constraining the agent to improve operational safety and it is still an open question which approach is preferred over the other. Probably, Differentiable MPC is more conservative as it imposes the system constraints throughout the

prediction horizon. This may lead to safer operation but offers less degrees of freedom to learn from the environment. Both approaches should be compared in BOPTEST to find out which one is preferred for the application in building climate control.

It is also important to note that, although BOPTEST is a simulation framework, it uses high-fidelity emulator building models. Uncertainties (e.g. related to weather predictions, measurements, model mismatch) are not yet included, but the provided emulators constitute a detailed representation of the system dynamics and only expose a realistic set of measurements and setpoint variables. Therefore, it is expected that the results obtained in the BOPTEST framework are representative of what would be obtained in real implementations.

In this work, the implementation of RL-MPC in BOPTEST has clearly outperformed classical RL. RL-MPC has also shown similar performance to the MPC in a deterministic setting, but RL-MPC incorporates the important advantage of learning as in a classical RL approach: RL-MPC learns the value function from rewards that are directly retrieved from the environment, and an analytical form is therefore not required. This opens the path towards learning perceived comfort from occupants e.g. through an App where people could indicate their thermal sensation from different levels as defined by ANSI/ASHRAE Standard 55-2010, or towards accounting long-term dynamics that cannot be captured by the typical MPC horizons e.g. in thermal systems with very large thermal inertia like geothermal borefields. Moreover, RL-MPC is expected to excel in uncertain environments where the uncertainty distribution could be learned by the agent and accounted for during operation. Confirmation of this hypothesis is a topic for future research.

This paper has focused on the definition and empirical demonstration of RL-MPC, but the current implementation of this new algorithm can be further improved. A major challenge for the implementation of RL-MPC relates to setting up an optimization framework comprising the software dependencies required for both: MPC and RL. Dedicated frameworks exist for either one or the other, but, to the best of our knowledge, there is no unified framework facilitating the configuration and implementation of both algorithms together. Moreover, the truncation of the RL-MPC algorithm's prediction horizon requires estimating the next state and its associated value (expected return) at each iteration of the optimization. Therefore, another major challenge is posed in the implementation of RL-MPC to incorporate the value function in the MPC optimization. The current implementation uses exhaustive search: the controller model simulates every possible action from the state vector obtained with the state observer. The simulation environment is used for this, such that the expected reward and the next expected state are returned from the simulation of each action. The sum of the immediate reward and the value of the expected next state is evaluated for each possible action to decide the action that leads to the highest expected value of this sum. This is obviously not computationally efficient. The most efficient way of implementing RL-MPC would be to extend the controller model with the equations that define the value function. Both could be optimized together using more effective optimization techniques.

8. Conclusion and future work

Buildings require advanced control algorithms for the efficient use of their energy systems. The control theory and machine learning communities are both working on solutions using MPC and RL, respectively, and prior literature has expressed interest in merging both methods. The complementarity is underlined from a conceptual reflection on their principal aspects like their approach, optimality, or computational complexity. This paper presents and assesses reinforced model predictive control RL-MPC, an algorithm that effectively combines elements from RL and MPC like state estimation, dynamic optimization, and learning. The BOPTEST framework is a standardized environment for the assessment and benchmarking of control algorithms in buildings. This simulation environment allows consistent and repeatable testing for the same building and boundary conditions. The results for one test case building of the BOPTEST framework reveal that MPC makes effective use of the controller model while RL incurs severe constraint violations using an equivalent formulation of the control problem. Both implementations are combined in the RL-MPC algorithm, which can satisfactorily meet the constraints. The new algorithm obtains performance results similar to the MPC in a deterministic setting even when using an imperfect value function. It also enables learning as in a classical RL approach which allows to naturally deal with uncertain environments or complex rewards without requiring their analytical form. Future research should compare RL-MPC with other methods merging MPC and RL like Differentiable MPC. Additionally, the current implementation of RL-MPC may be further improved by extending the controller model with the equations that define the value function to use efficient optimization techniques and enable scalability of the algorithm. An unified and open-source framework for optimal control may facilitate this task.

