A Building Automation and Control micro-service architecture using Physics Inspired Neural Networks

Johannes P. Maree[1], Marius Bagle[2,3*]
[1]SINTEF Digital, Trondheim, Norway
[2]SINTEF Community, Oslo, Norway
[3]Department of Architecture, NTNU, Trondheim, Norway

Abstract

In this work, we present a micro-service architecture which defines a Digital Twin (DT) framework for adaptive building automation and control. The DT framework primarily involves the orchestration of several containerized micro-services, promoting the scalability and deployability of the proposed framework within the industrial context. In the proposed framework, containerized micro-services facilitate: (i) model-based control strategies; (ii) data-driven learning; (iii) data management; (iv) the inclusion of an internal High-Fidelity Simulator (HFS) to enable bootstrapped learning; and (v) a User Interface/User Experience (UI/UE) micro-service orchestrator. To validate the usefulness of the proposed framework, we implement a Physics Inspired Neural Network (PINN) to adapt the model-based control strategies for plant-model uncertainty and utilize bootstrap sampling against an internal HFS.

Introduction

Central to smart building automation and control is the concept of Digital Twin (DT) architectures. A Digital Twin can be defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making. While it is already common to use numerical tools and simulation models in the prototyping and design phase of new projects, advances within computational pipelines, artificial intelligence, big data cybernetics bring the promise of digital twins closer to society (Rasheed et al., 2020).

In the context of buildings, DTs has mostly been explored in the first phase of development (i.e., design and construction), and less attention has been paid to the operation and management (O&M) phase, which has the longest time span of the asset life cycle (Lu et al., 2019). The process of developing, validating and maturing advanced control and monitoring strategies is often considered the biggest bottleneck, necessitating extensive off-line testing prior to deployment (Drgoňa et al., 2020). The adoption and deployment of complex building automation strategies, such as Model Predictive Control (MPC) and Fault Detection and Diagnosis (FDD), can extensively benefit from DT frameworks.

The emergence of distributed cloud-based computing infrastructure has culminated in the adoption of several service models such as Infrastructure-, Platform-, and Software-as-a-Service (IaaS, PaaS, SaaS) (Mohammed et al., 2021). Within the domain of SaaS, one is concerned with micro-services deployed on a cloud-based platform providing web-based functionality and data to end-point customers.

The Building and Automation Control System Software as a Service (BACS^{2}aaS) framework, proposed in this work (see Figure 1), adopts a SaaS service model characteristic by proving critical control, monitoring and learning applications for the automation and intelligent management of smart buildings. BACS^{2}aaS, in addition, adopts as PaaS characteristic where the micro-services within the SaaS domain are containerized within Docker containers. This enables the possibility for building and deploying platform agnostic micro-services, within the BACS^{2}aaS framework, on any platform infrastructure supporting Docker. BACS^{2}aaS comprises of five core Functional Units (FU)s, or micro-services. With reference to Figure 1, a high-level description of the BACS^{2}aaS framework is in order, where a more detailed description of the individual FU's are provided in subsequent sections. BACS^{2}aaS information and functional data flow is managed by the FU1, here called the Orchestrator. FU1 is the primary interface between BACS^{2}aaS and the external environment providing an informative User Interface/User

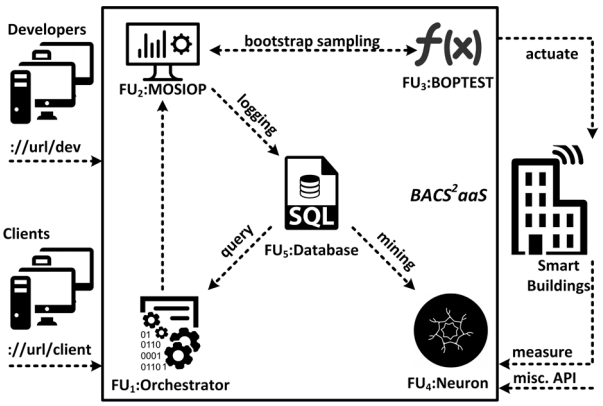

Figure 1: A cloud-based SaaS framework for the automation and management of smart buildings.

