Applied Energy 218 (2018) 199–216

Contents lists available at ScienceDirect

Applied Energy

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

Approximate model predictive building control via machine learning

Ján Drgoňa[a,*], Damien Picard[a], Michal Kvasnica[b], Lieve Helsen[a,c]

[a] KU Leuven, Department of Mechanical Engineering, Leuven, Belgium
[b] Slovak University of Technology in Bratislava, Radlinského 9, SK-812 37 Bratislava, Slovak Republic
[c] EnergyVille, Thor Park, Waterschei, Belgium

HIGHLIGHTS

- The construction of approximate model predictive control laws (MPC) is presented.
- Easy implementation of advanced control strategies suitable for low-level hardware.
- Multivariate regression and dimensionality reduction algorithms are used.
- Simulation case study employing temperature control in a six-zone building.
- Simplified control laws retain most of the performance of the original MPC.

ARTICLE INFO

Keywords:
Building climate control
Model predictive control
Machine learning
Dimensionality reduction
Time delay neural networks
Regression trees

ABSTRACT

Many studies have proven that the building sector can significantly benefit from replacing the current practice rule-based controllers (RBC) by more advanced control strategies like model predictive control (MPC). However, the optimization-based control algorithms, like MPC, impose increasing hardware and software requirements, together with more complicated error handling capabilities required from the commissioning staff. In recent years, several studies introduced promising remedy for these problems by using machine learning algorithms. The idea is based on devising simplified control laws learned from MPC. The main advantage of the proposed methods stems from their easy implementation even on low-level hardware. However, most of the reported studies were dealing only with problems with a limited complexity of the parametric space, and devising laws only for a single control variable, which inevitably limits their applicability to more complex building control problems. In this paper, we introduce a versatile framework for synthesis of simple, yet well-performing control strategies that mimic the behavior of optimization-based controllers, also for large scale multiple-input-multiple-output (MIMO) control problems which are common in the building sector. The approach employs multivariate regression and dimensionality reduction algorithms. Particularly, deep time delay neural networks (TDNN) and regression trees (RT) are used to derive the dependency of multiple real-valued control inputs on parameters. The complexity of the problem, as well as implementation cost, are further reduced by selecting the most significant features from the set of parameters. This reduction is based on straightforward manual selection, principal component analysis (PCA) and dynamic analysis of the building model. The approach is demonstrated on a case study employing temperature control in a six-zone building, described by a linear model with 286 states and 42 disturbances, resulting in an MPC problem with more than thousand of parameters. The results show that simplified control laws retain most of the performance of the complex MPC, while significantly decreasing the complexity and implementation cost.

1. Introduction

The total energy used in heating, cooling, ventilation and air-conditioning (HVAC) systems in commercial and residential buildings nowadays accounts for 40% of the global energy use [1]. Numerous studies reported that advanced HVAC control could significantly reduce the energy use and mitigate emissions of greenhouse gases, see, e.g. [2–4]. However, currently, the majority of the building control strategies adopt simple rule-based logic with only limited energy saving capabilities [5,6].

One of the control methods exploiting the full potential of the building's HVAC systems is model predictive control (MPC) [7]. The

* Corresponding author.
E-mail addresses: jan.drgona@kuleuven.be (J. Drgoňa), damien.picard@kuleuven.be (D. Picard), michal.kvasnica@stuba.sk (M. Kvasnica), lieve.helsen@kuleuven.be (L. Helsen).

https://doi.org/10.1016/j.apenergy.2018.02.156
Received 6 October 2017; Received in revised form 22 February 2018; Accepted 23 February 2018
Available online 09 March 2018

high performance of MPC is achieved by minimizing the energy use and maintaining high comfort standards while taking into account technological restrictions, weather forecasts and building dynamics. In MPC, control inputs that minimize a certain objective function (which accounts for energy use and thermal discomfort) subject to constraints are computed by solving a corresponding optimization problem at each sampling instant. In recent years, many energy efficient MPC approaches have been reported for control of HVAC systems [8–14], household's hot water systems [15,16], or incorporate demand response objectives [17,18].

Despite this intensive research efforts, the transfer of this technology to the commercial sector is still in its early stages, mainly because of the following four reasons as pointed out by [19]. First, an accurate yet simple building model is required. However, obtaining such well-performing model with a minimum of effort is a difficult and time-consuming task [20]. As a promising solution, in recent years a methodology for the automatic synthesis of a global model of a building and its equipment based on exploiting ontology-based Building Information Models (BIM) has been proposed [21,22]. Second, design and tuning of MPC controllers are challenging, because the commission engineers are usually not trained to set up such complex control systems based on numerical optimization. Moreover, contrary to the industrial applications of MPC, buildings are not operated with on-site engineers monitoring and supervising the functioning of the employed control system. For these reasons, there is a strong request in this field for a simple implementation of the control algorithms without loss of their high energy efficient performance [23]. Third, there is also a strong need for data availability and processing power as the computation of MPC control actions for complex systems can be easily based on hundreds or even thousands of parameters/states. These variables are provided either by direct real-time measurements from the network of sensors, by external services like weather forecasts, or by state estimators. And fourth, the on-line solution of the corresponding optimization problem and the extensive data processing impose considerable challenges on hardware and software infrastructure, which is not a standard in today's buildings.

Although some approaches for a fast and simple on-line implementation of MPC for building control applications have been suggested previously [24,25], the task remains very challenging, especially when using existing control hardware, such as programmable logic controllers (PLC). There are two main difficulties. First, such a simple hardware provides only limited computational capabilities with a limited amount of memory storage (typically in the range of kilobytes). Second, most PLCs do not allow the control algorithm to be implemented in high-level languages. As a result, implementation of the complex, optimization-based control algorithms on a simple hardware is cumbersome [26].

The ambition of this paper is to tackle the last three challenges from the list of issues by constructing a simple, yet well-performing, control policy that offers a smooth implementation suitable for low-level hardware. One approach for achieving this goal is to calculate the explicit representation of the MPC feedback law [27,28]. For a rich class of MPC problems, the explicit solution takes the form of a piecewise affine (PWA) function defined over a polyhedral domain of the parametric space. Obtaining the optimal control input then reduces to a mere function evaluation. Such a task can be easily performed, even by a simple hardware [29], and can be extended e.g. to stochastic MPC formulations [30]. The fundamental limitation of explicit MPC solution, however, is that the complexity of the computed PWA control law grows exponentially with the dimensionality of the parametric space imposed by prediction horizon and number of variables. Therefore it can be applied only on the hardware with storage capacity large enough to accommodate the PWA function. However, this is usually not a realistic assumption, since the size of explicit MPC solutions can easily exceed several megabytes even for systems with low complexity, making it infeasible for complex building control problems with several thousands of parameters.

An alternative way to tackle this problem is to employ approximations of the MPC solution. The central idea here is not new. In fact, a variety of approximate explicit MPC solutions has been proposed in the literature [31]. In general, there are two groups of approaches. The first group is represented by geometric methods that are based on an efficient polyhedral partitioning of the state space [32–34]. The second group consists of data-driven function approximation methods. The earliest work in this area was based on neural networks [35], while more recent works based on e.g. PWA [36], polynomial [37], and nonlinear [38] function approximations, or wavelet interpolations [39] have been reported as well.

One of the first attempts for the approximation of MPC laws in the building control context was introduced by [40]. This method is based on linear interpolations of MPC solutions generated for a grid of selected parameters. However, the complexity of such grid approach is increasing exponentially with the number of parameters, which strongly limits its applicability to large scale problems. Other researchers [23,41,42] used classification algorithms for extracting simple decision rules from MPC employing logical control actions. Approaches approximating the continuous control laws are also available, e.g., based on piecewise linear mixing architecture [43] or nonlinear regression employing Hammerstein-Wiener models [44]. The approximation of the continuous MPC laws via enhanced regression trees, with piecewise linear approximation, was presented in [45]. All the approaches listed above were developed and tested only on problems with modest complexity, usually with single (continuous or binary) control variable, and with only dozens of parameters.

Moreover, only few of these papers give an increased attention to the feature engineering (FE) process. However, due to the strong influence of the features on the performance of the trained model, FE is considered to be a fundamental part of almost any practical machine learning application [46]. The gains from using only the most relevant features are threefold: first, improved performance, second, reduced complexity, and third, improved interpretability of the developed models. This is, however, typically where most of the effort in a machine learning project goes, learning is often the quickest part [47]. Engineering features properly is primarily a manual, difficult and time-consuming task, mostly because it is domain-specific, while learners can be largely general-purpose. Additionally, each type of the model will respond differently to different types of engineered features [46]. Therefore, one of the holy grails of machine learning is to automate the feature engineering process [47]. For these reasons a substantial research interest has appeared in recent years into the development of the feature learning algorithms, see, e.g. [48,49], or advanced semi-automated feature engineering systems [50]. The main drawback of these methods, however, is that the physical meaning of the original features is lost. This is a significant drawback in our case, as we are interested in the selection of the most relevant features as a subset of the original feature vector, without the loss of their physical meaning. For this task several feature selection (FS) methods suitable for building energy models are available in the literature nowadays. In [51] authors presented a heuristic approach tailored for support vector machine (SVM) models, while in [52] a statistical Wald's test has been used for identification of irrelevant features for neural network (NN) models. However, the search for all relevant features is, in general, an NP-hard problem [53].

In this paper, we propose a compact methodology for the construction of simple suboptimal MPC-like control strategies for building control applications by using advanced machine learning algorithms. The focus is on the creation of a systematic and universal framework applicable to a variety of large-scale building control problems, while providing valuable insights into the selection of the most relevant features and appropriate type of approximation model. We investigate two multivariate regression algorithms. First, we use regression trees (RT), where the approximation of the control law is given as a binary tree.

Second, we use time delay neural networks (TDNN) with deep architecture, which act as universal approximators for time series problems. The performance of both algorithms is compared and discussed. To the authors' best knowledge, this is the first time that deep TDNN are used to mimic the behavior of MPC in the context of building control. Moreover, the proposed regression-based control policy is in no way tied to the particular MPC formulation. On the contrary, the procedure can be applied to training data generated by an arbitrary controller. As an example, the proposed approach can be used to extrapolate properties of controllers based on stochastic MPC formulations, such as those proposed by [54,55], or when nonlinear PMV-based thermal comfort criteria are included in the formulation [56,57]. In summary, the added value of this paper lies in devising computationally tractable MPC approximations for complex building control problems for multiple-input-multiple-output (MIMO) systems with hundreds or even thousands of parameters. Particularly, MPC with a prediction horizon equal to 22 steps for a linear model with 300 states, 6 controlled and 6 continuous control variables is used in the case study.

Recently, there has been an increasing research effort in learning control policies from data using Deep Learning, mostly in the robotics domain [58]. In general, there are two main categories for this task, by: methods based on supervised learning [59,60], and methods based on reinforcement learning (RL) [61,62]. The approach in this paper falls into the category of supervised learning which offers a viable way to train control policies when a sufficient amount of training data is available. However, as a handicap, it strongly depends on the quality of the provided teacher signal to imitate. Hence, in our case, the performance of the learned control policy is limited by the performance of the teacher MPC and thus by the accuracy of the controller model. Another limitation is that the learned controller does not come with an adaptive behavior per se. On the other hand, the second approach based on RL algorithms has been successfully used in learning of generalizing control policies and possesses the adaptive capabilities in its essence [63]. However, RL algorithms usually require a significant amount of experimental data which are costly to obtain in terms of substantial computational loads, thus limiting the practical applicability of this approach so far only to problems of modest dimensions. It does not appear to be ready yet for large-scale building control problems. Although, this is expected to change soon due to continually decreasing computation cost via dedicated hardware platforms like GPUs. In this context, the authors see a promising future research direction in merging both approaches by using the learned controller via supervision by the teacher MPC as an actor in the actor-critic scheme, which is a subclass of the RL family. As such, as sort of a hot start for the RL algorithm can be provided, reducing the need for huge loads of experimental data and significantly reducing the computational burden.

The paper is organized as follows. The overall methodology is presented in Section 2. Section 3 provides descriptions of a building model, control objectives, a traditional rule-based-control (RBC) approach and MPC formulation. Section 4 introduces the machine learning approximation of the MPC laws into more details, defining the problem of interest, discussing the selected regression algorithms and introducing the feature engineering techniques suitable for this class of problems. The setup and the results of the simulation case study on a six zones residential building are described in Section 5. Finally, in Section 6 the conclusions are drawn.

2. Methodology

This section introduces the methodology and provides a detailed step by step tutorial on the development of approximated MPC strategies with low complexity representations suitable for application in complex building control problems. The overall methodology is represented in Fig. 1 and can be compactly divided into three main parts: i, modeling part represented by first three blocks, ii, 'teacher' MPC control strategy development and evaluation represented by a fourth block, and iii, machine learning controller approximation and performance evaluation part represented by last three blocks in Fig. 1.

The accurate building model is a crucial prerequisite for the success of the model based control strategy [64]. To this end, an existing house with 6 zones is modelled with high accuracy using the open-source Modelica library `IDEAS` [65], a state-of-the-art BES program (see Section 3.1). In the next step, the Modelica non-linear building model is accurately linearized, and transformed into a linear time-invariant (LTI) state space model (SSM) [66]. The obtained SSM is further used for both simulations and controller model, such that no plant-model mismatch is considered. This case is therefore the theoretical benchmark, with the aim to investigate the upper performance bound of the proposed control strategy. In the next step, the MPC is formulated, tuned and implemented on-line in receding horizon mode in the MATLAB® environment. The performance of the 'teacher' MPC is evaluated based on simulations by calculating the thermal comfort and energy use. The simulation data are collected and serve as a training dataset for the machine learning approximation. Further, for the sake of the dimensionality reduction and improved approximation accuracy, only the most significant variables are selected as features in the machine learning model via various feature engineering techniques. The problem to be solved in the next step is the multivariate regression problem of finding the best model describing the relationship between multiple real-valued variables. For this task deep TDNNs are selected as universal function approximators [67]. The reason for TDNNs selection is that they are capable of handling complex multivariate time series regression problems that arise from the behavior of dynamic systems. Moreover, RT are also used for this task mainly because of their rule-based nature, making them suitable for easy implementation on today's building hardware. Next, the machine learning models are trained and tuned using the reduced data set obtained from simulations of the 'teacher' control strategy, in our case a linear MPC with quadratic cost. Finally, the best performing machine learning models approximating the behavior of the original 'teacher' MPC are evaluated by simulation. Additionally, the performances of the original and approximated MPC are compared altogether with a traditional RBC and a PID controllers.