CRediT authorship contribution statement

Javier Arroyo: Conceptualization, Methodology, Software, Formal analysis, Data Curation, Writing – original draft, Writing – review & editing, Visualization, Funding acquisition. **Carlo Manna:** Methodology, Writing – original draft, Writing – review & editing. **Fred Spiessens:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition. **Lieve Helsen:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work emerged from the IBPSA Project 1, an international project conducted under the umbrella of the International Building Performance Simulation Association (IBPSA). Project 1 will develop and demonstrate a BIM/GIS and Modelica Framework for building and community energy system design and operation. The work of Javier Arroyo is financed by VITO, Belgium through a PhD Fellowship (grant number 1710754). Finally, the authors wish to thank to Brida V. Mbuwir, Ján Drgoňa, and Iago Cupeiro Figueroa for kindly reviewing the paper.

References

[1] IEA, GlobalABC, UN Environmental Programme. Global status report for buildings and construction: Towards a zero-emissions, efficient and resilient buildings and construction sector. 2020, https://wedocs.unep.org/handle/20.500.11822/34572.

[2] De Coninck R, Helsen L. Quantification of flexibility in buildings by cost curves – Methodology and application. Appl Energy 2016;162:653–65.

[3] Arroyo J, Gowri S, De Ridder F, Helsen L. Flexibility quantification in the context of flexible heat and power for buildings. 2018, Proceedings of the Federation of European Heating, Ventilation and Air Conditioning associations conference, Brussels, Belgium,

[4] Mbuwir BV, Geysen D, Spiessens F, Deconinck G. Reinforcement learning for control of flexibility providers in a residential microgrid. IET Smart Grid 2019;3(1):1–11.

[5] Schwarm AT, Nikolaou M. Chance-constrained model predictive control. AIChE J 1999;45(8):1743–52.

[6] Bemporad A, Morari M. Robust model predictive control: A survey. In: Garulli A, Tesi A, editors. Robustness in identification and control. Lecture notes in control and information sciences, vol. 245, London: Springer; 1999, p. 207–26.

[7] Bacci E, Parker D. Probabilistic guarantees for safe deep reinforcement learning. 2020, arXiv, https://arXiv.org/abs/2005.07073.

[8] Koller T, Berkenkamp F, Turchetta M, Boedecker J, Krause A. Learning-based model predictive control for safe exploration and reinforcement learning. 2019, arXiv, https://arxiv.org/abs/1906.12189.

[9] Ernst D, Glavic M, Capitanescu F, Wehenkel L. Reinforcement learning versus model predictive control: A comparison on a power system problem. IEEE Trans Syst Man Cybern B 2009;39(2):517–29.

[10] Negenborn RR, De Schutter B, Wiering MA, Hellendoorn H. Learning-based model predictive control for Markov decision processes. In: IFAC Proceedings volumes of the 16th IFAC world congress, vol. 38, (1):2005, p. 354–9.

[11] Recht B. A tour of reinforcement learning: The view from continuous control. 2018, arXiv, http://arxiv.org/abs/1806.09460.

[12] Görges D. Relations between model predictive control and reinforcement learning. IFAC-PapersOnLine 2017;50(1):4920–8. http://dx.doi.org/10.1016/j.ifacol.2017.08.747, 20th IFAC World Congress.

[13] Dulac-Arnold G, Mankowitz D, Hester T. Challenges of real-world reinforcement learning. 2019, arXiv, https://arxiv.org/abs/1904.12901.

[14] Ojand K, Dagdougui H. Q-learning-based model predictive control for energy management in residential aggregator. IEEE Trans Autom Sci Eng 2021;1–12.

[15] Chen B, Cai Z, Bergés M. 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. New York, New York, USA; 2019. p. 316–325.

[16] Blum D, Arroyo J, Huang S, Drgoňa J, Jorissen F, Taxt Walnum H, et al. Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings. J Build Perform Simul 2021 14(5):586–610.