*Corresponding author: mariuseb@stud.ntnu.no

Experience (UI/UE) for querying and monitoring the automation of smart buildings. As a secondary prerogative, FU1 subscribes to external third-party application programming interface (API) data services to stream critical information required for other micro-services related to control and learning. FU2 defines a micro-service for proving model-based control law to actuate either an internal High Fidelity Simulator (HFS), being encapsulated in FU3, or some external building. Historical control actions, measurements taking from some external building or internal HFS, and third-party supplied data is all stored on a time-series database (FU5) which is consumed by the data-driven micro-sevice contained in NEURON (FU4). The latter, in principle, employ all data-driven learning strategies typically associated with machine learning and artificial intelligence.

BACS^{2}aaS micro-services

To follow is a more detailed discussion of the functionality associated with the respective micro-services, previously introduced in context of Figure 1.

High fidelity simulator

In building performance simulation, we distinguish between three main modelling paradigms: (i) black-box, (ii) grey-box, and (iii) white-box. For control-oriented purposes, all three paradigms can be used, either separately or in combination (Hensen and Lamberts, 2019). Furthermore, in a building energy model, simplifications like completely mixed air (as opposed to fine-grained computational fluid dynamics (CFD) for simulation of air flow transients arising from e.g., window or door opening) and 1D heat transfer (from 3D partial differential equations) are often made. This simplification is account for computational constraints on the one hand, and reduced impact on variables of interest on the other hand. It should be pointed out that white-box/high-fidelity simulators in this context implies *traditional* building energy models (with the above-mentioned simplifications, among other things). In Arroyo et al. (2022), the suitability of the three different modelling paradigms are compared by doing experiments on a thermally activated buildings with representative models from each paradigm. The authors concluded that a modelling approach that synthesizes the physics- and data-driven approaches is a promising avenue going forward. Strengths of each paradigm, such as e.g., generalizability of white-box models and the ease of calibration/parameter identification for grey-box and black-box models can be combined and leveraged for the most deployable, adaptive and scalable solution to smart building control.

Buildings Optimization Performance Tests (Blum et al., 2021) (BOPTEST) framework defines a virtual test-bed for prototyping control algorithms for HVAC systems in buildings. The main motivation behind the development of this framework was that test-case setup in different experiments in literature usually is done in an ad-hoc distributed fashion, rendering comparisons between results

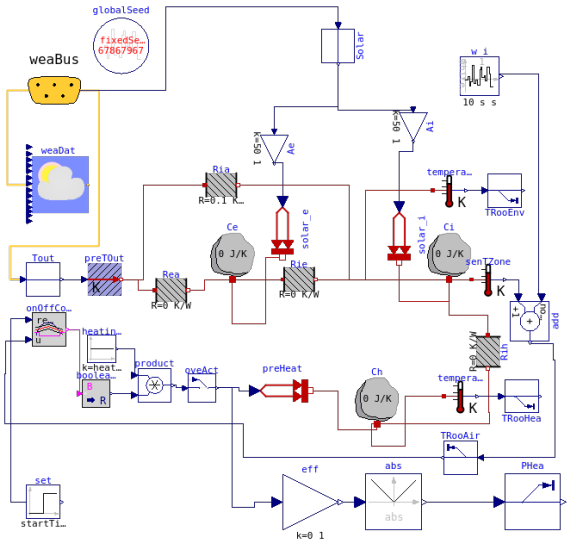

Figure 2: Simple Modelica testcase.

impossible. The framework is based on the modelling language Modelica, which enables object-oriented cyber-physical modelling and the international standard Functional Mock-up Interface (FMI), facilitating co-simulation and model exchange. Building models developed in a collection of Modelica libraries (ie., Buildings, IDEAS) are compiled into Functional Mock-up Units (FMU)s and packaged with the necessary boundary conditions in a Docker container (Anderson, 2015), allowing the control signals of local-level controllers in the building models to be overwritten via a RESTful API. This enables mimicking an ideal situation in which a building's Building Automation Control (BAC), and system, is integrated with a potential cloud-based platform, available for overwriting. In the context of BACS^{2}aaS, FU3 executes a fork operation on the main developer branch of BOPTEST, which is subsequently updated, modified, and packaged as its own micro-service within BACS^{2}aaS to serve as an internal HFS. As an example case study in this work, Figure 2 shows a relatively simple Modelica-model that has been implemented for testing purposes. This model is analogous to a 3R3C-network. (Bacher and Madsen, 2011), with parameters based on identification experiments carried out on a real building located in Børrestuveien 3, Oslo. The baseline controller is an on-off controller, modulating the heat flow to the heater state T_h such that the interior temperature T_i stays within certain bounds. A white noise generator is added to the measurement of T_i. Typical meteorological year (TMY) weather data from Oslo is used.

Model-based control

The MOdelling, SImulation and OPtimization (MOSIOP), encapsulated in FU2, defines a micro-service within the BACS^{2}aaS framework which includes a numerical framework for formulating and solving discrete-time optimal control optimization problems.

Model Predictive Control (MPC) defines a particular control strategy which primarily involves solving an optimal

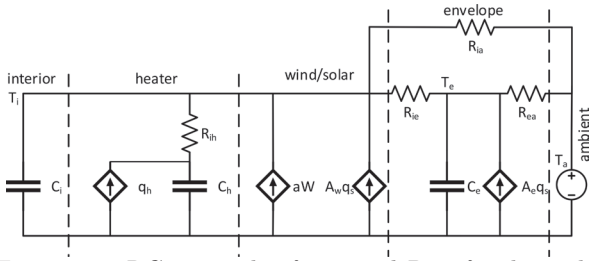

Figure 3: RC-network of TiTeThRia for heat dynamic modelling of buildings

control problem, sequentially, over a receding horizon and has seen significant adoption within industry (Qin and Badgwell, 2003). Benefits of MPC is that physical white-box models and their operational constraints are explicitly incorporated into an optimization problem which, when solved, defines a control law that promotes the optimization of several operational and economic objectives (often conflicting), simultaneously. Solving the MPC problem over a receding horizon provides, in addition, a form of robustness to external system disturbances as new information becomes available and is incorporated into the optimization problem.

The practical implementation of MPC within the building industry is, however, lagging behind the process industry due to: (i) challenges associated with controller model development and estimation of unknown states (Blum et al., 2019); and, (ii) the lack of operational knowledge among building management system engineers w.r.t modern optimal control methods (Drgoňa et al., 2020).

To formulate a numerical tractable optimal control problem, that incorporates a high-fidelity model including its operational constraints explicitly (i.e., the model associated with Figure 2), may be challenging from a modelling and computational perspective when applied within a MPC strategy. Often, MPC resorts to utilizing a reduced order physical model of the process of interest that promotes a computational feasible formulation when one is concerned with solving it in real-time during online operation. In context of the BACS^{2}aaS framework and the HFS model depicted by Figure 2; in this work we are concerned with a reduced order 3R3C model (see Figure (3) (Maree et al., 2021)) to be incorporated within a MPC formulation, defined by the following Ordinary Differential Equation (ODE) (Bacher and Madsen, 2011):

$$\frac{dT_i}{dt} = \frac{T_h - T_i}{R_{ih}C_i} + \frac{T_e - T_i}{R_{ie}C_i} + \frac{T_a - T_i}{R_{ia}C_i} + \frac{A_w q_s + aW}{C_i} \tag{1a}$$

$$\frac{dT_e}{dt} = \frac{T_i - T_e}{R_{ie}C_e} + \frac{T_a - T_e}{R_{ea}C_e} + \frac{A_e q_s}{C_e} \tag{1b}$$

$$\frac{dT_h}{dt} = \frac{T_i - T_h}{R_{ih}C_h} + \frac{q_h}{C_h} \tag{1c}$$

For numerical implementation of a MPC strategy, we are often concerned with discrete-time formulations for time $t = k \; \forall k \in \mathbb{I}_{\geq 0}$. For (1), let the $x_k^\top := [T_i, T_e, T_h] \in \mathbb{X}^n$ define the temperature states of the internal, building envelope and heater, respectively at time $t = k$. The process is explicitly actuated by controlling the energy flux from the heating system, $u_k := [q_h] \in \mathbb{U}^m$. Here, $\{n, m\}$ defines the state and control vector dimensions, respectively. The external forecast signals $r_k^\top := [T_a, q_s]$, being ambient air temperature and solar irradiance, actuates the process (1), implicitly. The MPC objective is to minimize the weighted mean square error of the following stage cost function:

$$l(x_k, u_k) := \|T_i - T_{ref}\|_{\alpha_1}^2 + \|q_h\|_{\alpha_2}^2 \tag{2}$$

with the primary optimization objective to track some user defined setpoint T_{ref} (weighted by α_1). As a secondary objective is to minimize the weighted (α_2) supplied energy q_h. The MPC value function is defined over the prediction horizon of N time steps as:

$$V_N(x_0, \mathbf{r}) := \sum_{k=0}^{N} l(x_k, u_k) \tag{3}$$

where x_0 is some admissible initial state $x_0 \in \mathbb{X}$. $\mathbf{r} = [r_0, ..., r_N]$ defines some externally supplied forecast trajectory. The MPC problem is concerned with evaluating the follow optimization problem:

$$\min_{\mathbf{u}} V_N(x_0, \mathbf{r}) \tag{4a}$$

$$x_{k+1} = f(x_k, u_k) \tag{4b}$$

$$x_k \in \mathbb{X}, \; u_k \in \mathbb{U} \tag{4c}$$

$$x_0 = \hat{x}_t \tag{4d}$$

In (4d), we initialize (4) by some externally supplied state estimate $\hat{x}_t$. The discrete system evolution (4b) is defined as the discrete-time counterpart of (1) and can readily be evaluated by any numerical integrator. The optimal solution to (4) is evaluated for $\mathbf{u}^{*\top} := [u_0^*, ..., u_{N-1}^*]$ where the MPC receding horizon control law is defined as the first optimal control move: $\kappa_N(x_0) := u_0^*$. Having defined this control law, one can either actuate the HFS in FU3, or some external smart building, by applying this control law. The the next sampling time instant, $t = k+1$, one can obtain system measurements and repeat the process.

Data-driven learning

All data-driven learning that necessitates the utilization of machine-learning and/or artificial intelligence is supported in the NEURON micro-service of the BACS^{2}aaS framework (see FU4 in Figure 1). Often, due to high computational loads, one would typically deploy the NEURON mircoservice on a platform that has support for GPU processing.

In the context of this work, we are concerned with using MPC as a model-based control strategy in FU2 to generate a receding horizon control law for controlling either: (i) some external process during the deployment of the BACS^{2}aaS framework; or/and (ii) controlling some high HFS during internal bootstrap learning (i.e., the model depicted by Figure 2 embedded in the BOPTEST framework in FU3).

The simplified internal model used in the MPC formulation (i.e, (1)) may exhibit significant plant-model mismatch when compared to the process models associated

with FU3, or the actual physics of a real physical building. A state estimator, in addition, is often combined with the MPC formulation to infer unknown (not measurable) process states subject to measurement noise and process uncertainties. The Kalman filter, considered an industry standard for state estimation (Auger et al., 2013), utilizes some linear model dynamics (i.e., (1)) and requires estimates to the process-, and measurement-covariance matrices. It has been noted that tuning/identifying the covariance matrices, however, may be time consuming and error prone.

Deep Neural Networks (DNN)'s are characterized by universality theorem which implies the ability of learning any function class in polytime (Abbe and Sandon, 2020). Training a DNN, however, to accurately identify function class mappings of real physical systems, using only a few representative samples, may at best be considered naive (Raissi et al., 2019). PINN's extends on DNN's by exploiting a-priori information of the underlying physical laws associated with process models to be learned. Here, physical laws can act as a neural network regularization agent to constrain the admissible space over which learning should be conducted.

To illustrate how model-based concepts within the MOSIOP micro-service, and data-driven learning strategies within the NEURON micro-service, can be facilitated in the proposed BACS^{2}aaS framework; we implement a PINN in the NEURON micro-service to infer initial process states for the MPC strategy, instead of using the more conventional Kalman filter. Motivation for using a PINN for state estimation is primarily to: (i) avoid the tuning of the Kalman filter; and (ii) to exploit the universality properties of the underlying DNN which, when conditioned with simplified physical laws of the process model (1), may generalize and learn non-linear associations present in the model depicted in Figure 2. Suppose the continuous process model of (1) is defined in the following compact ODE from:

$$\frac{dx}{dt} = f\left(x, u, r\right) \tag{5a}$$
$$y_k = h(x) + v \tag{5b}$$

where (5b) define a noisy process measurement ($v \sim \mathcal{N}(0, \sigma)$) observed at time $t = k$. Then, the objective here is to infer the initial condition x_0 in (4d) for time $t \in \mathbb{I}_{\geq 0}$ as new measurements y_k become available. Next, suppose the approximate solution to the differential equation (5a), at time $t = k$, can be evaluated by the following PINN trial function

$$\hat{x}_t := \bar{y}_k + t\mathbf{N}\left(t, y_k, \kappa_N(\hat{x}_{t-1}, r_k)\right) \tag{6}$$

For $n = 3$ states in (1), (6) defines $n-$trial functions with $\mathbf{N}(\cdot)$, in principle, being n-DNN functions (see Figure 4). The vector $\bar{y}_k$, for indexing $i \in \mathbb{I}_{1:n}$, is defined as

$$\bar{y}_k(i) := \begin{bmatrix} y_k(i) & \text{if } x(i) \text{ measured} \\ \hat{x}_{t-1}(i) & \text{else} \end{bmatrix} \tag{7}$$

In the context of training PINNs, let the gradient evaluation of (6) be $\frac{d\hat{x}_t}{dt}$ which serves as an approximation function for the system physics defined in (5) (Antonelo et al., 2021). We are subsequently interested in minimizing the following mean squared error lose term:

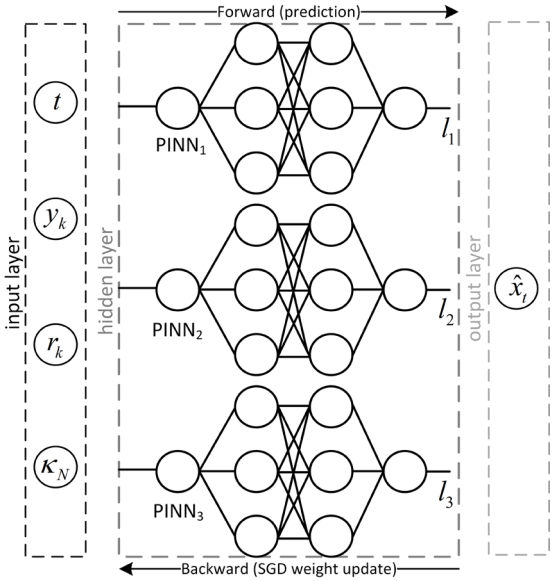

Figure 4: Input and hidden layers of the PINN formulation for inferring system states.

$$l_i := \left\| \frac{d\hat{x}_t}{dt}(i) - f\left(x, u, r\right)(i) \right\|^2 + \left\| C\hat{x}^t - y_k \right\|^2 \tag{8}$$

where we adopt $i \in \mathbb{I}_{1:n}$ to index the i^{th} ODE term in (1), and the i^{th} gradient PINN evaluations, respectively. In (8), we associate a loss term with each state where the latter includes the respective physical state dynamics to act as regularization agent of the network. A backward pass (evaluate a Stochastic Gradient Descent (SGD)) of this loss term allows updating the respective PINN weights.

Micro-service and data management

The BACS^{2}aaS framework needs to facilitate several communication endpoints, depending on the user. The management for information and data flow, and visualization, is overseen by FU1. We can differentiate between upstream endpoints (being developers and utilities) or downstream endpoints being third party API's sourcing information such as weather or pricing signals, or assets deployed in buildings (i.e., FutureHome access via MQTT (Hunkeler et al., 2008))

For developers, the BACS^{2}aaS framework integrates seamlessly with Visual Studio Code IDE (Microsoft, 2022) (remote container extensions enabled) which allows attaching to remote containers running respective micro-services. Third-party clients, on the other hand, can access BACS^{2}aaS via an internet browser where a locally deployed Flask server (Grinberg, 2018) will stream interactive visualizations by utilizing Plotly (Inc., 2015)

PostgreSQL (FU5), being a Time series databases (TSDBs), is deployed as a data-management solution. Both NEURON and MOSIOP micro-services interface with the database to log bootstrap learning results and historical closed-loop operation, respectively. In addition, informa-

tion mined from third party API's (via FU1) is stored to facilitate learning and help build informative forecast models (weather, load demands, pricing signals) to be incorporated in the MPC strategy running in MOSIOP.

Framework validation

To validate the proposed framework and demonstrate the effectiveness of combining physics-based modelling with data-driven learning, three simulation experiments have been carried out. These experiments utilizes all micro-services encapsulated within the BACS^{2}aaS framework (illustrated in Figure 2) with particular focus on validating closed-loop control and learning performance between FU2 (MOSIOP) and FU3 (BOPTEST). The primary aim of the experiments is to show that PINN can be used as an alternative to Kalman filtering to aid as an observer, with the benefit of avoiding the need for intrusive excitation experiments and/or empirically tuning the covariance matrices of the Kalman filter.

Experiment 1

For this case, MPC utilizes the Kalman filter for state estimation. The covariance matrices are empirically tuned to give reasonable performance. The MPC is run for k=1000 time steps, with $t_s = 900s$. The set-point of the interior temperature T_i is perturbed each $k = 200$ time steps, alternating between $22°C$ and $18°C$. The stage cost, as defined by 2, is weighted with coefficients $\alpha_1 = 1$ and $\alpha_1 = 1e - 5$, i.e., with a high priority weight on set-point tracking.

Experiment 2

In this case, MPC is combined with a PINN with the latter serving as an online state observer. Training of the PINN is partially done by using data previously generated from running the MPC in combination with the Kalman filter (i.e., Experiment 1). Motivation for this is to provide rich representative data-tuples $\hat{x}_t, y_k, r_k \kappa_N$ for training the network, also here considered partial supervised training.

Experiment 3

The last case entails combining MPC with a PINN as the state observer, however, the training data from the Kalman filter is made unavailable. This configuration may to some extent be considered unsupervised training of the PINN. In the experiments involving PINNs, the networks comprise of two hidden layers of 20 neurons each, with a ReLU activation function used for all neurons. The PINNs are trained with batches of 500 randomly sampled data points that was historically logged on FU5.

Results and discussion

Figure 5 shows the state estimates $\hat{x}_t$ obtained for experiment 1. It can be seen that the Kalman filter estimates the interior state T_i and T_e accurately, with the estimated state (solid grey line) closely following the true state (dashed grey line). However, we see that the tracking of the unmeasured states T_e and T_h does not come close to this performance. For the envelope state T_e, the estimation removes peaks and valleys, resulting in a flattened version of the true state. The estimation of the heater state appears similar, with the addition of an estimation bias leading the estimated value to be consistently underestimated for the duration of the experiment. As a result of the trouble the Kalman filter has in estimating the unmeasured states, the set-point tracking performance is not sufficient. Since the controller (MPC) only has an accurate estimate of the interior state T_i, which stores a small amount of energy compared to the envelope T_e state, it is not possible to track the supplied reference signal (solid black line) without the significant over- and undershoots that can be seen in the first subplot. In Figure 6, the results of experiment 2 are shown. Here, it can be seen that the estimation of the envelope temperature T_e is significantly more accurate than in the previous experiment. This leads to better set-point tracking, as the controller has a better overall state estimate. The estimation of the heater temperature T_h is still not very good, however, this state does not store much energy. Thus, it has less of an impact on the calculation of the optimal control action in the next step. One drawback of the setup in this experiment is that it requires training data from a Kalman filter to be present in first place. Thus, such a configuration cannot replace a Kalman filter entirely, but only after a certain amount of training data has been generated, which can only happen after the filter has been empirically tuned. However, it is worth noting that the performance of the Kalman filter in experiment 1 is not remarkable. Despite this less-than-ideal training data, the PINN is trained and able to give significantly better performance than the filter. Figure 7 shows the results of experiment 3. Since no training data is available in this experiment, and the selected batch size is 500, the PINN idle for 500 time steps before training starts. We observe a period of free-float for the estimation (where no estimation is performed) and only the measured temperature is used for the control. This can be seen by the large deviations incurred in the state estimates. After $k = 500$ time steps, PINN starts learning from sampling historic data samples. After a short period in which the estimated value of T_i drops very low, we see convergence of the estimated state and the true state $k = 800$ time steps. The set-point tracking performance stabilizes quickly after the PINN estimator is deployed, as the estimation of the envelope temperature is close to the true state initially. We see that after this initial delayed period of estimation, the unsupervised PINN performs better than the empirically tuned Kalman filter in experiment 1. Instead of providing the noise covariances explicitly and finding the optimal weighting between measurement and model, which is done in the Kalman filter, the noise is learned implicitly.

Concluding remarks

A prototype of the BACS^{2}aaS framework, comprising of 5 micro-services, intended to facilitate real-time model-based control through data-driven learning, and the DT concept, has been demonstrated. It has been shown by

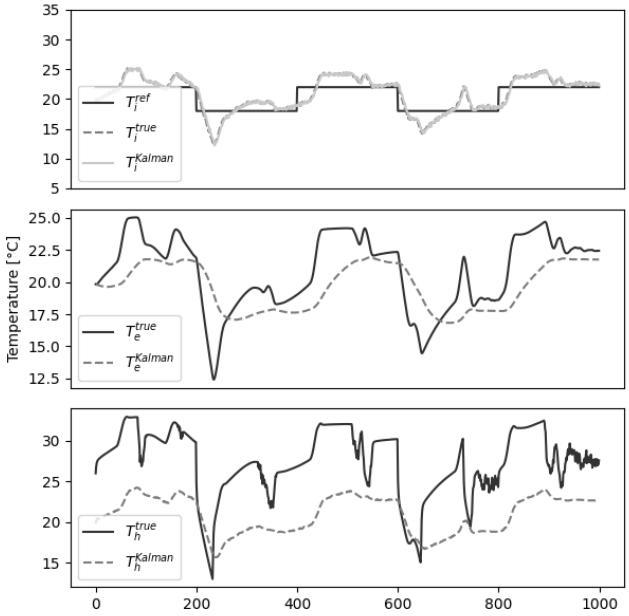

Figure 5: Using the Kalman filter as online observer for initializing the MPC strategy in FU2.

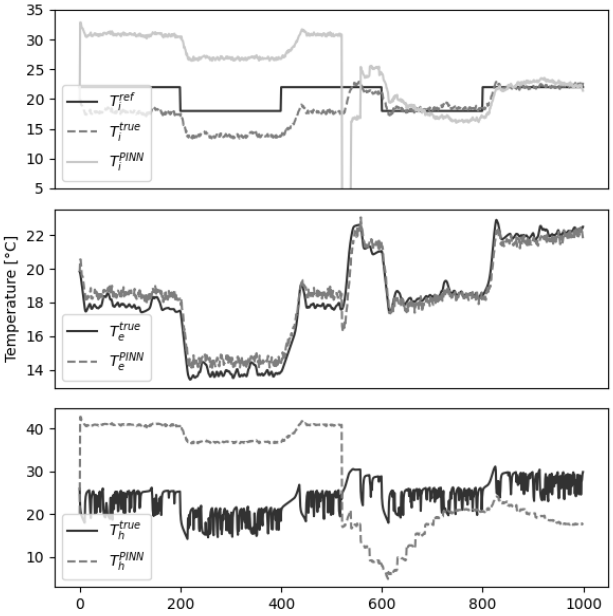

Figure 7: Using the PINN as a state observer in the context of unsupervised learning.

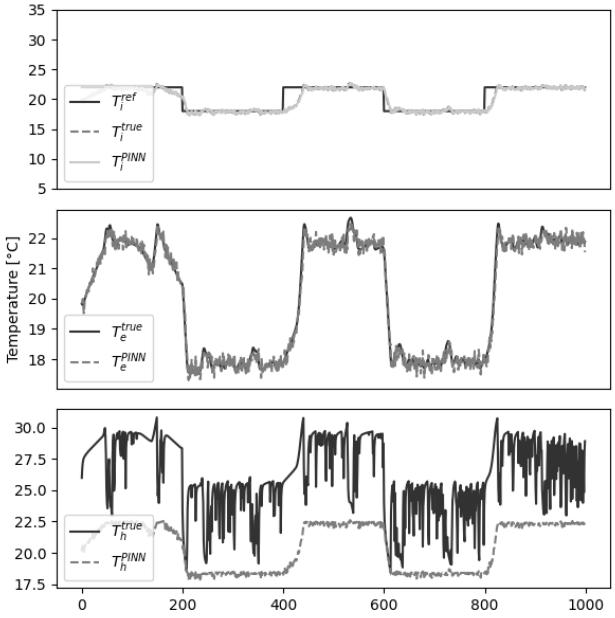

Figure 6: Training the PINN on historically generated data of the Kalman filter employed in experiment 1, and utilization of the former as state observer.

simulation that PINNs can easily be deployed in the proposed framework, and validated against model-based controllers and a HFS for improved learning and control performance. Further steps for development have been identified, and include in particular an emphasis on: (i) incorporating more support for high-fidelity emulators (whose structure can be exploited) (ii) combined state-parameter identification by utilizing concepts within machine learning (iii) interface against a real physical building. In the latter point, ongoing work is to stream data from a Future-Home device via the Futurehome IoT Messaging Protocol (FIMP) (Futurehome, 2021); and, build data-driven forecast models on energy consumption and user behaviour, to be further utilized in the model-based strategy encapsulated in FU3.

References

Abbe, E. and C. Sandon (2020). On the universality of deep learning. *Advances in Neural Information Processing Systems 33*, 20061–20072.

Anderson, C. (2015). Docker [software engineering]. *Ieee Software 32*(3), 102–c3.

Antonelo, E. A., E. Camponogara, L. O. Seman, E. R. de Souza, J. P. Jordanou, and J. F. Hubner (2021). Physics-informed neural nets for control of dynamical systems. *arXiv preprint arXiv:2104.02556*.

Arroyo, J., F. Spiessens, and L. Helsen (2022, May). Comparison of Model Complexities in Optimal Control Tested in a Real Thermally Activated Building System. *Buildings 12*(5), 539. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

Auger, F., M. Hilairet, J. M. Guerrero, E. Monmasson, T. Orlowska-Kowalska, and S. Katsura (2013). Industrial applications of the kalman filter: A review. *IEEE Transactions on Industrial Electronics 60*(12), 5458–5471.

Bacher, P. and H. Madsen (2011). Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings 43*(7), 1511–1522.

Blum, D., J. Arroyo, S. Huang, J. Drgoňa, F. Jorissen, H. T. Walnum, Y. Chen, K. Benne, D. Vrabie, M. Wetter, and L. Helsen (2021). Building optimization testing

framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings. *586-610*. Accepted: 2021-11-15T09:53:06Z Publisher: Taylor & Francis.

Blum, D. H., K. Arendt, L. Rivalin, M. A. Piette, M. Wetter, and C. T. Veje (2019, feb). Practical factors of envelope model setup and their effects on the performance of model predictive control for building heating, ventilating, and air conditioning systems. *Applied Energy 236*, 410–425.

Drgoňa, J., J. Arroyo, I. C. Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Ollé, J. Oravec, M. Wetter, D. L. Vrabie, et al. (2020). All you need to know about model predictive control for buildings. *Annual Reviews in Control*.

Drgoňa, J., J. Arroyo, I. