Proceedings of Machine Learning Research vol 144:1–12, 2021

Input Convex Neural Networks for Building MPC

Felix Bünning FELIX.BUENNING@EMPA.CH
Adrian Schalbetter ADRIASCH@STUDENT.ETHZ.CH
Automatic Control Laboratory, ETH Zürich, Switzerland
Urban Energy Systems Laboratory, Empa Dübendorf, Switzerland

Ahmed Aboudonia AHMEDAB@CONTROL.EE.ETHZ.CH
Mathias Hudoba de Badyn MBADYN@ETHZ.CH
Automatic Control Laboratory, ETH Zürich, Switzerland

Philipp Heer PHILIPP.HEER@EMPA.CH
Urban Energy Systems Laboratory, Empa Dübendorf, Switzerland

John Lygeros JLYGEROS@ETHZ.CH
Automatic Control Laboratory, ETH Zürich, Switzerland

Abstract

Model Predictive Control in buildings can significantly reduce their energy consumption. The cost and effort necessary for creating and maintaining first principle models for buildings make data-driven modelling an attractive alternative in this domain. In MPC, the models form the basis for an optimization problem whose solution provides the control signals to be applied to the system. The fact that this optimization problem has to be solved repeatedly in real-time implies restrictions on the learning architectures that can be used. Here, we adapt Input Convex Neural Networks that are generally only convex for one-step predictions, for use in building MPC. We introduce additional constraints to their structure and weights to achieve a convex input-output relationship for multi-step ahead predictions. We assess the consequences of the additional constraints for the model accuracy and test the models in a real-life MPC experiment in an apartment in Switzerland. In two five-day cooling experiments, MPC with Input Convex Neural Networks is able to keep room temperatures within comfort constraints while minimizing cooling energy consumption.

Keywords: Input Convex Neural Networks, Model Predictive Control, Building Energy Management

1. Introduction

Model Predictive Control (MPC) in buildings can significantly reduce the energy consumption for space heating and cooling. However, developing and maintaining building models based on first principles is often considered expensive and tedious due to each building being unique (Sturzenegger et al., 2016). Here, data-driven modelling approaches are viewed as a promising alternative (Bünning et al., 2020). Ideally, such approaches should lead to accurate predictions for reliable controller performance and should be usable in convex optimization to find optimal control inputs in a limited amount of time.

As Artificial Neural Networks (ANN) have shown promising results in various domains (Pimenidis and Jayne, 2020; Silver et al., 2017), they are natural candidates to be used as models in building MPC. However, a significant downside is that ANN generally do not lead to convex input-output mappings, making the resulting optimization problem intractable. To address this issue, Amos et al. (2017) present restrictions on the structure and weights of feed-forward ANN to build Fully Input Convex Neural Networks (FICNN), where the model output is convex with respect to all model inputs in single step predictions, and Partially Input Convex Neural Networks (PICNN), where the model output is convex with respect to a subset of the model inputs in single step predictions. They further demonstrate that such networks have high prediction accuracy in many domains, such as multi-label prediction, image completion, and reinforcement learning problems. Chen et al. (2019) extend these formulations to a recurrent network structure for one-shot multi-step ahead predictions, which means that a sequence of outputs is predicted with a sequence of inputs in a single prediction. The authors apply the approach in an MPC scheme for MuJoCo locomotion tasks (Todorov et al., 2012) and to building HVAC control in a simulation case in EnergyPlus.

In this work, we extend the work of Amos et al. (2017) to do multi-shot multi-step predictions with feed-forward networks; predictions are made by re-evaluating the same network repeatedly, by using the outputs of previous timesteps as subsequent inputs. Here, the output at timestep N is not only convex with respect to the input at timestep N, but also with respect to previous inputs. In comparison to (Chen et al., 2019) this allows for more lightweight model architectures with less parameters to fit. We do this for both FICNN and PICNN by further constraining the network weights and by adding activation functions in appropriate places. We then compare the accuracy of the resulting networks to those presented by Amos et al. (2017) on a dataset from a real apartment. We also use the networks as a basis for an MPC controller in a real-life cooling experiment in the same apartment. Our experiments demonstrate the capability of the approach to keep room temperatures within comfort constraints while minimizing cooling energy consumption.

In Section 2 we recap previously presented network structures and demonstrate their problem with keeping convexity in multi-step ahead predictions and MPC schemes. We then introduce further constraints on the networks and add activation functions to address this issue. We further show how the networks can be embedded in MPC schemes. In Section 3 we first compare the prediction accuracy of our networks with previously presented ones. We then introduce the experimental case study and discuss the results. We conclude in Section 4.

2. Methodology

2.1. Model Predictive Control

MPC is a receding horizon optimization scheme for optimal control. At every time instant the state of the system x_0 is measured and an optimisation problem of the form

$$\begin{aligned}
\min_{u,x} \quad & \sum_{k=0}^{N-1} J_k(x_{k+1}, u_k) && \text{(1a)}\\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k, d_k) && \text{(1b)}\\
& (x_{k+1}, u_k) \in (\mathcal{X}_{k+1}, \mathcal{U}_k) && \text{(1c)}\\
& \forall k \in [0, ..., N-1],
\end{aligned}$$

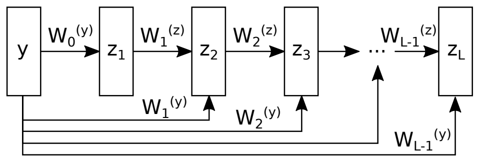

Figure 1: Schematic of a Fully Input Convex Neural Network

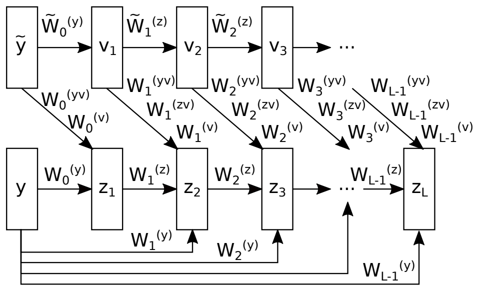

Figure 2: Schematic of a Partially Input Convex Neural Network

is solved, where x, u and d denote states, inputs and disturbances respectively, k denotes the timestep in the horizon N, J denotes the cost function, f denotes the system dynamics and $\mathcal{X}_k$, $\mathcal{U}_k$ encode desired constraints for states and inputs. The controller then applies the first element, u_0^*, of the optimal input sequence, u^*, to the system and the process is repeated.

To find optimal solutions to problem (1) fast and reliably, it is beneficial if the overall optimisation problem is convex in the decision variables u and x. This implies restrictions on the cost function J, dynamics function f, and constraint functions $\mathcal{X}$ and $\mathcal{U}$. A common way for ensuring convexity is to select J to be a convex function (typically, a convex quadratic), select $\mathcal{X}$ and $\mathcal{U}$ to be convex sets, and assume that f is a linear function. Here we show how convexity can be ensured for more general dynamics functions encoded by neural networks.

2.2. Input Convex Neural Networks

Amos et al. (2017) introduced an architecture for feed-forward Neural Networks where the scalar output of the network is convex with respect to all inputs (Fully Input Convex Neural Network or FICNN), or with respect to a subset of the inputs (Partially Input Convex Neural Network or PICNN). These networks are therefore promising candidates for f in problem (1).

Figure 1 shows the schematic of an FICNN. It shows a L-layer fully connected network in which the output of each layer follows

$$z_{i+1} = g_i(W_i^{(z)} z_i + W_i^{(y)} y + b_i), \tag{2}$$

with $z_0 = 0$ and $W_0^{(z)} = 0$, where g_i denotes the activation function for layer $i \in [1, .., L-1]$, $W_i^{(z)}$ and $W_i^{(y)}$ denote the network's weights, and b_i denotes a constant bias. Both weights and biases are model parameters that are determined during network training. The network output z_L is convex with respect to the elements of the input vector y if all $W_{1:L-1}^{(z)}$ are non-negative and all activation functions g_i are convex and non-decreasing (Amos et al., 2017).

Similarly, Figure 2 depicts a PICNN. Here, the output of each network layer follows

$$\begin{aligned} v_{i+1} &= \tilde{g}_i(\tilde{W}_i v_i + \tilde{b}_i), \\ z_{i+1} &= g_i\Big[W_i^{(z)}\Big\{z_i \circ [W_i^{(zv)} v_i + b_i^{(z)}]\Big\} + \quad W_i^{(y)}\Big\{y \circ [W_i^{(yv)} v_i + b_i^{(y)}]\Big\} + W_i^{(v)} v_i + b_i\Big], \end{aligned} \tag{3}$$

with $z_0 = 0$, $W_0^{(z)} = 0$ and $v_0 = \tilde{y}$, where $\tilde{g}_i$ and g_i are activation functions, $\tilde{W}_i$, $W_i^{(z)}$, $W_i^{(zv)}$, $W_i^{(yv)}$, $W_i^{(v)}$ are input weights, $\tilde{b}_i$, b_i, $b_i^{(z)}$, $b_i^{(y)}$ are constant biases, and $\circ$ denotes the Hadamard product. Under the condition that all weights $W_{1:L-1}^{(z)}$ are non-negative and all activation functions g_i are convex and non-decreasing, the model output z_L is convex with respect to the elements of the input vector y (but not necessarily with respect to the elements of $\tilde{y}$) (Amos et al., 2017).

FICNN and PICNN guarantee input-output convex behaviour, making them promising candidates for MPC schemes, as mentioned above. However, when ICNNs are applied for f in this context, for example as $x_{k+1} = f(y = (x_k, u_k, d_k))$ in problem (1), it becomes evident that there is an issue with convexity in multi-step ahead prediction. While the first step $x_1 = f(x_0, u_0, d_0)$ is a convex function of the decision variables, the second one is not guaranteed to be. For example, if a 1-layer FICNN is considered, the state at timestep 2 follows

$$x_2 = f(x_1, u_1, d_1) = g_0(W_0^{(y)}(x_1, u_1, d_1) + b_0). \tag{4}$$

As the elements of x_1 are convex functions of the elements of x_0 and u_0, take $x_1 = x_0^2 + u_0^2$ as a scalar example, and $W_0^{(y)}$ can attain any value (thus also a negative one), the term $W_0^{(y)}(x_1, u_1, d_1)$ can become a concave function in the elements of x_0 and u_0, for example $(-1)(x_0^2 + u_0^2)$. The state $x_2 = f(x_1, u_1, d_1)$ is thus not guaranteed to be convex with respect to the elements of x_0 and u_0.

2.3. ICNN for multi-step ahead prediction

To make ANNs input convex in the face of multi-step ahead prediction, Chen et al. (2019) have presented a solution for Fully Input Convex Recurrent Neural Networks. Here, we present a solution for Fully and Partially Input Convex Feed-Forward Neural Networks. We accomplish this through additional constraints in the network architecture and the use of ReLu activation functions in appropriate places in the network topology. Our approach readily extends to FICNN, where it becomes the feed-forward counterpart of the approach developed by Chen et al. (2019) for recurrent neural networks.

Proposition 1: *Consider the two FICNNs $f_1(y_1)$ and $f_2(y_2)$ as defined in eq. (2). The composition $f_2(y_2 = (\hat{y}_2, f_1(y_1)))$ is a convex function with respect to the elements of the vector y_1, if all weights $W_i^{(z)}$ and $W_i^{(y)}$ are non-negative and all functions g_i are convex and non-decreasing. Here, $\hat{y}_2$ are the inputs unrelated to f_1.*

The proof follows from the fact that non-negative sums of convex functions are convex and that compositions of a convex function and a convex non-decreasing function are also convex. Looking at eq. (2), the elements of $W_i^{(z)} z_i$ are convex functions assuming that z_i is convex in its inputs and all elements of $W_i^{(z)}$ are non-negative. The parameter b_i is a constant. Generally, the term $W_i^{(y)} y$ is convex in y for any $W_i^{(y)}$ if y is constant because a linear function with negative gradient is also convex (Amos et al. (2017)). Here, we require the elements of $W_i^{(y)}$ to be non-negative, because y itself might be a convex function in the form of an ICNN. The term $(W_i^{(z)} z_i + W_i^{(y)} y + b_i)$ is therefore a non-negative sum of convex functions and $g_i(W_i^{(z)} z_i + W_i^{(y)} y + b_i)$ a composition of a convex function and a convex non-decreasing function (for example the commonly used ReLu function $g_i = \max(x, 0)$). For the example of eq. (4), as $W_0^{(y)}$ is now constrained to be non-negative, $W_0^{(y)}(x_1 = f(x_0, u_0, d_0))$ is convex in the elements of x_0 and u_0, if x_1 is convex in the elements of x_0 and u_0 (which is the case). Thus, x_2 is also convex in the elements of x_0 and u_0. The network output z_L is a convex non-decreasing function of the elements of the input y. An alternative proof can be constructed from showing that a FICNN is a convex non-decreasing function and using the composition rule of convex and convex non-decreasing functions.

In the case of PICNN, we propose the following structure for the outputs of the network layers for input convex multi-step prediction,

$$
\begin{aligned}
v_{i+1} &= \tilde{g}_i(\tilde{W}_i v_i + \tilde{b}_i) \\
z_{i+1} &= g_i\left[W_i^{(z)}\left(z_i \circ g_i^{(zv)}[W_i^{(zv)} v_i + b_i^{(z)}]\right) + \quad W_i^{(y)}\left(y \circ g_i^{(yv)}[W_i^{(yv)} v_i + b_i^{(y)}]\right) + W_i^{(v)} v_i + b_i\right],
\end{aligned}
\tag{5}
$$

where the activation functions $g_i^{(zv)}$ and $g_i^{(yv)}$ are added compared to eq. (3).

Proposition 2: *Consider the two PICNNs $f_1(\tilde{y}_1, y_1)$ and $f_2(\tilde{y}_2, y_2)$ as defined in eq. (5). The composition $f_2(\tilde{y}_2, y_2 = (\hat{y}_2, f_1(\tilde{y}_1, y_1)))$ is a convex function with respect to the elements of y_1 (and y_2), but not necessarily with respect to the elemetns of $\tilde{y}_1$ (and $\tilde{y}_2$), if all weights $W_i^{(z)}$ and $W_i^{(y)}$ are non-negative, all functions $g_i^{(zv)}$ and $g_i^{(yv)}$ map to a non-negative value and the function g_i is convex and non-decreasing. Here, $\hat{y}_2$ are the inputs unrelated to f_1.*

The proof again follows from only applying operations that maintain convexity. Going through eq. (5) term by term, $z_i \circ g_i^{(zv)}[W_i^{(zv)} v_i + b_i^{(z)}]$ is convex in the elements of the inputs of z_i if z_i is a convex function because $g_i^{(zv)}$ maps all negative values of $[W_i^{(zv)} v_i + b_i^{(z)}]$ to zero. As $W_i^{(z)}$ is non-negative, $W_i^{(z)}(z_i \circ g_i^{(zv)}[W_i^{(zv)} v_i + b_i^{(z)}])$ is also convex. The same argument can be made for the next term $W_i^{(y)}(y \circ g_i^{(yv)}[W_i^{(yv)} v_i + b_i^{(y)}])$ regarding the convexity in the elements of y and the elements of inputs of y. Here, we need $g_i^{(yv)}$ to map all negative values to zero, because y is not a constant (as in Amos et al. (2017)), but a convex function itself. As g_i is convex non-decreasing, the composition of g_i and before-mentioned terms is a convex function. Eq. (5) is thus convex with respect to the elements of the input vectors of any convex functions y and z_i. We note in passing, that other formulations of FICNN and PICNN can be thought of and will give the same result as long as it is ensured that z_{i+1} is convex non-decreasing in the elements of y.

2.4. Embedding ICNN in convex MPC

The adaptations in Section 2.3 can be used to formulate a convex MPC scheme (Boyd and Vandenberghe, 2004), with a convex cost function and a convex feasible set, commonly used for building control as

$$
\begin{aligned}
\min_{u,x} \quad & \sum_{k=0}^{N-1} J_k(x_{k+1}, u_k) && \text{(6a)} \\
\text{s.t.} \quad & \Big(x_{k+1} = f(x_k, u_k, d_k)\Big) \le x_{max} && \text{(6b)} \\
& u_{min} \le u_k \le u_{max} && \text{(6c)} \\
& \forall k \in [0, ..., N-1],
\end{aligned}
$$

where x, u and d are states, inputs and disturbances respectively. J is an appropriate convex cost function, f are the dynamics represented by an ICNN, u_{min} and u_{max} are lower and upper input constraints and x_{max} is an upper state constraint. Constraint (6b) defines a convex set because the equality constraint can be eliminated. For example, $x_2 = f(x_1, u_1, d_1)$ and $x_2 \leq x_{max}$ can be replaced with $f(f(x_0, u_0, d_0), u_1, d_1) \leq x_{max}$. Note, that lower state constraints are generally not possible, even when f is a convex function, as super-level sets of convex functions are generally not convex.

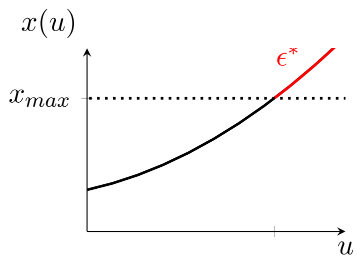

Figure 3: Optimal slack variable ϵ^* at upper state constraints

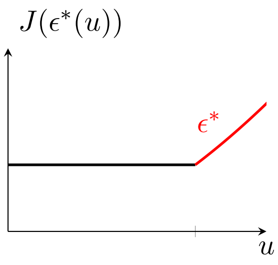

Figure 4: Convex cost function J with slack variable on state constraint

Soft state constraints in the form of slack variables are often necessary in practical applications of MPC to ensure feasibility at all times. Adding soft constraints changes the problem to

$$\min_{u,x,\epsilon} \quad \sum_{k=0}^{N-1} J_k(x_{k+1}, u_k, \epsilon_k) \tag{7a}$$

$$\text{s.t.} \quad \Big(x_{k+1} = f(x_k, u_k, d_k)\Big) \leq x_{max} + \epsilon_k \tag{7b}$$

$$\epsilon_k \geq 0 \tag{7c}$$

$$u_{min} \leq u_k \leq u_{max} \tag{7d}$$

$$\forall k \in [0, ..., N-1],$$

where ϵ is the slack variable, and J is now a function of x, u and ϵ. The slack variable ϵ^* (i.e the ϵ that minimizes J) is a convex non-decreasing function of u for $x \geq x_{max}$, as depicted on the right of Fig. 3. J remains convex as shown for the example $J = const + \epsilon$ in Fig. 4.

3. Case studies

3.1. Problem configuration

We apply both introduced networks in a numerical and an experimental case study in the Urban Mining and Recycling (UMAR) unit of the NEST demonstrator building (Richner et al., 2017). The unit is an occupied apartment comprising two bedrooms, a living room, two bathrooms, and an entrance area. The rooms are equipped with ceiling heating and cooling panels, which are connected to the central heating and cooling system of NEST through a heat exchanger. In standard operation, the room temperature is controlled through thermostats that open or close valves to the ceiling panels. The supply temperature and the supply pump pressure are constant.

In different case studies, the ICNN are used to predict the room temperature $T_{br,k+1} = T_{br,k} + \Delta T_{br,k+1}$ of one of the bedrooms, by predicting the temperature change

$$\Delta T_{br,k+1} = f\Big(\dot{Q}_{sol,k}, \dot{Q}_{sol,k-1}, \dot{Q}_{sol,k-2}, t_{sin}, t_{cos}, \\ \delta T_{l,k}, \delta T_{amb,k}, \Delta T_{br,k}, \Delta T_{br,k-1}, \Delta T_{br,k-2}, Q_{u,k}\Big), \tag{8}$$

where, $\dot{Q}_{sol,k}, \dot{Q}_{sol,k-1}, \dot{Q}_{sol,k-2}$ are the global solar irradiation at the current timestep and the two previous timesteps, t_{sin} and t_{cos} are the time of the day encoded as a sine and cosine function, $\delta T_{l,k}$ and $\delta T_{amb,k}$ are the temperature differences between the bedroom and living room, and the bedroom and ambient at timestep k respectively, and $Q_{u,k}$ denotes the heating/cooling energy (i.e. the control input). Note, that Δ defines differences in terms of time, while δ defines differences in terms of location. The feature selection is a result of an extensive study based on k-fold cross validation, where measurement data of 1 year from the UMAR unit is used (Schalbetter, 2020). Function f represents either a FICNN or a PICNN; in the case of PICNN, the features are divided into convex features $y = (\delta T_{amb,k}, \Delta T_{br,k}, \Delta T_{br,k-1}, \Delta T_{br,k-2}, Q_{u,k})$, which are related to the state or control inputs, and features $\tilde{y} = (\dot{Q}_{sol,k}, \dot{Q}_{sol,k-1}, \dot{Q}_{sol,k-2}, t, \delta T_{l,k})$, which do not require convexity because they are related to the disturbances and are not optimized over.[1]

We use models with two different sampling rates, 20 minutes and 180 minutes, for predictions up to 1 hour and predictions >1 hour respectively, as models with larger sampling rates showed better prediction performance for long horizons in preliminary experiments. The hyperparameters for both models can be seen in Table 1. All parameter decisions are results of the cross validation study (Schalbetter, 2020). To allow the network output z_L (i.e. $\Delta T_{br,k+1}$) to be negative, we use a shifted ReLu function, $g_i(x) = \max(x, 0) - \beta$, as the final activation function in the output layer instead of a regular one. Here, β is a hyperparameter of the network.

hyperparameters	**1 hour model**	**>1 hour model**
optimizer	Adam	Adam
epochs	20	40
layers	4	4
nodes per layer	9	8
ReLU-offset (β)	0.8	12

Table 1: Tuned hyperparameters for the ICNN for 1 hour (left) and >1 hour predictions (right).

3.2. Numerical case study

In the numerical study, we compare the prediction performance of our ICNN to those introduced by Amos et al. (2017) for room temperature predictions of 1 hour and 6 hours. To account for the fact that training neural networks is a non-convex problem, we divide the measurement data of 1 year (measured every minute and down sampled to 20 and 180 minutes) from UMAR into 12 folds of size one month, use 9 randomly selected folds for training, the remaining 3 folds for validation and

1. Note, that $\delta T_{l,k}$ is kept constant during the prediction and thus does not need to be in the convex inputs.

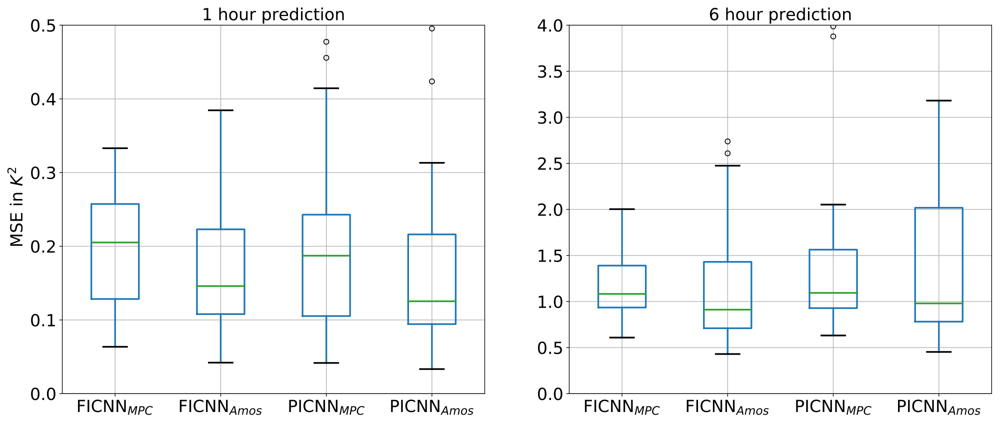

Figure 5: Prediction accuracy of different ICNN for 1 hour predictions and 6 hour predictions in UMAR

repeat this step 100 times for each network. The training time on a laptop computer[2] lies between 3 and 3.5 seconds per epoch for all different types of networks.

The results of the comparison are shown in Figure 5. The left plot shows the boxplots of the mean squared error (MSE) in K^2 for 1 hour predictions, with median, interquartile range (IQR), and minimum and maximum for the different types of networks. The index MPC denotes the networks presented in this article, while the index $Amos$ denotes the ones presented in Amos et al. (2017). It can be seen that the median of the MSE is significantly lower for the Amos networks, for both FICNN and PICNN. The IQR is comparable for all networks, while minimum and maximum do not show a decisive trend. In the case of the 6 hour prediction, depicted on the right of Figure 5, the IQR and min-max range are smaller for our networks, while the median is again lower for the networks presented in Amos et al. (2017). The increase in median compared to Amos et al. (2017) can be expected for general system dynamics, as our networks are restricted to convex non-decreasing functions while the those of Amos et al. (2017) allow more general convex functions.

The results raise a question regarding the general restrictiveness of the approach. Amos et al. (2017) have performed numerical analyses in a variety of learning domains and found that the enforced convexity does not significantly hamper model accuracy. Our approach adds the constraint of positive monotonicity. The class of monotone systems (Hirsch and Smith, 2006; Cosner, 1996) comprises many chemical (Banaji, 2009), biological (Angeli, 2015), physical (Sánchez, 2002) and economic (Kamihigashi and Stachurski, 2014) models. Buildings are also generally positive (Khosravi and Smith, 2019) and also monotone systems, for example the room temperature is a positive monotone function of the valve position of a radiator or of the supply temperature of the heating system. Therefore, the unambiguity of the numerical results is surprising. A possible explanation is that removing the monotonicity requirement allows a better approximation of internal gains from occupancy as a function of the time and solar gains through windows as a function of the time and the global horizontal irradiation.

2. Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 16.0GB RAM

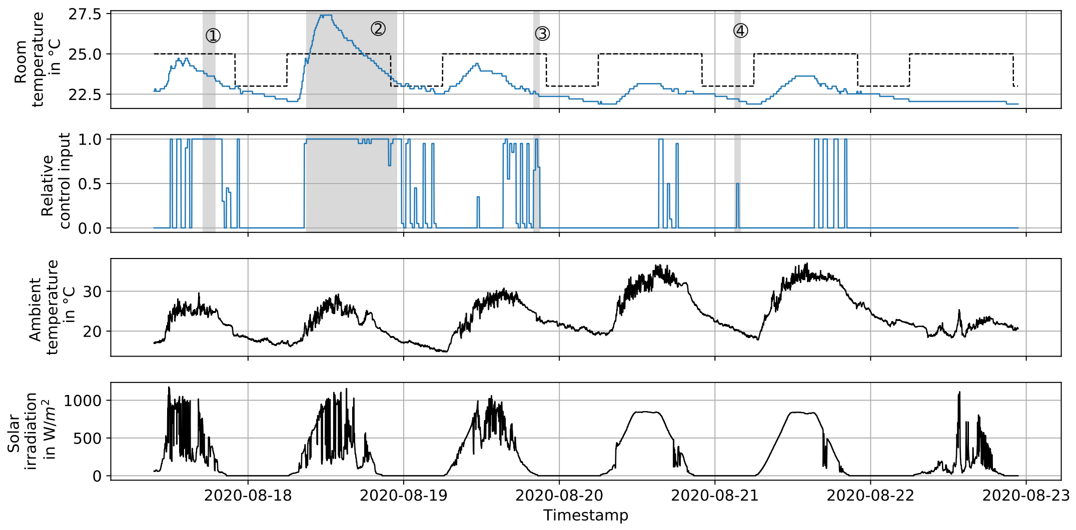

Figure 6: Experimental results of room temperature control by MPC with a FICNN.

3.3. Experimental case study

In two real-life experiments we have applied the ICNN to MPC for room temperature control in a bedroom of the UMAR apartment during the cooling season. The MPC set up corresponds to problem (8) with the cost function $J_k = Ru_k^2 + \lambda\epsilon_k^2$. The problem parameters are set to $R = 1$, $\lambda = 100$, which proved to be successful in earlier studies (Bünning et al., 2020). There are no state costs, which is a common choice, as there are usually no stability issues with buildings. The horizon is set to 7 hours, divided into three predictions with the network with 20 minutes sampling time and two with 180 minutes sampling time. *Move blocking* (Cagienard et al., 2007) is applied here because longer sampling times led to better prediction accuracy for long horizons in preliminary studies. The input constraints are $u_{min} = -0.6$ kWh and $u_{max} = 0$ kWh and are implemented with pulse-width modulation of the supply valves. The state constraint x_{max} reflects a comfort constraints for the room temperatures and is time varying and shown in the results. The scheme was programmed in Python 3 and with COBYLA (Powell, 1994) as a solver, leading to solution times of less than ten seconds on the same laptop computer. This is more than sufficient for applications in building control, where sampling times commonly range from 10 to 30 minutes.

Figure 6 shows the results of MPC in combination with a FICNN applied to one of the bedrooms of UMAR. The first plot depicts the room temperature in blue and the time-varying comfort constraints in dashed black. The comfort constraint is 25 °C between 6 am and 10 pm, and 23 °C otherwise. The second plot shows the relative control (cooling) input, where *Relative control input* maps u_{min} to 1 and u_{max} to 0. It can be seen that the MPC controller keeps the temperature below the comfort constraint during most times and allows the temperature to rise during times when the comfort constraint is higher. One exception is the second day (marked with ②). Here, the normally shut window blinds were automatically opened due to high winds, and the cooling system was not able to compensate the solar gains although running at full capacity. During the other times that are marked in grey (and with ①, ③, ④), connection to the actuators was lost and the standard thermostat controller took over. The lower two plots show the ambient conditions, which are measured at the top of the building.

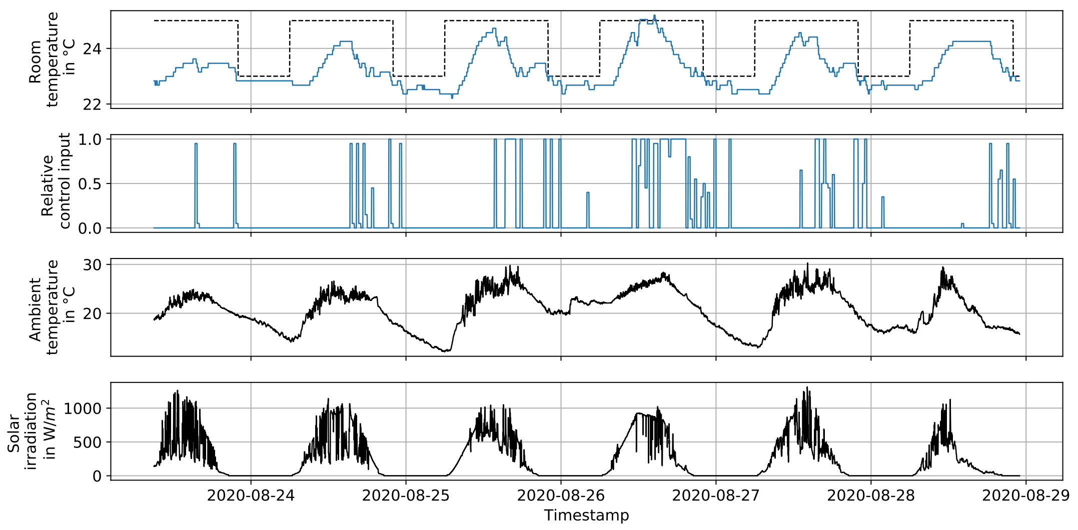

Figure 7: Experimental results of room temperature control by MPC with a PICNN.

Figure 7 shows the results of the second experiment, where MPC with a PICNN was applied in the other bedroom. As this bedroom is more sensitive to solar gains due to a neighbouring apartment, the temperature rise during the day is much more visible. It is evident that the used PICNN has sufficient prediction accuracy for MPC because the controller exactly meets the lowered comfort constraint at 10 pm on every single occasion. Comparing this result to the performance of FICNN in Figure 6 it appears that the additional generality of PICNN pays off, as the FICNN tends to cool down the room more than necessary. As the experiments are conducted in different rooms on different days, the evidence is not conclusive, but there is at least a strong indication in this direction.

4. Conclusion

In this work, we have proposed constraints and activation functions to make ICNN input-convex, not only for single step predictions, but also for multi-step ahead predictions. Although these networks show a decreased model accuracy compared to one-step ahead ICNN, the adoptions enable the networks to be used in (quasi-)convex MPC schemes. In two real-life experiments with ICNN in building energy MPC, the controller kept the room temperatures within comfort constraints, while exploiting time periods with relaxed constraints to save cooling energy. Ongoing research focuses on more general formulations of ICNN and and constraints in related MPC formulations.

Acknowledgments

This research project is financially supported by the Swiss Innovation Agency Innosuisse and is part of the Swiss Competence Center for Energy Research SCCER FEEB&D.

References

Brandon Amos, Lei Xu, and J. Zico Kolter. Input Convex Neural Networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 1, pages 192–206. PMLR, jul 2017. ISBN 9781510855144.

David Angeli. Monotone Systems in Biology. In *Encyclopedia of Systems and Control*, pages 769–776. Springer London, 2015. doi: 10.1007/978-1-4471-5058-9_90.

Murad Banaji. Monotonicity in chemical reaction systems. *Dynamical Systems*, 24(1):1–30, mar 2009. ISSN 14689367. doi: 10.1080/14689360802243813.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Felix Bünning, Benjamin Huber, Philipp Heer, Ahmed Aboudonia, and John Lygeros. Experimental demonstration of data predictive control for energy optimization and thermal comfort in buildings. *Energy and Buildings*, 211:109792, mar 2020. ISSN 03787788. doi: 10.1016/j.enbuild.2020.109792.

R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, jul 2007. ISSN 09591524. doi: 10.1016/j.jprocont.2007.01.001.

Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Chris Cosner. Book Review: Monotone dynamical systems: An introduction to the theory of competitive and cooperative systems. *Bulletin of the American Mathematical Society*, 33(02):203–210, mar 1996. ISSN 0273-0979. doi: 10.1090/s0273-0979-96-00642-8.

M. W. Hirsch and Hal Smith. Chapter 4 Monotone Dynamical Systems. *Handbook of Differential Equations: Ordinary Differential Equations*, 2:239–357, jan 2006. ISSN 18745725. doi: 10.1016/S1874-5725(05)80006-9.

Takashi Kamihigashi and John Stachurski. Stochastic stability in monotone economies. *Theoretical Economics*, 9(2):383–407, may 2014. ISSN 19336837. doi: 10.3982/TE1367.

Mohammad Khosravi and Roy S. Smith. Kernel-Based Identification of Positive Systems. In *Proceedings of the IEEE Conference on Decision and Control*, volume 2019-Decem, pages 1740–1745. Institute of Electrical and Electronics Engineers Inc., dec 2019. ISBN 9781728113982. doi: 10.1109/CDC40024.2019.9029276.

Elias Pimenidis and Chrisina Jayne. Special issue on engineering applications of neural networks, aug 2020. ISSN 14333058.

Michael J. D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer Netherlands, 1994. doi: 10.1007/978-94-015-8330-5_4.

Peter Richner, Philipp Heer, Reto Largo, Enrico Marchesi, and Mark Zimmermann. NEST - A platform for the acceleration of innovation in buildings. *Informes de la Construccion*, 69(548): 1–8, jan 2017. ISSN 19883234. doi: 10.3989/id.55380.

Luis A. Sánchez. An application of the theory of monotone systems to an electrical circuit. In *Royal Society of Edinburgh - Proceedings A*, volume 132, pages 711–728. Royal Society of Edingburgh Scotland Foundation, 2002. doi: 10.1017/s0308210500001852.

Adrian Schalbetter. Input Convex Neural Networks for Energy Optimization in an occupied Apartment. 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, oct 2017. ISSN 14764687. doi: 10.1038/nature24270.

David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S. Smith. Model Predictive Climate Control of a Swiss Office Building: Implementation, Results, and Cost–Benefit Analysis. *IEEE Transactions on Control Systems Technology*, 24(1):1–12, jan 2016. ISSN 1063-6536. doi: 10.1109/TCST.2015.2415411.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6386109.