[17] Arroyo J, Manna C, Spiessens F, Helsen L. An OpenAI-Gym environment for the Building Optimization Testing (BOPTEST) framework. In: Proceedings of the 17th IBPSA Conference. Bruges, Belgium. September 2021.

[18] Sturzenegger D, Gyalistras D, Morari M, Smith RS. Model predictive climate control of a Swiss office building: Implementation, results, and cost-benefit analysis. IEEE Trans Control Syst Technol 2016;24:1–12.

[19] Mason K, Grijalva S. A review of reinforcement learning for autonomous building energy management. Comput Electr Eng 2019;78:300–12.

[20] Drgoňa J, Arroyo J, Cupeiro Figueroa I, Blum D, Arendt K, Kim D, et al. All you need to know about model predictive control for buildings. Annu Rev Control 2020;50:190–232, Published on-line https://doi.org/10.1016/j.arcontrol.2020.09.001.

[21] Drgoňa J, Klaučo M, Kvasnica M. MPC-based reference governors for thermostatically controlled residential buildings. In: Proceedings of the 54th IEEE conference on decision and control. Osaka, Japan; 2015. p. 1334–9. http://dx.doi.org/10.1109/CDC.2015.7402396.

[22] Jorissen F, Boydens W, Helsen L. TACO, an automated toolchain for model predictive control of building systems: Implementation and verification. J Build Perfor Simul 2018;12(2):180–92.

[23] Wetter M, Zuo W, Nouidui T, Pang X. Modelica buildings library. J Build Perfor Simul 2014;7(4):253–70.

[24] Jorissen F, Reynders G, Baetens R, Picard D, Saelens D, Helsen L. Implementation and verification of the IDEAS building energy simulation library. J Build Perfor Simul 2018;11(6):669–88.

[25] Müller D, Lauster M, Constantin A, Fuchs M, Remmen P. AixLib – An open-source modelica library within the IEA-EBC annex 60 framework. In: Proceedings of the BauSIM IBPSA conference. Dresden, Germany; 2016. p. 3–9.

[26] Nytsch-Geusen C, Banhardt C, Inderfurth A, Mucha K, Möckel J, Rädler J, et al. Buildingsystems - Eine modular hierarchische Modell-Bibliothek zur energetischen Gebäude und Anlagensimulation. In: Proceedings of the BauSIM IBPSA conference. Dresden, Germany; 2016. p. 473–480.

[27] Coninck RD, Magnusson F, Åkesson J, Helsen L. Toolbox for development and validation of grey-box building models for forecasting and control. J Buil Perform Simul 2016;9(3):288–303.

[28] Atam E, Helsen L. Control-oriented thermal modeling of multizone buildings: Methods and issues: Intelligent control of a building system. IEEE Control Syst Mag 2016;36(3):86–111.

[29] Arroyo J, Spiessens F, Helsen L. Identification of multi-zone grey-box building models for use in model predictive control. J Build Perform Simul 2020;13(4):472–86.

[30] Mnih V, Kavukcuoglu K, Silver D, Rusu A, Veness J, Bellemare M, et al. Human-level control through deep reinforcement learning. Nature 2015;518:529–33.

[31] Vázquez-Canteli JR, Nagy Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89.

[32] Vázquez-Canteli JR, Ulyanin S, Kämpf J, Nagy Z. Fusing TensorFlow with building energy simulation for intelligent energy management in smart cities. Sustainable Cities Soc 2019;45:243–57.

[33] Nagai T. Dynamic optimization technique for control of HVAC system utilizing building thermal storage. In: Proceedings of the 6th building simulation conference, vol. I. Kyoto, Japan; 1999. p. 1311–7.

[34] Peirelinck T, Ruelens F, Decnoninck G. Using reinforcement learning for optimizing heat pump control in a building model in modelica. In: Proceedings of the IEEE international energy conference, ENERGYCON. 2018, p. 1–6.

[35] An energy-efficient predictive control for HVAC systems applied to tertiary buildings based on regression techniques. Energy Build 2017;152:409–17.

[36] Nagy A, Kazmi H, Cheaib F, Driesen J. Deep reinforcement learning for optimal control of space heating. 2018, arXiv, https://arxiv.org/abs/1805.03777.

[37] Mbuwir BV, Spiessens F, Deconinck G. Benchmarking regression methods for function approximation in reinforcement learning: Heat pump control. In: Proceedings of the IEEE PES innovative smart grid technologies Europe. Bucharest, Romania; 2019. p. 1–5.

[38] Patyn C, Ruelens F, Deconinck G. Comparing neural architectures for demand response through model-free reinforcement learning for heat pump control. In: Proceedings of the IEEE international energy conference. Limassol, Cyprus; 2018. p. 1–6. http://dx.doi.org/10.1109/ENERGYCON.2018.8398836.

[39] Picard D, Drgoňa J, Kvasnica M, Helsen L. Impact of the controller model complexity on model predictive control performance for buildings. Energy Build 2017;152:739–51.

[40] Zhang Z, Lam K. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In: Proceedings of the 5th ACM international conference on systems for energy-efficient buildings, cities, and transportation. Shenzen, China; 2018. p. 148–157. http://dx.doi.org/10.1145/3276774.3276775.

[41] Liu S, Henze GP. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis. Energy Build 2006;38(2):148–61.

[42] Scharnhorst P, Schubnel B, Fernández Bandera C, Salom J, Taddeo P, Boegli M, et al. Energym: A building model library for controller benchmarking. Appl Sci 2021;11(8). http://dx.doi.org/10.3390/app11083518.

[43] Vázquez-Canteli JR, Kämpf J, Henze G, Nagy Z. CityLearn v1.0: An OpenAI gym environment for demand response with deep reinforcement learning. In: Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation. New York, New York, USA; 2019. p. 356–7. http://dx.doi.org/10.1145/3360322.3360998.

[44] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. 2015, arXiv, https://arXiv.org/abs/1603.04467v2.

[45] Hewing L, Wabersich KP, Menner M, Zeilinger MN. Learning-based model predictive control: Toward safe learning in control. Annu Rev Control Robot Auton Syst 2020;3(1):269–96.

[46] Liu S, Henze GP. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 1. Theoretical foundation. Energy Build 2006;38(2):142–7.

[47] Amos B, Rodriguez IDJ, Sacks J, Boots B, Kolter JZ. Differentiable MPC for end-to-end planning and control. 2019, arXiv, arXiv:1810.13400.

[48] Drgoňa J, Tuor A, Skomski E, Vasisht S, Vrabie D. Deep learning explicit differentiable predictive control laws for buildings. 2021, arXiv, arXiv:abs/2107.11843v1.

[49] Gros S, Zanon M. Data-driven economic NMPC using reinforcement learning. IEEE Trans Automat Control 2020;65(2):636–48.

[50] Gros S, Zanon M. Reinforcement learning for mixed-integer problems based on MPC. 2020, arXiv, arXiv:2004.01430.

[51] Gros S, Zanon M. Reinforcement learning based on MPC and the stochastic policy gradient method. In: 2021 American control conference. 2021, p. 1947–52.

[52] Zanon M, Gros S. Safe reinforcement learning using robust MPC. IEEE Trans Automat Control 2021;66(8):3638–52.

[53] Gros S, Zanon M. Safe reinforcement learning with stability & safety guarantees using robust MPC. 2020, arXiv, arXiv:2012.07369.

[54] Kamthe S, Deisenroth MP. Data-efficient reinforcement learning with probabilistic model predictive control. CoRR 2017;abs/1706.06491.

[55] Zhang H, Li S, Zheng Y. Q-learning-based model predictive control for nonlinear continuous-time systems. Ind Eng Chem Res 2020;59(40):17987–99.

[56] Buşoniu L. BR. Approximate dynamic programming and reinforcement learning. Technical report. In: Interactive collaborative information systems. Studies in computational intelligence, vol. 281, Berlin, Heidelberg: Springer; 2010.

[57] Drgoňa J, Picard D, Kvasnica M, Helsen L. Approximate model predictive building control via machine learning. Appl Energy 2018;218:199–216.

[58] Kelly MP. Transcription methods for trajectory optimization: A beginners tutorial. 2017, arXiv, https://arxiv.org/abs/1707.00284.

[59] Fletcher R. Practical methods of optimization. 2nd ed.. Wiley-Interscience; 1987, https://dl.acm.org/doi/book/10.5555/39857.

[60] Ernst D, Geurts P, Wehenkel L. Iteratively extending time horizon reinforcement learning. In: Lavrač N, Gamberger D, Blockeel H, Todorovski L, editors. Proceedings of the 14th European conference on machine learning. Dubrovnik, Croatia; 2003. p. 96–107.

[61] Chua K, Calandra R, McAllister R, Levine S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. 2018, arXiv, http://arxiv.org/abs/1805.12114.

[62] Bhardwaj M, Handa A, Fox D, Boots B. Information theoretic model predictive Q-learning. 2020, arXiv, https://arXiv.org/abs/2001.02153.

[63] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing atari with deep reinforcement learning. 2013, arXiv, http://arxiv.org/abs/1312.5602.

[64] Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, et al. Mastering Chess and Shogi by self-play with a general reinforcement learning algorithm. 2017, arXiv, http://arxiv.org/abs/1712.01815.
[65] Weber T, Racanière S, Reichert DP, Buesing L, Guez A, Rezende DJ, et al. Imagination-augmented agents for deep reinforcement learning. 2017, arXiv, http://arxiv.org/abs/1707.06203.
[66] Cupeiro Figueroa I, Drgoňa J, Helsen L. State estimators applied to a linear white-box geothermal borefield controller model. In: Proceedings of the 16th international conference of IBPSA. 2019.
[67] Vande Cavey M, Bonvini M, Helsen L. Comparison and application of different state estimation techniques for control in buildings. In: Workshop on optimal control of thermal systems in buildings using modelica. Freiburg, Germany; 2015.
[68] Åkesson J, Årzén KE, Gäfvert M, Bergdahl T, Tummescheit H. Modeling and optimization with optimica and JModelica.org-Languages and tools for solving large-scale dynamic optimization problems. Comput Chem Eng 2010;34:1737–49.
[69] Axelsson M, Magnusson F, Henningsson T. A framework for nonlinear model predictive control in JModelica.org. In: Proceedings of the 11th international modelica conference, vol. 118. Versailles, France. 2015. p. 301–10.
[70] Andersson J, Gillis J, Horn G, Rawlings J, Diehl M. CasADi – A software framework for nonlinear optimization and optimal control. Math Program Comput 2019;11(1):1–36.
[71] Sun F, Li G, Wang J. Unscented Kalman filter using augmented state in the presence of additive noise. In: Proceedings of the IITA international conference on control, automation and systems engineering. Zhangjiajie, China; 2009, p. 379–82.
[72] Wan EA, Merwe RVD. The unscented Kalman filter for nonlinear estimation. In: Proceedings of the IEEE adaptive systems for signal processing, communications, and control symposium. Alberta, Canada; 2000. p. 153–158.
[73] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. OpenAI Gym. 2016, arXiv, http://arxiv.org/abs/1606.01540.
[74] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. 2015, arXiv, http://arxiv.org/abs/1509.06461.
[75] Technical Committee CENTC 156 "Ventilation for Buildings". EN16798. Energy performance of buildings - Ventilation for buildings - Part 2: Interpretation of the requirements in EN 16798-1 - Guideline for using indoor environmental input parameters for the design and assessment of energy performance of buildings. London, BSI: BSI; 2019.
[76] Barbato A, Bolchini C, Geronazzo A, Quintarelli E, Palamarciuc A, Pitì A, et al. Energy optimization and management of demand response interactions in a smart campus. Energies 2016;9:398.