Remark 1. In this paper, a white box modeling approach is used in order to increase the accuracy of the case study. However, the methodology of this paper is more general and suitable for any type of modeling approach, either white-, black- or grey-box models. Similarly, the presented methodology is not limited to a particular control strategy that serves as 'teacher' for the machine learning model. On the contrary, any type of advanced control strategies that require high computational resources can be approximated by this approach.

3. Building modeling and control

This section is based on previous work of the authors published in [68]. Building climate controllers are responsible for the comfort experienced in buildings. They control the HVAC of the building ensuring that comfort remains in its prescribed time-dependent bands for each room. A similar comfort can be achieved with different sequences of control actions, however the energy used can vary significantly with the controller type. Because of this, a substantial effort has been made in recent years in development and application of optimization based control strategies such as model predictive control (MPC) for building comfort control problems. The key part of all model based approaches such as MPC is the accurate building model, used either for prediction or for simulation of the dynamic behavior of the real building.

Section 3.1 describes a particular model of a 6-rooms house in the form of a linear time-invariant (LTI) state space model (SSM) derived by linearization of the accurate non-linear building model developed in the building energy simulation software Modelica. From control point of view we focus in this paper on the thermal comfort and energy use objectives, which are formulated in Section 3.2. Section 3.3 introduces

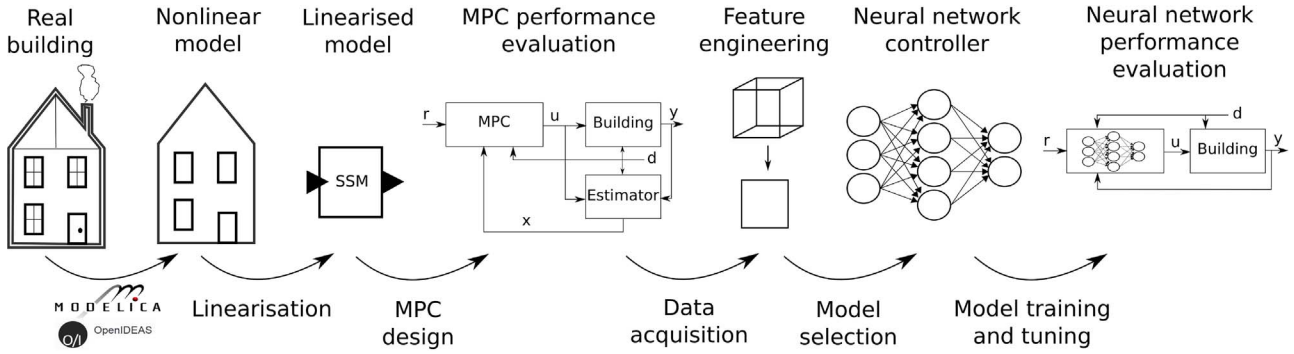


Fig. 1. Schematic view of the methodology. From left to right: a real building is modelled using the building energy simulation (BES) Modelica library IDEAS. The obtained nonlinear model is then linearized and converted to a state space model (SSM) representation. The model is employed for both emulating the building dynamics in the simulations, as well as a controller model in the MPC. The MPC is being implemented and evaluated in a MATLAB® environment through on-line coupling. The simulation data are collected and reduced in dimensions by selecting only the most significant feature variables. Next, the machine learning model is selected and further trained and tuned to mimic the behavior of the original 'teacher' MPC. Finally, the performance of the original and approximated MPC is being evaluated and compared with traditional controllers.

two conventional building control strategies as benchmark cases: a rule-based-controller (RBC) and a proportional-integral-derivative (PID) controller. Section 3.4 presents the formulation and implementation setup of the model predictive controller (MPC) considered in this paper.

3.1. Building model

This section describes the linear time-invariant (LTI) state space (SS) model of a residential building used in this study. The building model is based on an existing 6-rooms terraced house in Bruges, Belgium (see Fig. 2) with general parameter values summarized in Table 1. In this study, a building envelope, consisting of 6 thermal zones, 5 windows, 11 outer walls, 5 boundary walls with neighboring buildings, 6 roof surfaces, 3 floor surfaces on the ground, 3 floor surfaces between the ground floor and the first floor, and 6 internal walls between the zones, is considered. The heating system is composed of one radiator per room fed by a central gas-boiler. The heating system is idealized as perfectly controllable, with limited heating power, which can be directly injected in each room. The radiators and the gas boiler are thus not modeled but they are replaced by one heat input per zone. Cooling and ventilation are not considered.

The building is modeled in the building energy simulation (BES) program Modelica by using the IDEAS library [65]. IDEAS (Integrated District Energy Assessment by Simulation) is a recently developed district energy commodity flow modeling environment that enables multi-zone thermal building simulation, including building envelope, heating, ventilation and air-conditioning systems, and electric system simulation. The governing equations are discretized partial differential equations, ordinary differential equations and algebraic equations. However the Modelica building model, is not directly usable as controller model due to its high complexity. Therefore the Modelica reference building model is linearized in order to obtain a LTI SSM, which is directly suitable for the use in MPC as a controller model. For typical European weather the linearization error for this model remains below 1 K. However, the equations for the solar transmission and absorption through the windows are highly non-linear and they should not be linearized. The solar transmission and absorption are instead pre-computed using the IDEAS model and they are considered as inputs to the linearized SSM. For a detailed description of the developed building model we refer to [69,68], and for the linearization process to [66].

Fig. 2. Picture of the modeled house (Bruges, Belgium).

Table 1
General building parameters.

Floor area	[m^2]	48.3
Conditioned volume	[m^3]	130.6
Total exterior surface area	[m^2]	195
Window to wall ratio	[–]	19%
Windows orientation	[–]	North-East

The discretized LTI SS model has the following form:

$$x_{k+1} = Ax_k + Bu_k + Ed_k, \tag{1a}$$

$$y_k = Cx_k + Du_k, \tag{1b}$$

where x_k, u_k and d_k are states, inputs and disturbances at the k-th time step, respectively. The model is discretized with the sampling frequency T_s = 15minutes. Disturbances signal d_k represents the heat absorbed and the direct and diffuse solar radiation transmitted by each window, the direct, diffuse solar radiation and the environment temperature (i.e. a radiation temperature taking both the environment and the sky temperature into account) per orientation and inclination present in the model, the ambient temperature, and the ground temperature. In this study a standard weather file of Uccle, Belgium [70] is used to represent the weather conditions. All inputs of the non-linear and the linear models are exactly the same. The overall dimensions of the used building model are summarized in Table 2.

3.2. Control objectives

A Building Automation and Control System (BACS) control buildings such that certain comfort and economic criteria are fulfilled. Instead of tracking particular reference values, a BACS typically considers ranges, also called comfort bands. The task is then to manipulate

Table 2
Dimensions of the building model.

Notation	Description	Values
n_x	Number of states	286
n_u	Number of inputs	6
n_y	Number of outputs	6
n_r	Number of output references	6
n_d	Number of measured disturbances	44

Table 3
Notation and meaning of the variables used in control.

Notation	Units	Description	Control setup
x	[K]	Building temperatures	States
y	[K]	Room operative temperatures	Outputs
r	[K]	Desired room temperatures	References
u	[W]	Radiators heat flows	Inputs
d	[K, W]	Temperatures, heat flows and radiation gains	Disturbances
s	[K]	Comfort band violations	Slack variables
ub	[K]	Upper comfort boundary	Constraints
lb	[K]	Lower comfort boundary	Constraints

the building inputs such that required comfort criteria are kept within the band while the total amount of used energy is minimized. It should be noted that comfort and energy use criteria are often contradictory as the increase of comfort typically leads to the increase of energy use. The notation and meaning of the variables used in this section are presented in Table 3.

3.2.1. Thermal comfort

The thermal comfort objective is achieved by maintaining each room operative temperature y_i of the house in the comfort band as defined by the European standard ISO-7730. The lower and upper temperature bounds (lb,ub) vary between [20,23] °C and [24,26] °C, respectively, as a function of a 7-days average of the ambient temperature. The comfort objective corresponds thus to the constraint:

$$lb_k - s_k \leqslant y_{i,k} \leqslant ub_k + s_k \tag{2}$$

with s the slack variable that should be minimized, here the index k denotes the time index.

3.2.2. Energy use

The second objective is to use a minimal amount of energy to achieve the thermal comfort. In this paper, the energy use is the sum of the thermal energy injected by all radiators. The RBC and the PID controllers have been tuned such that they keep the zone temperatures as close as possible to the lower comfort bound, without violating the comfort constraints and the energy used is an output of the simulation and is not explicitly taken into account during the tuning process. For the case of MPC, in the objective function the square of the energy use is minimized in order to avoid power peaks and keep the optimization problem a quadratic program.

3.3. Conventional building control strategies

The current practice in building control is based on so called rule-based controllers (RBC). These controllers consists of a set of rules that determine the control action as a function of inputs (e.g., a room temperature, the outside weather conditions, etc.) and a set of set points. They are widely used mainly because of their simple design and low computational demands, which allows cheap hardware solutions. Their main drawbacks are however that they are not optimal, not adaptive, not predictive and usually they need to be manually tuned. The RBC implementation used in this paper is described in Section 3.3.1. An alternative conventional approach is the PID controller, which is better suited for tracking a reference signal, but performes worse when a comfort range is being considered. The PID controllers are usually difficult to tune for more complex systems, which can lead to oscillations, overshoots and time-lags. Moreover because of their simple nature, both RBC and PID controllers have difficulties in handling the multiple-input multiple-output (MIMO) systems due to the coupled dynamics of the outputs. For this paper, a single RBC controller was designed based on measurements in the central room of the house. In the case of the PID, one controller was constructed for each zone individually.

3.3.1. Rule based controller

The working principle of the RBC is as follows: a temperature sensor is placed in the main room, typically the living room. Based on this temperature and a comfort band, the central heating is turned on or off. Hot water can only flow to the radiators when the central heating is on. All radiators are equipped with thermostatic valves, except those in the room of the thermostat. The valve acts as a proportional controller by controlling the water mass flow rate through the radiator and so controlling its power.

The supply temperature T_{sup} is equal for all radiators and calculated using a typical heating curve equation:

$$T_{sup} = r + \left(\frac{T_{sup,n} + T_{ret,n}}{2} - y_{j,n}\right) q^{1/m} + \frac{T_{sup,n} - T_{ret,n}}{2} q \tag{3}$$

$$q = \frac{r - (T_{e,6h} + \epsilon)}{y_{j,n} - (T_{e,n} + \epsilon)} \tag{4}$$

where *sup* and *ret* stand for supply and return water temperatures, the subscript n refers to the nominal conditions ($T_{sup,n} = 70$ °C, $T_{ret,n} = 50$ °C, $T_{e,n} = -10$ °C), and the index j refers to the room with the thermostat. The exponent m depends on the heating system (for radiator, $m = 1.3$ [71]). A realistic value of the correction term $\epsilon = 8$ K on the outside temperature $T_{e,6h}$ (averaged over 6 h) is added to take the solar gain into account.

The binary control action z_k of the central heating in *k-th* time step, based on the temperature measurement in *j-th* (central) room $y_{j,k}$ and given reference temperature r_k is defined by a switching rule of the relay based thermostat formulated by following equation

$$z_k = \begin{cases} 1 & \text{if} \quad (z_{k-1} = 1 \ \wedge \ (y_{j,k} \leqslant r_k + \gamma)) \vee \\ & \quad (z_{k-1} = 0 \ \wedge \ (y_{j,k} \leqslant r_k - \gamma)) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\wedge$ is the logic conjunction and $\vee$ denotes the logic disjunction. The parameter 2γ represents the hysteresis width. The values of the control action represent the heating mode if $z_k = 1$ and not heating if $z_k = 0$.

Finally the actual power $u_{i,k}$ delivered by the *i-th* radiator to the *i-th* zone at the *k-th* time step is given by:

$$u_{i,k} = \begin{cases} G_i z_k (T_{sup,k} - y_{i,k}), & \text{if } i = j \\ \alpha_i G_i z_k (T_{sup,k} - y_{i,k}), & \text{otherwise} \end{cases} \tag{6}$$

with $\alpha_i \in [0,1]$ the proportional gain of the thermostatic valve and G_i the total thermal conductance of the radiator. Each radiator is sized such that its maximum power is required when the outside temperature drops to −10 °C. The same power bounds are used for all controllers (RBC, PID, MPC).

3.3.2. PID controller

A set of six PID controllers is used to control the temperatures in individual zones. The corresponding parameters of PID controllers implemented in a parallel form for each zone are provided in Table 4. The computed control signals are saturated to satisfy the power bounds. In

Table 4
PID parameters for individual zones.

Zone no.	P	I	D
1.	1010.0	0.4520	379000
2.	127.5	0.1562	90800
3.	60.6	0.0402	20500
4.	371.0	0.1980	134000
5.	124.0	0.0708	44100
6.	112.0	0.0652	39500

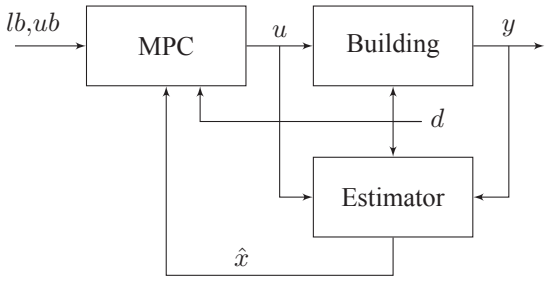


Fig. 3. Schematic representation of the closed-loop system with state estimator and MPC as a controller. Here, d are measured disturbances, y denotes the outputs, u are the control actions, $\hat{x}$ are state estimates, lb and ub are lower and upper bounds, respectively.

order to prevent the integral windup in control the integral term is bounded within the limits $[-1e3,2e3]$.

3.4. Model predictive control

Model predictive control (MPC) is a control strategy that optimizes the control actions over a finite time-horizon with respect to given objective criteria, predicted dynamic behavior of the system, system constraints and forecast of future disturbances. MPC has the ability to directly take into account given control objectives (see Section 3.2) by penalizing them in the cost function of the optimization problem. The main drawback of this strategy is the difficulty of obtaining an accurate and computationally efficient controller model for the building considered and the high computational cost (CPU) needed to solve the optimization problem.

Fig. 3 illustrates a general MPC setup used in practice. The control loop consists of the emulator model representing the real building, the state estimator and the MPC controller. The emulator model is represented by the LTI SSM obtained by linearization of the Modelica model (see Section 3.1). In order to investigate the upper MPC performance bound, in this paper the controller model equals the emulator model, i.e. no plant-model mismatch is being considered. Moreover, we consider that the perfect information about states is provided by the state estimator, i.e. $\hat{x} = x$. The building is affected by disturbances d (e.g., weather conditions), which are measured and used as perfect predictions (with zero prediction error) by the MPC. MPC optimally manipulates the control action u, which directly represents the heat flow per zone injected in the building. The heating system is thus idealized. The output vector y consists of the room operative temperatures and vectors lb and ub represent the corresponding lower and upper comfort bounds.

The MPC for minimization of the energy consumption and maximization of the thermal comfort can be cast as the following quadratic optimization problem:

$$\min_{u_0,\dots,u_{N-1}} \sum_{k=0}^{N-1} (||s_k||^2_{Q_s} + ||u_k||^2_{Q_u}) \tag{7a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k + Ed_k, \quad k \in \mathbb{N}_0^{N-1} \tag{7b}$$

$$y_k = Cx_k + Du_k, \quad k \in \mathbb{N}_0^{N-1} \tag{7c}$$

$$lb_k - s_k \leqslant y_k \leqslant ub_k + s_k, \quad k \in \mathbb{N}_0^{N-1} \tag{7d}$$

$$\underline{u} \leqslant u_k \leqslant \overline{u}, \quad k \in \mathbb{N}_0^{N-1} \tag{7e}$$

$$x_0 = x(t), \tag{7f}$$

$$d_0 = d(t). \tag{7g}$$

where $\mathbb{N}_a^b = \{a,a+1,\dots,b\}$ is a set of integers, x_k, u_k, y_k and d_k represent the values of the states, inputs, outputs and disturbances, respectively, predicted at the k-th step of the prediction horizon N. The predictions are obtained from the LTI prediction model given by Eqs. (7b) and (7c). The lb_k and ub_k parameters represent the comfort band given by the constraints (7d), where the variables s_k are used as the slack variables of a comfort band violation. The min/max constraints for the control input amplitude are given by (7e). For particular initial conditions (7f) and (7g), the optimization computes the sequence $u_0^*,\dots,u_{N-1}^*$ of control inputs that are optimal with respect to the quadratic objective function (7a) and the constraints. The term $\|a\|_Q^2$ in the objective function represents the weighted squared 2-norm, i.e., a^TQa, with the weighting matrices Q_s and Q_u given as positive definite diagonal matrices. The first term of the quadratic cost function minimizes the square of the comfort violation, while the second term minimizes the square of the used energy.

Denote by ξ the vector that encapsulates all time-varying parameters of (7), i.e. the current states $x(t)$, current and future disturbances $d(t),\dots,d(t+NT_s)$, as well as comfort boundaries signals $lb(t),\dots,lb(t+NT_s)$ and $ub(t),\dots,ub(t+NT_s)$. The receding horizon feedback law is then given by

$$u(\xi) = [\mathbf{I} \;\; \mathbf{0} \;\; \cdots \;\; \mathbf{0}]\, U_N, \tag{8}$$

where $\mathbf{I}$ and $\mathbf{0}$ represent, respectively, identity and zero matrices of appropriate dimensions. The Eq. (8) means, that for a closed-loop implementation of MPC, only the first element of U_N needs to be applied to the plant at each sampling instant. Note that the optimal open-loop sequence U_N in Eq. (8) is defined as the optimal solution of (7), formulated for a particular initial conditions given in (7f). Therefore to obtain the optimal control action for a particular value of ξ from (8), one needs to solve (7) at each sampling instant. Such a procedure, however, requires significant computational effort and must be implemented on a hardware platform that allows running optimization algorithms. This is in contrast to our objective of implementing the control strategy on very simple hardware with limited computational and memory storage resources. Therefore in the next section we show how to derive simple regression-based controllers that mimic the behavior of MPC policy (8) and can be implemented on simple hardware.

4. Approximate model predictive control

This section elaborates on the methodology for the approximation of an arbitrary controller behavior with multiple manipulated variables in the context of building climate control. Section 4.1 defines the problem of interest as a multivariate regression and discusses the computational and practical aspects of the proposed approximation method. Section 4.2 provides an overview and differentiation of the various regression algorithms and justifies the selection of the regression trees (RT) and time delay neural networks (TDNN), which are further described in Sections 4.2.1, and 4.2.2, respectively. Section 4.3 introduces a systematic feature selection approach for predictive models in the scope of building climate control applications.

4.1. Machine learning problem definition

From mathematical point of view, the parametric solution of problem (7) can be viewed as a piecewise affine function defined over polyhedral regions, mapping the parametric space to the control inputs space, i.e. $f_{\text{MPC}}: \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_u}$. However, this property holds only for linear and quadratic objective functions, respectively, subject to linear constraints [27]. In the following sections, we show how to find explicitly defined approximations for the solutions of the MPC problem with an arbitrary type of cost function and constraints by using multivariate regression.

4.1.1. MPC-like regression-based feedback law

In a regression a set of m training data[1] $\{(\xi^{(1)},u^{(1)}),\dots,(\xi^{(m)},u^{(m)})\}$ is

[1] Here, $\xi^{(i)} \in \mathbb{R}^{n_\xi}$ denotes the i-th sample of a vector ξ.

given with $\xi^{(i)} \in \mathbb{R}^{n_\xi}$ and $u^{(i)} \in \mathbb{R}^{n_u}$. The objective is to devise a regression function $f_\Theta: \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_u}$, which predicts the values of u (often called the *response* or *target variable*) that correspond to the parameters ξ (representing the *feature vector* in Machine Learning jargon) as accurately as possible.

The central idea here is to replace the implicitly defined feedback policy (8) by an explicit representation of the feedback law $u = f_\Theta(\xi)$, constructed by regression on a set of training data. The main advantage over the explicit MPC approach is that in regression we can control the complexity of $f_\Theta(\cdot)$ directly. In other words, we can design the regression-based feedback strategy while considering limitations of the control hardware. Moreover, the regression approach is not limited to lower dimensional parametric spaces as it is in the case of the explicit MPC, which allows construction of the approximated explicit control laws also for complex problems with many parameters. The implied limitation of the approach is that the regression-based control policy is suboptimal w.r.t. the solution of the MPC problem (7). Therefore, our objective in this section is to synthesize the approximated explicit MPC control law, or regressor $f_\Theta(\cdot)$ that minimizes the deterioration of the control performance w.r.t. the performance criteria presented in Section 5.1.1. The regression algorithm selection will be further described in Section 4.2.

In general, we propose to construct the regression-based feedback policy by using Algorithm 1.

Algorithm 1. Regression-based feedback policy.

1. Select the set $\{\xi^{(1)},...,\xi^{(m)}\}$ of initial values of parameters for the MPC problem (7).
2. For each $\xi^{(i)}$, obtain the corresponding optimal control action $u^{(i)}$ from (8) by solving (7).
3. Collect $\xi^{(i)}$ and $u^{(i)}$ into the training data set $\{(\xi^{(1)},u^{(1)}),...,(\xi^{(m)},u^{(m)})\}$ and apply machine learning to devise a regressor $f_\Theta(\cdot)$ that predicts the value of the control actions for an arbitrary vector of features ξ by $u = f_\Theta(\xi)$.

4.1.2. Data generation

The selection of the training datasets in the first step of the Algorithm 1 can be performed in two ways. The first option is to use a grid approach as presented in [40]. However, this approach is only applicable if the number of parameters n_ξ is low. In particular, let n_g be the number of equidistantly-placed grid points for each of the parameters ξ. Then the total number of generated points $m = n_g^{n_\xi}$. Please note that MPC problem (7), employing the model from Section 3.1, has 1518 parameters, i.e., $n_\xi = 1518$, and therefore the grid-based approach is far from practical scope in our case.

If n_ξ is large (which is usually the case in building climate control applications), an alternative way is to extract the pairs ($\xi^{(i)}$, $u^{(i)}$) from closed-loop profiles, as introduced in [42]. Here, the building is controlled according to the MPC strategy (7) for a limited amount of time. While doing so, we record the values of $\xi(t)$ and the corresponding optimal control moves $u(t)$ at a fixed sampling rate T_s. The training dataset is then composed of the tuples $(\xi^{(i\Delta)},u^{(i\Delta)})$ for $\forall\, i \in \mathbb{N}_0^m$, where Δ is the collection period and m is the number of samples to be collected. Due to the problem size the training data used in this paper were extracted from the closed-loop MPC profiles.

Remark 2. In this paper, the collection period Δ is considered to be equal to the sampling period T_s, for the sake of simplicity. In general the collection period can be an arbitrary positive integer multiple of the sampling period gives as $\Delta = \alpha T_s$, with $\alpha \in \mathbb{Z}_{>0}$. Here $\alpha > 1$ can be useful in order to reduce the computational burden in a case where big data with small sampling rate for a long time period are available.

Table 5
Comparison of the different regression algorithms. The preferred properties are highlighted in bold.

Algorithm	Function	Multivariate	Parametric	Memory	Dimensions
GLM	Linear	**Yes**	Yes	No	**High**
NLR	**Nonlinear**	No	Yes	No	**High**
RT	PW constant	No	**No**	No	**High**
SVM	**Nonlinear**	No	**No**	No	Moderate
MLP	**Nonlinear**	**Yes**	**No**	No	**High**
TDNN	**Nonlinear**	**Yes**	**No**	**Yes**	**High**
VAR	Linear	**Yes**	Yes	**Yes**	**High**

4.2. Regression algorithm selection

In general, the problems naturally arising in building control applications are those with multiple controlled and manipulated variables as opposed to single input control problems. Therefore the task of finding the approximate control law $u = f_\Theta(\xi)$ belongs to the class of multivariate regression problems, for which the objective function is given by (9).

$$\min_\Theta \sum_{j=1}^{n_u} \sum_{i=1}^{m} (f_\Theta(\xi^{(i)})_j - u_j^{(i)})^2 \tag{9}$$

The objective here is to minimize over the Θ the square of the difference between the precomputed control inputs $u_j^{(i)}$ (obtained from Eq. (8)) and regressor functions $f_\Theta(\xi^{(i)})_j$ w.r.t. the given features $\xi^{(i)}$ parametrized by Θ. The superscript i denotes the i-th sample of the training data, and subscript j stands for the j-th target variable, representing manipulated variable in the control context.

Table 5 summarizes the comparison of the seven most commonly used regression algorithms nowadays: generalized linear models (GLM), nonlinear regression (NLR), regression trees (RT), support vector machines (SVM), multilayer perceptron (MLP) neural network architecture, time delay neural networks (TDNN) and vector autoregressive model (VAR). The comparison is based on following properties: the nature of the regressor function (e.g. linear, nonlinear, piecewise constant); the ability to handle problems with more than a single target variable; the division between parametric models where the model structure needs to be selected a priori before learning and non-parametric models where the model structure is not chosen beforehand, but it is being learned instead; the memory property that indicates the ability of the model to handle the time series problems; and finally the ability of the compared models to handle high dimensional learning datasets with many samples. For the MPC problem formulations resulting in linear or quadratic optimization problems, the feedback law takes the form of a piecewise affine (PWA) function defined over a polyhedral domain of the state space. However, this property no longer holds for the MPC formulations where nonlinearity is present in the objective functions or constraints. Therefore, for the sake of versatility and flexibility of the developed methodology, we opted for the regression models with nonlinear nature of the fit. Further, we are forced to consider only the models that can handle the multivariate nature of the problem (9). The non-parametric models were chosen instead of the parametric ones because setting up the model structure before the learning process for the problems of this complexity is a very tricky task. Moreover, the abundance of the training data supports the notion of learning the structure of the model as well as the model parameters. The ability of the model to capture the dynamics of the data is also a desired property because buildings have high thermal inertia, which is optimally exploited by a predictive controller. And finally, we want to choose a regression model that can also handle large and high dimensional datasets, generated by the precomputed MPC solutions. After considering all these aspects, the natural choice for this type of problems is to use TDNN as regression model. However, also the RT are chosen for this study due to the human readable nature of the regressor

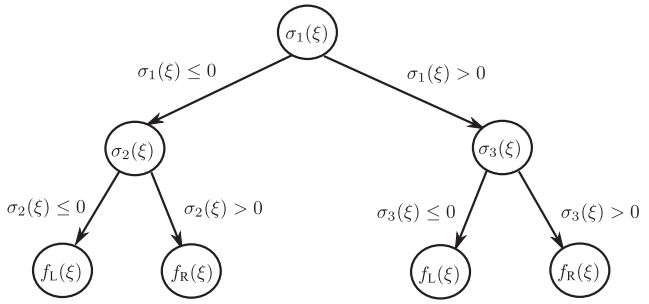


Fig. 4. Schematic representation of the regression tree.

and the ability to handle time series data after appropriate feature engineering procedures.

4.2.1. Regression trees

This section is based on the previous work of the authors as presented in [45]. A popular approach to construct the regressor functions $f_\Theta(\cdot)$ is to employ binary regression trees [72] with characteristic structure as shown in Fig. 4. Such trees consist of a finite number of nodes, each of which may contain pointers to two child nodes, *left* and *right*. Nodes without any children are called *leaf nodes*. Each leaf node contains a local expression of the regressor $f_L: \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_u}$ and $f_R: \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_u}$ for left and right child node, respectively. All non-leaf nodes, also called *tree nodes*, on the other hand, contain an expression of a splitting function $\sigma: \mathbb{R}^{n_\xi} \to \mathbb{R}$ and pointers to a maximum of two child nodes. The left child node is visited if $\sigma(\xi) \leqslant 0$, while the right node is explored if $\sigma(\xi) > 0$.

The regression tree can be constructed by a recursive procedure. In particular, we need to determine the optimal splitting function $\sigma(\cdot)$, along with optimal local regressors $f_L(\cdot)$ and $f_R(\cdot)$ that solve the following optimization problem:

$$\min_{\sigma, f_L, f_R} \left(\sum_{\xi^{(i)} \in \mathscr{P}_L} \|u^{(i)} - f_L(\xi^{(i)})\| + \sum_{\xi^{(j)} \in \mathscr{P}_R} \|u^{(j)} - f_R(\xi^{(j)})\| \right), \tag{10}$$

where

$$\mathscr{P}_L = \{\xi \mid \sigma(\xi) \leqslant 0\}, \ \mathscr{P}_R = \{\xi \mid \sigma(\xi) > 0\}, \tag{11}$$

are the cells generated by the split $\sigma(\cdot)$. Once the optimal split and the optimal local regressors are computed, we need to determine whether the currently explored node needs to be subdivided. This decision is based on two criteria: the number of points in each of the split cells, and the regression error in each cell. If the number of points in the cell (cf. (11)) drops below a pre-defined threshold $m_{\min}$, or when the local regression error is smaller than $e_{\min}$, exploration of the cell is terminated and a leaf node containing the corresponding local regressor $f_L(\cdot)$ and $f_R(\cdot)$, respectively, is returned. Otherwise, the cell is explored recursively.

Remark 3. Alternatively, the stopping criteria can be modified to stop the recursion when a maximal tree depth $D_{\max}$ is reached. Such a criterion gives the designer a direct control over tree size, and hence over implementation complexity in the control hardware.

The difficulty of constructing an optimal regression tree stems from the nonlinearity of the optimization problem (10) for general types of the split function $\sigma(\cdot)$ and of the local regressors $f_L(\cdot), f_R(\cdot)$. To simplify the computation, standard regression tree approaches restrict splits to orthogonal hyperplanes of the form $\sigma = \alpha^T \xi + \beta$, where the optimal vector α is selected from the finite set $\{[1,0,...,0]^T, [0,1,0,...,0]^T, ..., [0,...,0,1]^T\}$. Moreover, the local regressors are typically assumed to be constant functions, i.e., $f_L(\xi) = g_L$ and $f_R(\xi) = g_R$. Hence this approach generates a binary tree that encodes a piecewise constant regressor $f_\Theta(\cdot)$ defined over a rectangularly partitioned parametric space. However, such simplifications decrease the quality of the regression. As a consequence, a high number of nodes is typically required to achieve the allowed regression error. Moreover, the RTs are only single variate regression models, and due to this fact, a standalone RT needs to be constructed for each zone separately, which increases the complexity of the final controller and also causes some loss of performance by not taking into account the coupling between the controlled outputs.

4.2.2. Deep time delay neural networks

It has been shown that a neural network (NN) with a sufficient number of hidden units can approximate arbitrary continuous functions defined on a closed and bounded set [67,73]. Because of this, the NNs are popularly used as general nonlinear regression models. NNs are mathematical models inspired by the human brain, defined by the interconnections (synapses) of the individual neurons in successive layers. This can be visualized as a directed graph in Fig. 5. When no cycles are present in the graph the network is called feedforward; otherwise, it is called recurrent.

Mathematically, NN is a function $f(\cdot)$ composed of weighted sums (synapses) of bounded monotone functions $g(\cdot)$, also called neuron activation functions, which are again composite of the activation functions from the previous layers, all the way down to the input layer, which consist of the features ξ. The compact representation of the general multilayer feedforward NN, also called multilayer perceptron (MLP) is given by the set of Eqs. (12).

$$f(\xi)_i = g_{M,i}(z_M) \tag{12a}$$

$$z_M = B_M + \sum_{j=1}^{n_M} W_{M,j} g_{M-1,j}(z_{M\text{-}1}) \tag{12b}$$

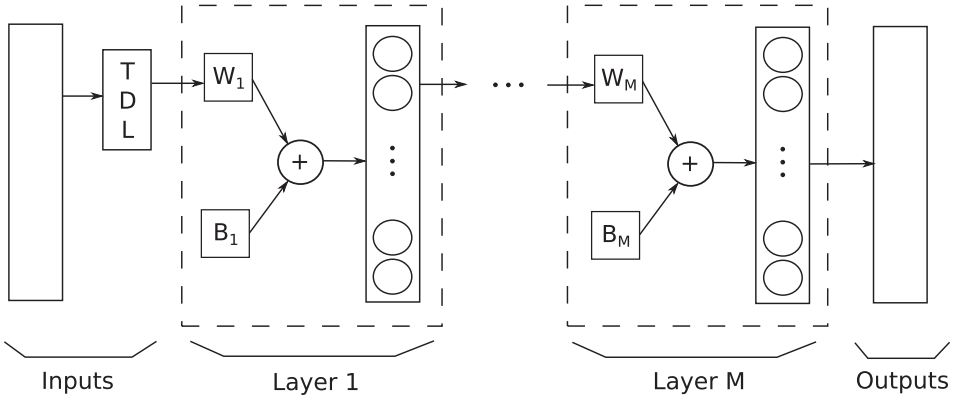


Fig. 5. Schematic representation of the time delay neural network with deep architecture. Here, TDL stands for the time delay operator, W_M and B_M are weights and biases for the M-th layer, respectively.

$$z_{M-1} = B_{M-1} + \sum_{k=1}^{n_{M-1}} W_{M-1,k} g_{M-2,k}(z_{M-2}) \tag{12c}$$

$$\vdots$$

$$z_1 = B_1 + \sum_{l=1}^{n_\xi} W_{1,l} \xi_l \tag{12d}$$

Here, the function $f(\cdot)_i$ represents the i-th output of the net, $g_{M,i}(\cdot)$ is the activation function of the i-th neuron, while z_M is the total weighted sum of the inputs in the M-th layer (i.e. the outputs from the (M−1)-th layer). ξ_l stands for the l-th feature. The number of the units (neurons) in the the M-th layer is given by n_M. $W_{M,j}$ denotes the weight of the output of the j-th neuron in the M-th layer and B_M is the bias term for the M-th layer. The weights and biases are grouped into a single vector Θ, regarded as a vector of optimized variables in the problem (9), which needs to be solved in order to train the NN.

The individual types of neural networks are being differentiated by the different setup of three parameters: first, the structure of the neural network imposed by the interconnections of the neurons, second, the activation function type that determines the output of the individual neurons and third, the weights of the interconnections. The first two parameters are to be chosen prior to the learning process, while the weights are updated during learning. In our case, we opted for the deep time delay neural network (TDNN) with sigmoid activation functions. Here, the deep architecture stands for the NN with more than a single hidden layer. The only difference between TDNN and classical feed-forward NN lies in the input layer, where the dynamics are captured by the set of delayed time signals $\{\xi^{(t)},\xi^{(t-1)},...,\xi^{(t-N)}\}$ derived from the original features ξ. The reason for using a deep architecture is a substantially better performance in comparison with the shallow nets, as shown by [74,75]. The compact schematic representation of the TDNN with deep architecture, consistent with Eqs. (12) is shown in Fig. 5.

4.3. Feature engineering

In this section, we present a simple and systematic approach for efficient feature selection (FS) for predictive models in the scope of building climate control applications. The method presented here is versatile and can be used for identification and selection of the most relevant variables in the dynamic building model, either for controller approximation, reducing the complexity of the model, or for reduction of the cost of the sensory equipment in the real building. The presented method is composed of three independent steps, first the manual selection and elimination of linearly dependent features, second, a principal component analysis (PCA) based feature selection, and third, selection of the disturbance features based on the model dynamics.

4.3.1. Manual selection and elimination of linearly dependent features

The most straightforward step is the manual FS, which exploits the engineering knowledge of the system. In contrast to the standard MPC scheme from Fig. 3, the machine learning (ML) controller from the closed-loop system representation, as shown in Fig. 6, is approximating and replacing not only the MPC behavior but also the state observer. In our case, this setup reduces the parametric space ξ by discarding the 286 state variables x and manually replacing them by the six output variables y as a part of the reduced feature vector $\tilde{\xi}$. Further, all linearly dependent features can be discarded, because they do not carry additional information for the machine learning model [76]. In our case, we consider the same comfort boundaries for all zones, and therefore the 12 original variables are reduced to a single variable, particularly the lower comfort boundary. Similarly, by elimination of the linearly dependent variables from the disturbance vector d, we reduced the set of 44 original disturbances to 26 features.

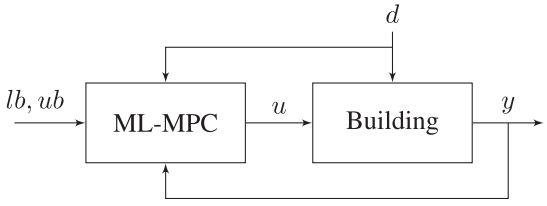


Fig. 6. Schematic representation of the closed-loop system with machine learning (ML) controller mimicking the behavior of the MPC and state observer. Here, d are measured disturbances, y denotes the outputs, u are the control actions, lb and ub are lower and upper bounds, respectively.

4.3.2. Principal component analysis feature selection

Principal component analysis (PCA) is a well-known multivariate statistical technique used mainly for dimensionality reduction of high-dimensional datasets. Because FS is essentially a dimensionality reduction problem several techniques based on PCA have been developed, see, e.g. [77–79]. In this work, we choose a simple and computationally efficient adaptation of the method designed in [80]. The procedure is defined by Algorithm 2.

Algorithm 2. PCA-based feature selection.

1. Compute the covariance matrix of the feature vector ξ, given as: $\Sigma = \frac{1}{m}\xi^T\xi$.
2. Perform singular value decomposition (SVD) of Σ, to obtain: $\Sigma = USV^T$, where U are the principal component coefficients, and S are the principal component variances, i.e. eigenvalues of Σ.
3. Compute the percentage of the total variance captured by i-th principal component, given as: $\nu_i = \frac{S_{i,i}}{tr(S)}$. ▷ Here, $tr(S)$ denotes the trace of matrix S.
4. Define the precision threshold η value for the retained variability of the data and select only the q most significant principal components, for which the total accumulative variance is within the given threshold: max$q$, s.t. $\sum_{i=1}^{q} \nu_i \leqslant \eta$.
5. Compute the normalized contribution ν_j of the j-th feature ξ_j on selected principal components from Step 4 by summation of the absolute values of the coefficients of the first q columns of matrix U: $\nu_j = \frac{\sum |U_{j;1,\dots,q}|}{\max_{k=1,\dots,n_\xi}(\sum_{}^{q}|U_{k,\dots}|)}$.
6. Define the threshold ψ value for the minimal contribution of the features on the principal coefficients and select the p most important features by satisfying: $\nu_j \geqslant \psi$, $\forall\, j \in \mathbb{N}_0^{n_\xi}$.

The method is demonstrated on the disturbance vector d with two months data. The results for the parameter values $\eta = 99.9\%$, and $\psi = 99.0\%$ are shown in Fig. 7, where Fig. 7a illustrates the PCA coefficient matrix U (Step 2), Fig. 7b shows the total variance percentage of the individual principal components (Step 3), Fig. 7c illustrates only the 9 most significant PCA coefficients of the U matrix (Step 4), and Fig. 7d shows the normalized contribution of the individual features ξ_j on selected most significant PCA coefficients (Step 5). By this method, 22 out of 44 disturbances are selected.

4.3.3. Feature selection based on model disturbances dynamics

The idea behind the feature selection via analysis of the disturbances dynamics is based on the investigation of the disturbance matrix E from LTI model (1). The method uses the importance metric Id for each disturbance given by the following equation:

$$Id_j = \max(d^j) \sum_{i=1}^{n_x} \left| E_{i,j} \right|. \tag{13}$$

The sum of the absolute values of the coefficients for the j-th column of matrix E represents the aggregated dynamic effect of the j-th disturbance on the state variables. This effect is scaled with the maximal values of the disturbances d from the dataset. Then the relative importance rId_j for the j-th disturbance is computed by dividing the metric

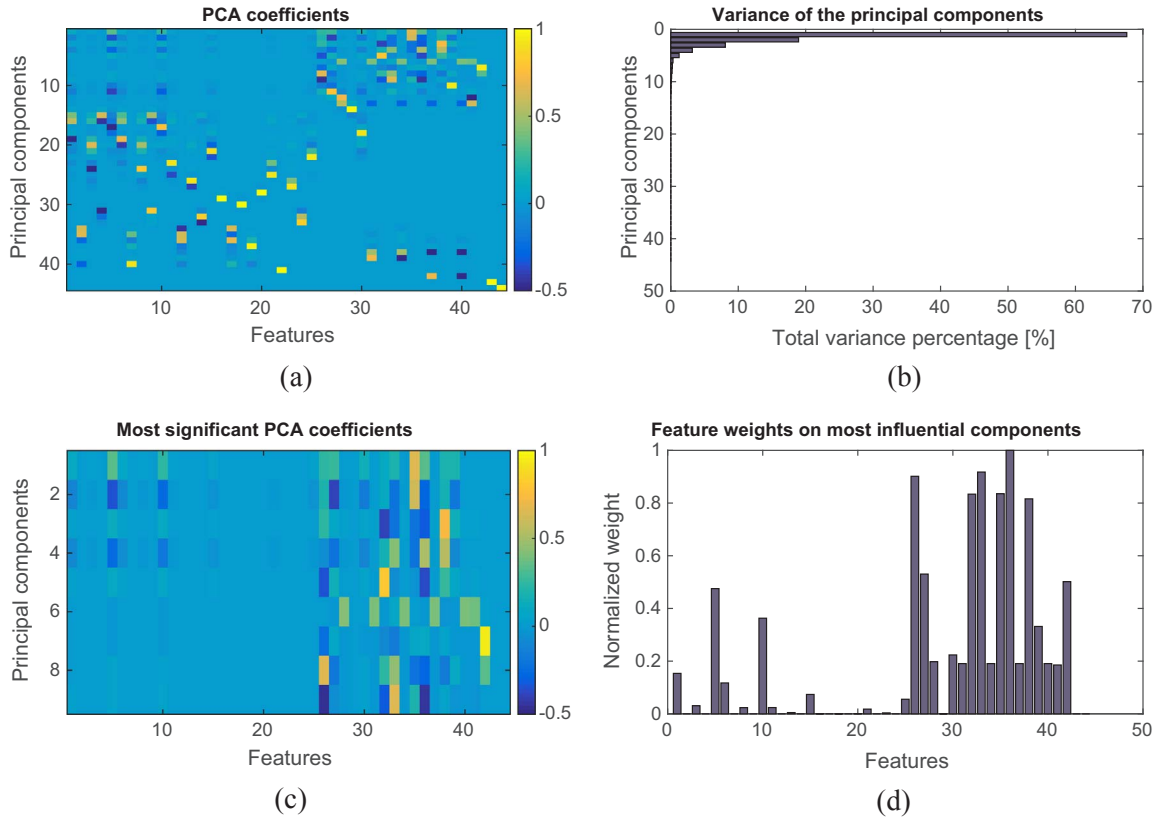
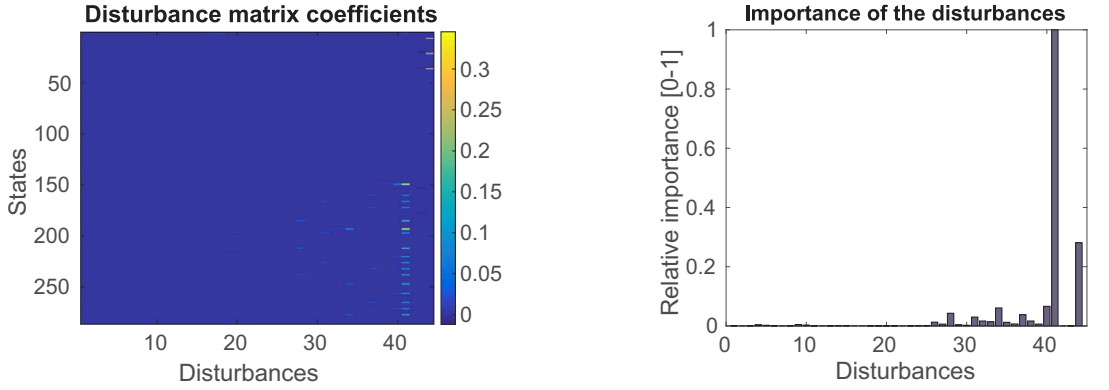


Fig. 7. Feature selection via principal component analysis.



(a) Visualization of the disturbance matrix coefficients. *(b) Relative importance of the disturbances.*

Fig. 8. Feature selection via disturbance dynamics analysis.

Id_j by the overall maximal value of metric $\max(Id_j)$,$\forall\, j \in \mathbb{N}_0^{n_d}$. Fig. 8 shows the visual representation of this metric for the particular model presented in Section 3.1. We can see from Fig. 8a, that the matrix E is very sparse. Therefore it is not surprising that the relative importance of the disturbances differs strongly as shown in Fig. 8b. The feature selection is now based on the selection of the threshold for the relative importance and elimination of all disturbances not meeting this criterion. In our case, we choose this threshold to be equal to 0.001, which in combination with the previous steps as explained in Sections 4.3.1 and 4.3.2, results in the selection of 12 out of 44 original disturbance variables.

4.3.4. Feature engineering overview

The dimensionality of the original parametric space ξ for the MPC problem (7), with prediction horizon $N = 22$, and with the model from Section 3.1, is given by the formula: $n_x + Nn_r + Nn_d = 1518$ (see Table 2). By applying the FS steps presented in Sections 4.3.1, 4.3.2 and 4.3.3, the dimensionality of the reduced parametric space $\tilde{\xi}$ is now given by the formula $n_y + Nn_{\tilde{r}} + Nn_{\tilde{d}}$ and contains only 204 selected features. Here $n_{\tilde{r}} = 1$ and $n_{\tilde{d}} = 8$ represent the selected reference and disturbance signals dimensions, respectively. Table 6 summarizes the FS procedure applied to the disturbances, where the 8 most important signals are selected by the intersection of the results from the three FS method. Note that in this particular case most of the features are discarded by the method from Section 4.3.3. This result, however, is case specific and depends on the particular building model, weather data profiles and tuning parameters used in FS methods and does not imply that the method described in Section 4.3.3 is superior to the other two.

The time delayed features are used, in general, to improve the performance of the ML models. The value of the prediction horizon $N = 22$ in the MPC setup is used as a time delay operator in the context

Table 6
Selection of the disturbances as features based on the methods from Sections 4.3.1, 4.3.2 and 4.3.3. Selected features are highlighted in bold.

Notation	Description	Units	Selected 4.3.1	Selected 4.3.2	Selected 4.3.3
d_1	Absorbed heat in layer 1 of window 1	[W]	Yes	Yes	No
d_2	Absorbed heat in layer 2 of window 1	[W]	No	No	No
d_3	Absorbed heat in layer 3 of window 1	[W]	Yes	No	No
d_4	Direct solar radiation through window 1	[W]	Yes	No	No
d_5	Diffuse solar radiation through window 1	[W]	Yes	Yes	No
d_6	Absorbed heat in layer 1 of window 2	[W]	Yes	Yes	No
d_7	Absorbed heat in layer 2 of window 2	[W]	No	No	No
d_8	Absorbed heat in layer 3 of window 2	[W]	Yes	No	No
d_9	Direct solar radiation through window 2	[W]	No	No	No
d_{10}	Diffuse solar radiation through window 2	[W]	Yes	Yes	No
d_{11}	Absorbed heat in layer 1 of window 3	[W]	Yes	No	No
d_{12}	Absorbed heat in layer 2 of window 3	[W]	No	No	No
d_{13}	Absorbed heat in layer 3 of window 3	[W]	No	No	No
d_{14}	Direct solar radiation through window 3	[W]	No	No	No
d_{15}	Diffuse solar radiation through window 3	[W]	Yes	Yes	No
d_{16}	Absorbed heat in layer 1 of window 4	[W]	No	No	No
d_{17}	Absorbed heat in layer 2 of window 4	[W]	No	No	No
d_{18}	Absorbed heat in layer 3 of window 4	[W]	No	No	No
d_{19}	Direct solar radiation through window 4	[W]	No	No	No
d_{20}	Diffuse solar radiation through window 4	[W]	No	No	No
d_{21}	Absorbed heat in layer 1 of window 5	[W]	Yes	No	No
d_{22}	Absorbed heat in layer 2 of window 5	[W]	No	No	No
d_{23}	Absorbed heat in layer 3 of window 5	[W]	No	No	No
d_{24}	Direct solar radiation through window 5	[W]	No	No	No
d_{25}	Diffuse solar radiation through window 5	[W]	Yes	Yes	No
d_{26}	Direct sun radiation on horizontal surface	[W]	**Yes**	**Yes**	**Yes**
d_{27}	Diffuse sun radiation on horizontal surface	[W]	Yes	Yes	No
d_{28}	Weighted sun radiation temperature between ground and sky temperature 1	[K]	**Yes**	**Yes**	**Yes**
d_{29}	Direct sun radiation on vertical surface with orientation 1	[W/m^2]	Yes	No	No
d_{30}	Diffuse sun radiation on vertical surface with orientation 1	[W/m^2]	Yes	Yes	No
d_{31}	Weighted sun radiation temperature between ground and sky temperature 2	[K]	**Yes**	**Yes**	**Yes**
d_{32}	Direct sun radiation on vertical surface with orientation 2	[W/m^2]	**Yes**	**Yes**	**Yes**
d_{33}	Diffuse sun radiation on vertical surface with orientation 2	[W/m^2]	**Yes**	**Yes**	**Yes**
d_{34}	Weighted sun radiation temperature between ground and sky temperature 3	[K]	No	Yes	Yes
d_{35}	Direct sun radiation on vertical surface with orientation 3	[W/m^2]	**Yes**	**Yes**	**Yes**
d_{36}	Diffuse sun radiation on vertical surface with orientation 3	[W/m^2]	Yes	Yes	No
d_{37}	Weighted sun radiation temperature between ground and sky temperature 4	[K]	No	Yes	Yes
d_{38}	Direct sun radiation on vertical surface with orientation 4	[W/m^2]	**Yes**	**Yes**	**Yes**
d_{39}	Diffuse sun radiation on vertical surface with orientation 4	[W/m^2]	Yes	Yes	No
d_{40}	Weighted sun radiation temperature between ground and sky temperature 5	[K]	No	Yes	Yes
d_{41}	Ambient temperature	[K]	**Yes**	**Yes**	**Yes**
d_{42}	Convective heat coefficient	[W/m^2]	Yes	Yes	No
d_{43}	Dummy input for constant value	[–]	No	No	No
d_{44}	Ground temperature	[K]	Yes	No	Yes

of the TDNN, generating 22 shifted signals, which correspond to the predicted values of the comfort boundary *lb* and disturbance *d* signals, respectively. Further, due to the effect of the building mass, also the past values of the output variables *y* are used, which allows preserving more of the process memory and hence improving performance. The dimensionality of the reduced parametric space $\widetilde{\xi}$ for the TDNN is now given as $22(n_y + n_{\tilde{r}} + n_{\tilde{d}}) = 330$. However, the feature engineering for RT differs due to the different nature of the model. Here the best performance was achieved by using only two shifted signals for the predicted values of the comfort boundary and disturbance signals, respectively. The delayed signals of the output variables are not used in this case, because they did not improve the performance of the RT. Instead, linear time is transformed into three sinusoidal signals with different frequencies corresponding to days, weeks and months. The dimensionality of the reduced parametric space $\widetilde{\xi}$ for the RT is now given as $n_y + 2(n_{\tilde{r}} + n_{\tilde{d}}) + n_t = 27$. Table 7 gives an overview of the control and machine learning setup with overall dimensions of the selected and transformed features is given.

5. Simulation case study

In this section we present the simulation results and performance

Table 7
Machine learning setup and feature selection overview.

Notation	Control setup	ML setup	RT dimensions	TDNN dimensions
ξ	MPC parameters	All original features	1518	1518
$\widetilde{\xi}$	Subset of MPC parameters	All selected/ transformed features	27	330
y	Outputs	Selected features	6	6
lb	Lower comfort bounds	Selected features	1	1
d	Disturbances	Selected features	8	8
t	Time	Transformed features	3	–
N	Prediction horizon	Number of time delays	2	22
u	Control inputs	Targets	1	6

comparison of the investigated controllers. The performance of the approximated MPC strategies via machine learning algorithms (see Section 4) are compared with performance bound MPC (see Section 3.4), and benchmark control strategies: RBC and PID (see Section 3.3). Section 5.1 elaborates on the case study, particularly Section 5.1.1 describes the performance criteria for evaluation of the controller's performance. The simulation parameters and tuning of the controllers are presented in Section 5.1.2 and 5.1.3, respectively. Section 5.1.4 describes the training and tuning of the machine learning algorithms. Section 5.1.5 discusses hardware requirements of the developed machine learning-based controllers. Section 5.1.6 introduces simple heuristic rules for enhancement of the TDNNs performance. Finally, Section 5.2 presents and discusses the results.

5.1. Simulation setup

This section discusses the performance criteria, setup of the simulation parameters, datasets divisions, tuning of the classical controllers and MPC, and training and tuning of the approximated machine learning controllers.

5.1.1. Performance criteria

The control performance is evaluated using three performance keys: energy use, thermal comfort/discomfort level and CPU time. The energy used corresponds to the heat delivered by the radiators and is expressed in kW h. The thermal comfort/discomfort is defined by three different measures. First, thermal comfort κ is defined as the number of sampling instants in which one of the zone temperatures y_i falls inside the relaxed comfort bounds $[lb_k-\tau, ub_k+\tau]$, divided by the simulation length N_{sim} and the number of zones N_{rooms}:

$$\kappa = \frac{\sum_{i}^{N_{\text{rooms}}} \sum_{k}^{N_{\text{sim}}} \epsilon_{i,k}}{N_{\text{rooms}} N_{\text{sim}}} \times 100\% \tag{14a}$$

$$\epsilon_{i,k} = \begin{cases} 1 & \text{if} \quad lb_k-\tau \leqslant y_{i,k} \leqslant ub_k+\tau \\ 0 & \text{otherwise} \end{cases} \tag{14b}$$

with a violation tolerance $\tau = 0.1$ K. Second, thermal discomfort is evaluated as the number of Kelvin hours when the zone temperatures y_i are outside the comfort range (2), computed as a sum of magnitudes of each violation multiplied by its duration. The discomfort is further divided by the number of zones to represent a single measure for a whole building. And third, we use the Predicted Mean Vote (PMV) index as a more complex thermal comfort measure [81]. The PMV value is dimensionless, and when equal to zero it corresponds to most comfortable conditions. Here we evaluate the violations of the PMV time-varying bounds defined by the ISO 7730 standard [82]). During the day PMV is to be kept within $\pm$ 0.5; otherwise, the night setback $\pm$ 1.0 is being used. The details on the particular implementation of the PMV index and values of the corresponding parameters used in this study were adopted from previous work of the authors [83]. The CPU time corresponds to the average time needed for on-line computation of the control law.

5.1.2. Simulation parameters and datasets division

The overall simulation time was 90 days (8640 data points) that corresponds to three months (January to March) of the winter period. The control profiles generated by MPC in the first 60 days (5760 data points) were used as a training set for the machine learning models, while the last 30 days (2880 data points) acted as a test set for evaluation and comparison of the designed controller. The sampling period was chosen to be $T_s = 900$ s. The maximum heating power of the radiators $\bar{u}$ was selected such that the maximum comfort could be achieved for all considered disturbances and was equal to $[1680 \quad 685 \quad 154 \quad 1000 \quad 320 \quad 232]^T$ W. The initial state values $x(0)$ were equal to 20 °C. The disturbance vector d was generated from a typical year in Uccle, Belgium [70]. All simulations were performed on a 2.8 GHz machine with two CPU units each with six cores, under a GNU/Linux 64-bit Debian 3.16.7 operating system.

5.1.3. Tuning of RBC, PID and MPC

To improve thermal comfort satisfaction of the benchmark controllers (RBC and PID) and as such ensuring a fair comparison with MPC and neural network controllers, the reference temperatures r_k are shifted slightly above the lower boundary of the comfort band lb_k. For the RBC controller: $r_k = lb_k + 2.5$ °C, while the width of the switching zone is equal to 0.5 °C. For the PID controller: $r_k = lb_k + 1$ °C. On top of that, the positive change in the reference signal is shifted by one hour (4 sampling instants) before the positive change of the lower comfort boundary lb occurs. This is done to increase the comfort satisfaction for the RBC and PID controllers. For the purpose of PID design, a decoupling of the MIMO SSM into the set of six SISO models is done a priori. The derivation of the PID parameters is performed using MATLAB command `pidtune` with a posteriori manual tuning for enhancing the performance. Both RBC and PID are well tuned with respect to given performance criteria to provide a fair comparison with MPC.

In the case of MPC, the values of the prediction horizon N and the weighting factor $\frac{Q_s}{Q_u}$ are chosen with an emphasis on thermal comfort satisfaction with values $N = 22$ steps (i.e., 5.5 h), and $\frac{Q_s}{Q_u} = 10^8$. The particular choice of N and $\frac{Q_s}{Q_u}$ were determined by the sensitivity analysis performed in previous work of the authors [68] using the same controller model and MPC formulation. We assume here that MPC has full disturbances preview; hence the perfect weather predictions are provided. The comfort band is defined by two time-varying parameters, lb_k and ub_k, representing the lower and upper bound, respectively as defined in Section 3.2 based on ISO-7730. The MPC is constructed in the MATLAB environment, using the modeling and optimization toolbox YALMIP [84]. The closed-loop simulation is performed by applying the optimal control inputs $u^\star(t)$, computed at each sampling instant T_s by MPC to the building model. The objective function (Eq. (7a)) is quadratic, and all constraints are linear; therefore the problem (7) can be solved as a convex quadratic program (QP). In this study, we use the state of the art optimization solver GUROBI [85].

5.1.4. Machine learning models training and tuning

In the case of TDNN, all features and targets are normalised to fall in the range $[-1,1]$. The scaled conjugated gradient is chosen as the fastest training algorithm based on comparison of algorithms on similar dataset types provided by MathWorks.[2] To further accelerate the training

[2] https://www.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html.

of the NN, we use the hyperbolic tangent sigmoid activation function for all hidden neurons. Linear functions define the output layer. The initial training performance measure for TDNN is a mean squared error (MSE), however, from the control point of view, the actual performance measure is given by the criteria from Section 5.1.1. Because the maximum number of iterations is 4000, the evaluation of the closed loop simulation performance of TDNN in each iteration would dramatically slow down the learning process. The simulation performance is therefore evaluated only each 100 iterations as a stopping criterion for the training. This modification in the learning process is done mainly because the MSE of the trained models is not strongly correlated with their control performance. In fact, the NN with lower MSE may have lower control performance due to overfitting. To reduce overfitting and to increase the generalisation of the NN, we retrain each net several times with randomly initialized weights Θ. Another simple method for improving the generalisation of the NN, inspired by Occams razor, is to use only the NN that is just large enough to provide an adequate fit [86]. By training and evaluation of the different architectures multiple times, we choose the simplest best performing network structure with two hidden layers with 24 and 12 neurons, respectively. For the recall, our TDNN has 330 features in the input layer and six-dimensional output layer. This accounts for a total number of 8280 weight parameters Θ, which are represented as floating-point numbers.

The advantage of the RT is that no pre-processing of the data is necessary. A single RT is constructed for each building zone via MATLAB's `fitrtree` function. The MSE is used as a training performance measure as well as a splitting criterion for the RT. The stopping criteria are given as follows: the minimum number of branch node observations $m_{\min} = 10$, the quadratic error tolerance per node $e_{\min} = 10^{-6}$, the maximal number of decision splits (maximal tree depth) $D_{\max} = m-1$. These values are chosen in order to grow deep trees, trading the better regression performance for increased complexity. The final set of the six trees, consist of 1567,1723,1385,1397,1391 and 1139 nodes, respectively. The number of terminal nodes for each tree is 784,862,693,699,696 and 570, respectively. In total, the combined number of all nodes for six RTs is 8602, out of which 4298 are splitting nodes and 4304 are terminal nodes. For the binary regression tree that employs 27 features, the single splitting node requires storing 28 floating-point numbers (the vectors $\alpha \in \mathbb{R}^{27}$ and the scalar β for each split), while storing one terminal node requires just a single floating-point number.

Both, TDNN and RT, are designed and trained by using built-in functions from MATLAB's Statistics and Machine Learning Toolbox™. The satisfaction of the input constraints is achieved by post-processing saturation of the control laws.

5.1.5. Hardware demands of machine learning models

Assuming that 4 bytes are needed to store a single-precision floating-point number, the memory footprint of the TDNN parameters used in this study is 33.1 kilobytes, while the total memory footprint of the six RTs used in this study is 481.4 kilobytes, which is roughly fifteen times more than in the case of TDNN. The summarizing theoretical complexity comparison of TDNN and RT is shown in Table 8. The proposed neural network-based control strategy can be implemented on a "simple" hardware as long as it can process compiled C-code. Specifically, we have exported the trained network to ANSI C code using MATLAB's codegen toolbox. Such code can then be compiled for various low-cost hardware platforms. For the network of this paper, the code size was just 110 kilobytes, which enables its implementation on, e.g., Arduino ATmega2560 microcontroller with 256 kB of RAM, or using the Raspberry Pi boards. For demonstration, we have implemented the proposed TDNN controller on a Raspberry Pi 2 Model B minicomputer (900 MHz CPU, 1 GB of RAM) in a hardware-in-the-loop simulation setup. It took under 1 ms for the Raspberry Pi to evaluate the network for given inputs at each sampling instant. This shows that the proposed control strategy indeed has modest implementation requirements.

Table 8
Complexity comparison of the machine learning models.

Method	# nodes/parameters [–]	# floating-point numbers [–]	Memory footprint [kB]
RT 4.2.1	8602	124648	481.4
TDNN 4.2.2	8280	8280	33.1

5.1.6. Heuristic rules for post processing of TDNN control laws

To enhance the performance of TDNN, we introduce simple heuristic rules defined by Algorithm 3. The rules can be used for post-processing of the generated control actions u for each zone, adjusting the room temperatures y to be closer to the lower comfort boundary lb. This behavior is achieved by slightly decreasing the control action when the temperature is too high, and slightly increasing the control action when the temperature is about to violate the comfort boundary lb. The tuning parameters of the algorithm, together with their values for our particular case are as follows: lower threshold $thr_{low} = 0.1$, upper threshold $thr_{up} = 0.4$ and corresponding lower $G_{low} = 0.1$ and upper $G_{up} = 0.05$ correction gain, respectively. Because the rules are intuitive and easily tunable in the field, they increase the adaptivity of the proposed approach also for various cases with some degree of uncertainty. Please note that in our experience Algorithm 3 did not improved the performance of RT. Hence it is applied only for TDNN.

Algorithm 3. Enhancement of the time delay neural network control laws.

function CtrlEnhance$u$ ▷ given TDNN control action u
  if day **then** ▷ apply this procedure only during the daytime
    if $y < lb + thr_{low}$ **then** ▷ room temperature too close to lower comfort boundary
      $u \leftarrow u + G_{low}\overline{u}$ ▷ increase control action
      return u ▷ obtain enhanced control action
    end if
    if $y > lb + thr_{up}$ ▷ room temperature too far from lower comfort boundary
      $u \leftarrow u - G_{up}\overline{u}$ ▷ decrease control action
      return u ▷ obtain enhanced control action
    end if
  end if
end function

5.2. Results

In this section the results of the simulation case study for heating control of a 6-zones house are presented. To validate the performance of the proposed approximated MPC strategies we have performed closed-loop simulations under the same conditions for all investigated controllers. Section 5.2.1 shows and discusses the dynamic behavior of the investigated controllers, while Section 5.2.2 summarizes and compares the control performances calculated in Section 5.1.1.

5.2.1. Control profiles

To demonstrate the dynamic behavior of the building controlled via different controllers and to verify the tuning, we provide the closed-loop profiles over a representative time window of six days chosen from the test set. For the sake of clarity, we decide to show the temperature profiles only for the second zone of the building. The other zones show similar behavior. Fig. 9 gives the corresponding disturbance profiles.

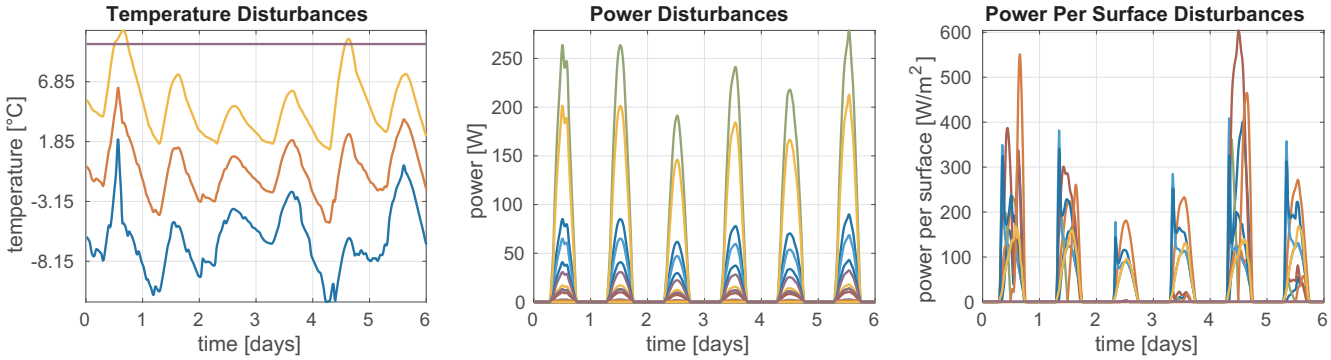
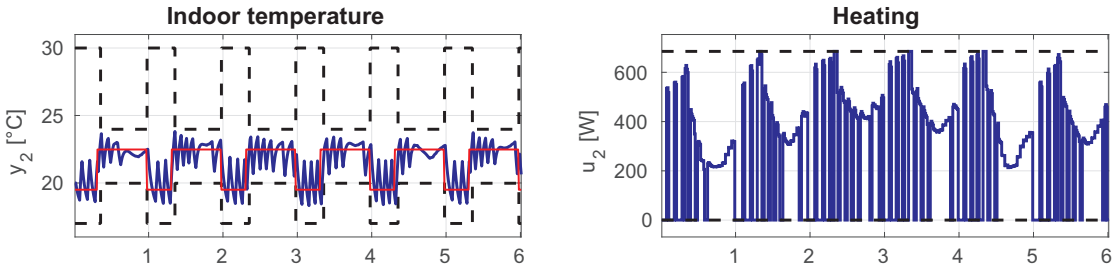
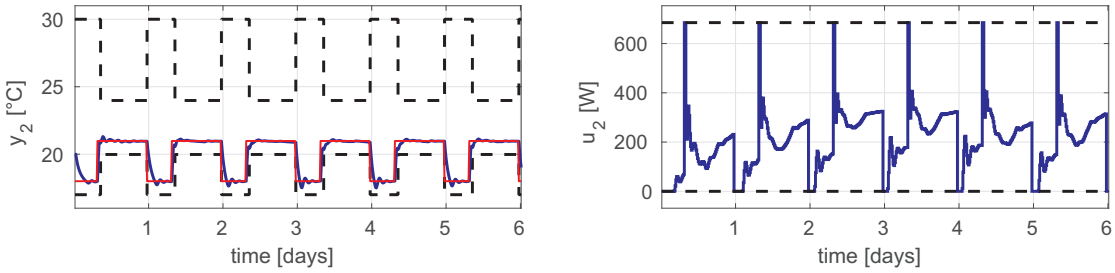


Fig. 9. 6-days test set disturbance profiles. Left: temperature disturbances, ambient temperature (yellow), ground temperature (purple) and radiation temperatures (red and blue). Middle: solar radiation through and absorbed by each window. Right: solar radiation per surface orientation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Control profiles of the RBC.

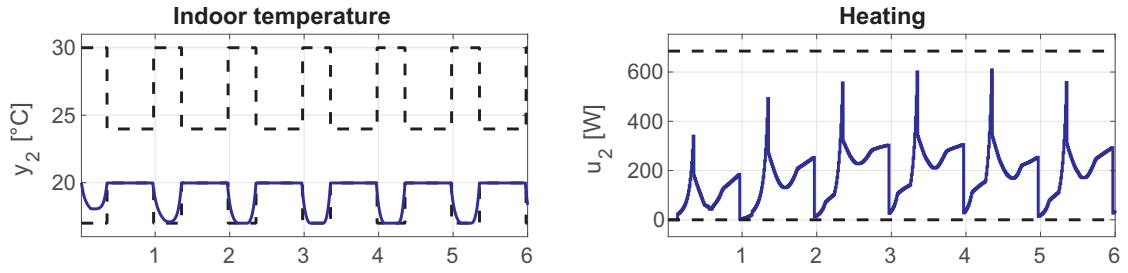


(b) Control profiles of the PID.

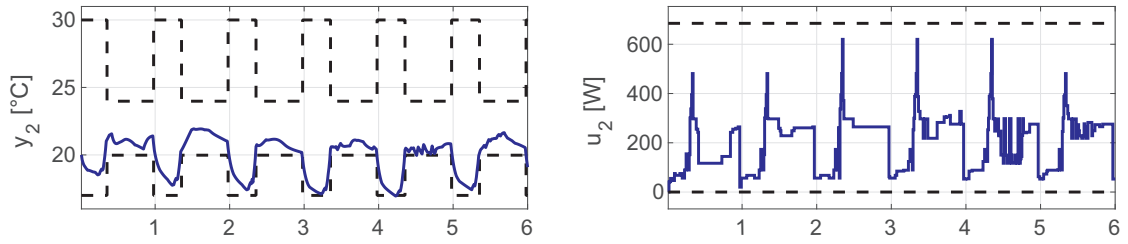
Fig. 10. Single zone 6-days test set control profiles of the conventional controllers. Left column: closed-loop response of the indoor temperature (blue) in the second building zone w.r.t. the reference (red) and the comfort constraints (black). Right column: corresponding profile of the control action (blue) w.r.t. the control boundaries (black). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The RBC profiles with typical switching behavior of the control action and oscillatory indoor temperature profiles are shown in Fig. 10a. Conservative reference setting in the middle of the comfort bounds achieves the highest comfort satisfaction. The control profiles of the well-tuned PID controller with reference tracking behavior are shown in Fig. 10b. Again we opt here for the reference setting slightly above the lower comfort boundary in order to achieve the highest possible comfort satisfaction.

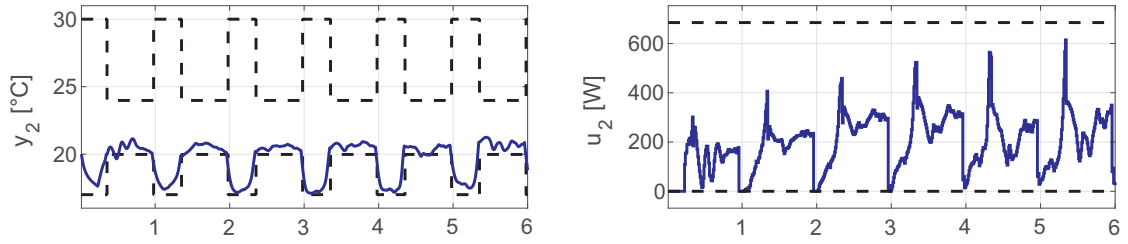
The control profiles of the performance bound (PB) MPC are presented in Fig. 11a. Here we observe the optimal performance of MPC minimizing the energy used by keeping the zone temperature as close as possible to the lower comfort boundary while taking the full disturbances preview into account. The control profiles of the supervised machine learning controllers trained on two months data approximating MPC actions acting as a 'teacher' are shown in Figs. 11d. First, regression trees (RT) behavior is presented in Fig. 11b. The RT are roughly mimicking MPC and can maintain high comfort standard with some energy saving potential compared to the RBC. However, in overall, they fail to generalise the learned MPC behavior well on the new test set data. The main limitation of the RT is the piecewise constant nature of their approximation, which for the problems of this complexity generates excessively deep trees with too many branches and nodes that are hard to tune. The second limiting factor of the RT is their single variate nature. Because of that, a single RT controller needs to be constructed for each zone separately, not allowing to capture the coupling between the individual zones. Not taking into account the heat gains from adjacent zones causes the temperature increase in some zones, which leads to additional increase in energy use by the RT controllers. On the other hand, the time delayed neural networks (TDNN) can mimic the MPC actions for all controlled zones with limited complexity and good generalisation using new data. The control profiles of the TDNN are shown in Fig. 11c. However, as the TDNN tries to mimic the optimal MPC behavior by keeping the zones temperatures closer to the lower comfort boundary, small violations of the comfort bounds occur due to the approximation errors. In order to improve the performance of the TDNN, simple heuristic rules introduced in Section 5.1.6 were implemented, slightly adjusting the control actions generated by the TDNN. As shown in Fig. 11d, the enhanced TDNN with



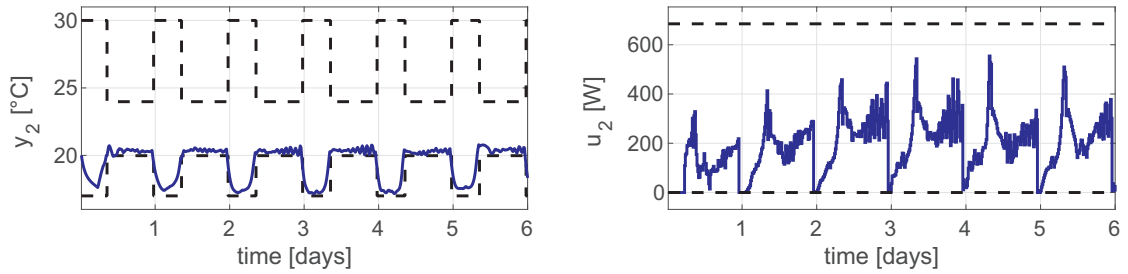
(a) Control profiles of the MPC.



(b) Control profiles of the RT.



(c) Control profiles of the TDNN.



(d) Control profiles of the TDNN enhanced with heuristics.

Fig. 11. Single zone 6-days test set control profiles of the MPC-based controllers. Left column: closed-loop response of the indoor temperature (blue) in the second building zone w.r.t. the comfort constraints (black). Right column: corresponding profile of the control action (blue) w.r.t. the control boundaries (black). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

heuristics are keeping the room temperatures closer to the lower boundaries with only occasional violations, which results in the improved performance for both, thermal comfort as well as energy use.

5.2.2. Performance comparison

The overall simulation performance of the investigated controllers on a 30-days test set based on the performance criteria defined in Section 5.1.1 is compared in Table 9. The comparisons w.r.t. the energy use, and comfort zone satisfaction, total discomfort per zone, and cumulative PMV violations, respectively, are also compactly represented in Fig. 12.

All designed controllers can achieve high comfort satisfaction performance; this comes however with different costs on energy use. With regards to the energy savings, the RBC is taken here as a benchmark controller, then the PID energy savings potential is equal to 7.62%, while the original MPC acting as a performance bound (PB) scores close to 16%. Even though, the RT-based control policy is able to decrease the energy use of the RBC by 4%, it scores worse than well-tuned PID, which makes RT less attractive in practice due to longer development time and tuning effort in comparison with the PID. Moreover, the evaluation time of RT is almost twice as long as for the TDNN. The TDNN is outperforming both RBC and PIDs in energy use; w.r.t RBC by 11.6% and 12.8%, w.r.t PID by 4.0% and 5.2%, before and after enhancement of the control laws, respectively. In comparison with the PB MPC, the energy

Table 9
30-days test set performance comparison.

Method	Defined in section	Comfort [%]	Discomfort [Kh]	PMV viol [–]	Energy savings [%]	Evaluation time [ms]
RBC	3.3.1	98.15	33.64	0.15	–	0.13
PID	3.3	100.00	0.00	0.00	7.62	0.14
MPC (Gurobi)	3.4	100.00	0.00	0.00	15.89	15.19
RT	4.2.1	98.23	36.78	0.02	4.10	3.70
TDNN	4.2.2	95.14	87.83	1.19	11.64	2.01
TDNN + Heuristic	5.1.6	97.54	33.41	0.38	12.82	2.13

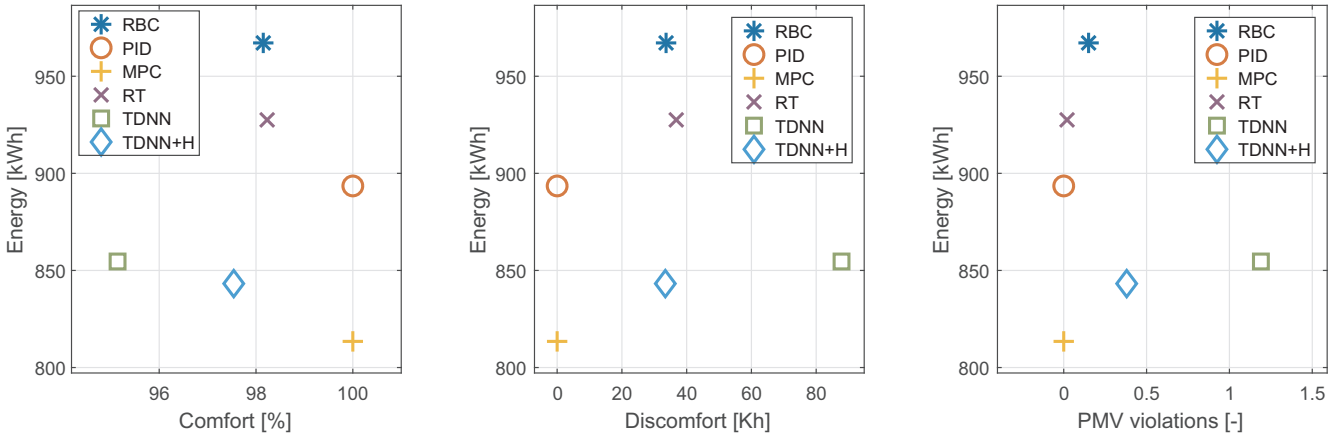
savings of the TDNN drop only by 3%, this accounts roughly for 80.7% restoration of the optimal MPC performance, with only a small increase in thermal discomfort equal to 2.5%. The main difference between machine learning controllers and MPC feedback, though, is in the implementation cost. While the MPC policy requires solving the optimization problem (7) at each sampling instant, the TDNN-based controller only needs to evaluate the function formulated in (12). Hence, the implementation of TDNN is reducing the computational effort imposed by the MPC about by the factor of 7. Advanced software libraries are no longer required, and the representation of TDNN can be easily implemented on low-level hardware with low memory footprint, which makes this approach applicable on typical building automation hardware used nowadays, such as on programmable logic controllers. Moreover, the TDNN-based controllers trained on a limited dataset of two months showed good generalization capabilities with respect to new data. The only requirement for a good generalization is that the new measurements have the same probability distribution as the training data. This fact provides a valuable clue for selection of the representative training data set, which can be chosen based on the historical weather profiles for a particular location and season. Selection of the most relevant training data significantly reduces the training time of the TDNN, because the complete coverage of all operating conditions would be practically infeasible due to the high dimensionality of the parametric space.

6. Conclusions

In this paper, we have shown how to design well-performing approximations of optimization-based controllers applicable on embedded hardware by using advanced machine learning algorithms. The focus of this paper was on complex multi-zones building control problems, employing multiple continuous control inputs. The method presented in this work is based on two multivariate regression algorithms, namely deep time delay neural networks (TDNN) and regression trees (RT). The regression is used to mimic the complex behavior, in our case the model predictive controller (MPC), based on the simulation data. A simple tunable feature selection (FS) method was further used in order to decrease the dimensionality of the parametric space, hence reducing the complexity of the devised controller and reducing the implementation cost of the sensory equipment. This selection was based on prior engineering knowledge of the system, principal component analysis (PCA) and dynamic analysis of the building's model.

To the authors best knowledge, this is the first time that deep TDNN are used to mimic the behavior of MPC in the context of building control. The approach was demonstrated on a case study for temperature control of a six-zones building, described by a linear model with 286 states, 42 disturbances, and 6 manipulated variables. In comparison with related works on this topic the dimensionality of the used model in this study is several factors higher, proving the scalability and computational efficiency of the proposed method. The results from a three months simulation showed that the TDNN-based controllers were able to maintain high comfort and energy savings, with just a small loss of performance compared to the original MPC, while drastically reducing the complexity of the solution. The computational efficiency of the neural network-based controllers was further demonstrated by hardware in the loop simulation tests using Raspberry Pi embedded platform. Moreover, the TDNN controllers showed good generalization capabilities with respect to new data and proved to be robust to weather variations within the training season. To further enhance the TDNN controllers, we implemented simple heuristic rules that have improved the performance in both terms, namely in comfort as well as energy use. The set of RT, however, scored worse, even though they outperformed the traditional RBC, their performance was worse than a well-tuned PID controller. Moreover, because the deep branching of the RT, the memory footprint of the RTs was roughly fifteen times higher compared to the TDNN and thus did not prove to be an efficient model to employ in practice.

In summary, we demonstrated a semi-automatic procedure for the synthesis of near-optimal MPC-like controllers with low complexity. The advantage of this approach is that the devised controllers are



(a) Comfort zone satisfaction via eq. (14). *(b) Total discomfort per zone.* *(c) Cumulative PMV violations.*

Fig. 12. 30-days test set performance comparison.

readily applicable to low-level hardware to control the building in real time, without the need for advanced software libraries. The methodology presented in this paper is versatile and applicable to any building controlled by an arbitrary optimization-based control algorithm. All the aspects mentioned above makes the use of this type of control strategies very promising for practical applications.

Acknowledgement

The Authors gratefully acknowledge the contribution of the Slovak Research and Development Agency under the project APVV-15-0007, and the financial support of the Scientific Grant Agency of the Slovak Republic under the grants 1/0403/15 and 1/0112/16. The authors would like to thank financial contribution from the STU Grant scheme for Support of Young Researchers. The authors also acknowledge the financial support by the European Union through funding the research work of Jan Drgona in the EU-H2020-GEOTCH project Geothermal Technology for conomic Cooling and Heating. The authors also acknowledge the financial support by IWT and WTCB in the frame of the IWT-VIS Traject SMART GEOTHERM focusing on integration of thermal energy storage and thermal inertia in geothermal concepts for smart heating and cooling of (medium) large buildings. The modeling work of Arnout Aertgeerts in the frame of the IWT Proeftuin De Schipjes (Integration of sustainable buildings and energy optimization in historical residential clusters) is acknowledged as a good starting point of the models used in this study. Also, the valuable insights into machine learning algorithms shared by Peter Laurinec [3] are being acknowledged.

References

[1] Parry M, Canziani O, Palutikof J, van der Linden P, Hanson C. Climate change 2007: impacts, adaptation and vulnerability. Intergovernmental Panel on Climate Change; 2007.
[2] Roth KW, Westphalen D, Dieckmann J, Hamilton SD, Goetzler W. Energy consumption characteristics of commercial building HVAC systems - Volume III: Energy savings potential. Technical report; 2002.
[3] Gyalistras D, Gwerder M, Schildbach F, Jones CN, Morari M, Lehmann B, et al. Analysis of energy savings potentials for integrated room automation. In: Clima - RHEVA World Congress, Antalya, Turkey; May 2010.
[4] del Mar Castilla M, Álvarez JD, de A Rodriguez F, Berenguel M. Comfort control in buildings. London: Springer-Verlag; 2014.
[5] Mechri Houcem Eddine, Capozzoli Alfonso, Corrado Vincenzo. Use of the ANOVA approach for sensitive building energy design. Appl Energy 2010;87(10):3073–83.
[6] Aghemo C, Virgone J, Fracastoro GV, Pellegrino A, Blaso L, Savoyat J, et al. Management and monitoring of public buildings through ICT based systems: control rules for energy saving with lighting and HVAC services. Front Architect Res 2013;2(2):147–61.
[7] Maciejowski JM. Predictive control with constraints. Prentice Hall; 2002.
[8] Lešić V, Martinević A, Vašak M. Modular energy cost optimization for buildings with integrated microgrid. Appl Energy 2017;197:14–28.
[9] Razmara M, Maasoumy M, Shahbakhti M, Robinett RD. Optimal exergy control of building HVAC system. Appl Energy 2015;156:555–65.
[10] Candanedo JA, Dehkordi VR, Stylianou M. Model-based predictive control of an ice storage device in a building cooling system. Appl Energy 2013;111:1032–45.
[11] Široký J, Oldewurtel F, Cigler J, Prívara S. Experimental analysis of model predictive control for an energy efficient building heating system. Appl Energy 2011;88(9):3079–87.
[12] Oldewurtel F, Parisio A, Jones CN, Gyalistras D, Gwerder M, Stauch V, et al. Use of model predictive control and weather forecasts for energy efficient building climate control. Energy Build 2012;45:15–27.
[13] Fiorentini Massimo, Wall Josh, Ma Zhenjun, Braslavsky Julio H, Cooper Paul. Hybrid model predictive control of a residential HVAC system with on-site thermal energy generation and storage. Appl Energy 2017;187:465–79.
[14] Chen Xiao, Wang Qian, Srebric Jelena. Occupant feedback based model predictive control for thermal comfort and energy optimization: a chamber experimental evaluation. Appl Energy 2016;164:341–51.
[15] Wanjiru Evan M, Zhang Lijun, Xia Xiaohua. Model predictive control strategy of energy-water management in urban households. Appl Energy 2016;179:821–31.
[16] Wanjiru Evan M, Sichilalu Sam M, Xia Xiaohua. Model predictive control of heat pump water heater-instantaneous shower powered with integrated renewable-grid energy systems. Appl Energy 2017;204:1333–46.
[17] Bianchini Gianni, Casini Marco, Vicino Antonio, Zarrilli Donato. Demand-response in building heating systems: a Model Predictive Control approach. Appl Energy 2016;168:159–70.
[18] Razmara M, Bharati GR, Hanover Drew, Shahbakhti M, Paudyal S, Robinett RD. Building-to-grid predictive power flow control for demand response and demand flexibility programs. Appl Energy 2017;203:128–41.
[19] Cigler J, Gyalistras D, Široký J, Tiet V, Ferkl L. Beyond theory: the challenge of implementing model predictive control in buildings. In: Proceedings of 11th Rehva World Congress, Clima; 2013.
[20] Li Xiwang, Wen Jin. Review of building energy modeling for control and operation. Renew Sust Energy Rev 2014;37:517–37.
[21] Dibowski H, Holub O, Rojíček J. Ontology-based automatic setup of virtual sensors in building automation systems. In: 2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT); 2016. p. 375–81.
[22] Ferrari R, Dibowski H, Baldi S. A message passing algorithm for automatic synthesis of probabilistic fault detectors from building automation ontologies. IFAC-PapersOnLine 2017;50(1):4184–90. 20th IFAC world congress.
[23] Domahidi A, Ullmann F, Morari M, Jones CN. Learning decision rules for energy efficient building control. J Process Control 2014;24(6):763–72 Energy Efficient Buildings Special Issue.
[24] Ma Y, Anderson G, Borrelli F. A distributed predictive control approach to building temperature regulation. American Control Conference (ACC), 2011. IEEE; 2011. p. 2089–94.
[25] Ma Y, Vichik S, Borrelli F. Fast stochastic MPC with optimal risk allocation applied to building control systems. 2012 IEEE 51st Annual Conference on Decision and Control (CDC). IEEE; 2012. p. 7559–64.
[26] Huyck B, Ferreau H, Diehl M, De Brabanter J, Van Impe J, De Moor B, et al. Towards online model predictive control on a programmable logic controller: practical considerations. Math Probl Eng 2012;2012.
[27] Bemporad A, Morari M, Dua V, Pistikopoulos EN. The explicit linear quadratic regulator for constrained systems. Automatica 2002;38(1):3–20.
[28] Borrelli F. Constrained optimal control of linear and hybrid systems vol. 290. Springer-Verlag; 2003.
[29] Kvasnica M, Rauová I, Fikar M. Automatic code generation for real-time implementation of model predictive control. In: Proceedings of the 2010 IEEE international symposium on computer-aided control system design; 2010. p. 993–8.
[30] Drgoňa J, Kvasnica M, Klaučo M, Fikar M. Explicit stochastic MPC approach to building temperature control. In: IEEE conference on decision and control, Florence, Italy; 2013. p. 6440–5.
[31] Alessio A, Bemporad A. A survey on explicit model predictive control. Nonlinear model predictive control: towards new challenging applications. Berlin Heidelberg: Springer; 2009. p. 345–69.
[32] Bemporad A, Filippi C. Suboptimal explicit mpc via approximate multiparametric quadratic programming. In: Proceedings of the 40th IEEE conference on decision and control (Cat. No.01CH37228), vol. 5; 2001. p. 4851–6.
[33] Grieder P, Morari M. Complexity reduction of receding horizon control. In: 42nd IEEE international conference on decision and control (IEEE Cat. No.03CH37475), vol. 3; 2003. p. 3179–90.
[34] Pannocchia G, Rawlings JB, Wright SJ. Fast, large-scale model predictive control by partial enumeration. Automatica 2007;43(5):852–60.
[35] Parisini T, Zoppoli R. A receding-horizon regulator for nonlinear systems and a neural approximation. Automatica 1995;31(10):1443–51.
[36] Bemporad A, Oliveri A, Poggi T, Storace M. Synthesis of stabilizing model predictive controllers via canonical piecewise affine approximations. In: 49th IEEE Conference on Decision and Control (CDC); 2010. p. 5296–301.
[37] Kvasnica M, Löfberg J, Herceg M, Čirka L, Fikar M. Low-complexity polynomial approximation of explicit MPC via linear programming. In: Proceedings of the 2010 American control conference; June 2010. p. 4713–8.
[38] Domahidi A, Zeilinger MN, Morari M, Jones CN. Learning a feasible and stabilizing explicit model predictive control law by robust optimization. In: 2011 50th IEEE conference on decision and control and European control conference; 2011. p. 513–9.
[39] Summers S, Jones CN, Lygeros J, Morari M. A multiscale approximation scheme for explicit model predictive control with stability, feasibility, and performance guarantees. In: Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese control conference; 2009. p. 6327–32.
[40] Coffey B. Approximating model predictive control with existing building simulation tools and offline optimization. J Build Perform Simul 2013;6(3):220–35.
[41] Le Khang, Bourdais Romain, Guéguen Hervé. From hybrid model predictive control to logical control for shading system: a support vector machine approach. Energy Build 2014;84:352–9.
[42] May-Ostendorp P, Henze GP, Corbin ChD, Rajagopalan B, Felsmann C. Model-predictive control of mixed-mode buildings with rule extraction. Build Environ 2011;46(2):428–37.
[43] Baldi S, Michailidis I, Ravanis Ch, Kosmatopoulos EB. Model-based and model-free "plug-and-play" building energy efficient control. Appl Energy 2015;154:829–41.
[44] Žáčeková E, Pčolka M, Tabačk J, Těžký J, Robinett R, Čelikovský S, et al. Identification and energy efficient control for a building: Getting inspired by MPC. In: 2015 American Control Conference (ACC); July 2015. p. 1671–6.
[45] Klaučo M, Drgoňa J, Kvasnica M, Di Cairano S. Building temperature control by simple mpc-like feedback laws learned from closed-loop data. In: IFAC proceedings. 19th IFAC world congress, vol. 47(3); 2014. p. 581–6.
[46] Heaton Jeff. An empirical analysis of feature engineering for predictive modeling. CoRR, abs/1701.07852; 2017.
[47] Domingos P. A few useful things to know about machine learning. Commun ACM 2012;55(10):78–87.
[48] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural

[3] https://petolau.github.io/.

networks. Science 2006;313(5786):504–7.
[49] Le Quoc, Ranzato Marc'Aurelio, Monga Rajat, Devin Matthieu, Chen Kai, Corrado Greg, et al. Building high-level features using large scale unsupervised learning. In: International conference in machine learning; 2012.
[50] Anderson MR, Antenucci D, Bittorf V, Burgess M, Cafarella MJ, Kumar A, et al. Brainwash: a data system for feature engineering. In: CIDR; 2013.
[51] Magoules Door Frédéric, Zhao Hai-Xiang. Data mining and machine learning in building energy analysis. John Wiley and Sons; 2016.
[52] Dodier Robert H, Henze Gregor P. Statistical analysis of neural networks as applied to building energy prediction. J Solar Energy Eng 2004;126(1):592–600.
[53] Guyon Isabelle, Elisseeff André. An introduction to variable and feature selection. J Mach Learn Res 2003;3:1157–82.
[54] Oldewurtel F, Parisio A, Jones CN, Morari M, Gyalistras D, Gwerder M, et al. Energy efficient building climate control using stochastic model predictive control and weather predictions. American Control Conference (ACC), 2010. IEEE; 2010. p. 5100–5.
[55] Ma Y, Borrelli F, Hencey B, Coffey B, Bengea S, Haves P. Model predictive control for the operation of building cooling systems. IEEE Trans Control Syst Technol 2012;20(3):796–803.
[56] Freire R, Oliveira G, Mendes N. Thermal comfort based predictive controllers for building heating systems. In: Proc of the 16th IFAC world congress; 2005.
[57] Cigler J, Prívara S, Váňa Z, Žáčeková E, Ferkl L. Optimization of predicted mean vote index within model predictive control framework: Computationally tractable solution. Energy Build 2012;52.
[58] Loquercio A, Maqueda AI, del Blanco CR, Scaramuzza D. DroNet: learning to fly by driving. IEEE Robot Autom Lett 2018;3(2):1088–95.
[59] Zhang Tianhao, Kahn Gregory, Levine Sergey, Abbeel Pieter. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. CoRR, abs/1509.06791; 2015.
[60] Kahn Gregory, Zhang Tianhao, Levine Sergey, Abbeel Pieter. PLATO: policy learning using adaptive trajectory optimization. CoRR, abs/1603.00622; 2016.
[61] Lillicrap Timothy P, Hunt Jonathan J, Pritzel Alexander, Heess Nicolas, Erez Tom, Tassa Yuval, et al. Continuous control with deep reinforcement learning. CoRR, abs/1509.02971; 2015.
[62] Ruelens Frederik, Claessens Bert, Quaiyum Salman, De Schutter Bart, Babuska Robert, Belmans Ronnie. Reinforcement learning applied to an electric water heater: from theory to practice. CoRR, abs/1512.00408; 2015.
[63] Sutton RS, Barto AG, Williams RJ.Reinforcement learning is direct adaptive optimal control. In: 1991 American control conference; June 1991. p. 2143–6.
[64] Pr'ıvara S, Cigler J, Váňa Z, Oldewurtel F, Sagerschnig C, Žáčeková E. Building modeling as a crucial part for building predictive control. Energy Build 2013;56(0):8–22.
[65] Baetens R, De Coninck R, Jorissen F, Picard D, Helsen L, Saelens D. OpenIDEAS - an open framework for integrated district energy simulations. In: IPSBA building simulation 2015, Hyderabad; 2015.
[66] Picard D, Jorissen F, Helsen L. Methodology for obtaining linear state space building energy simulation models. In: 11th international modelica conference, Paris; 2015. p. 51–8.
[67] Hornik K. Approximation capabilities of multilayer feedforward networks. Neural Netw 1991;4(2):251–7.
[68] 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.
[69] Baetens R. On externalities of heat pump-based low-energy dwellings at the low-voltage distribution grid. PhD thesis, KU Leuven, Belgium; 2015.
[70] Meteotest. METEONORM Version 6.1 - Edition 2009; 2009.
[71] Technical Committee CEN/TC 156. NBN EN442-1:1996. Radiators and convectors - Part 1: Test methods and rating. Technical report; 1996.
[72] Breiman L. Classification and regression trees. CRC Press; 1993.
[73] Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 1989;2(4):303–14.
[74] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput 2006;18(7):1527–54.
[75] Busseti E, Osband I, Wong S. Deep learning for time series modeling. Technical report; 2012.
[76] Das SK. Feature selection with a linear dependence measure. IEEE Trans Comput 1971;20(9):1106–9.
[77] Jolliffe IT. Principal component analysis. Springer Verlag; 1986.
[78] Krzanowski WJ. Selection of variables to preserve multivariate data structure, using principal components. J Roy Stat Soc. Ser C (Appl Stat) 1987;36(1):22–33.
[79] Lu Y, Cohen I, Zhou XS, Tian Qi. Feature selection using principal feature analysis. Proceedings of the 15th ACM international conference on multimedia, MM '07. New York, NY, USA: ACM; 2007. p. 301–4.
[80] Song F, Guo Z, Mei D. Feature selection using principal component analysis. In: 2010 international conference on system science, engineering design and manufacturing informatization, vol. 1; 2010. p. 27–30.
[81] Fanger P. Thermal comfort. Analysis and applications in environmental engineering. Thermal comfort. Analysis and applications in environmental engineering; 1970.
[82] EN ISO 7730:2006: ergonomics of the thermal environment - analytical determination and interpretation of using calculation of the PMV and PPD indices and local thermal comfort criteria. Number ISO 7730. ISO, Geneva, Switzerland; 2006.
[83] Klaučo M, Kvasnica M. Explicit MPC approach to PMV-based thermal comfort control. In: 53rd IEEE conference on decision and control, vol. 53, Los Angeles, California, USA, December 15–17, 2014; 2014. p. 4856–61.
[84] Löfberg J. YALMIP: a toolbox for modeling and optimization in MATLAB. In: Proc of the CACSD conference, Taipei, Taiwan; 2004. < http://users.isy.liu.se/johanl/yalmip/ > .
[85] Inc. Gurobi Optimization. Gurobi optimizer reference manual; 2012.
[86] Hagan Martin T, Demuth Howard B, Beale Mark. Neural network design. Boston, MA, USA: PWS Publishing Co.; 1996.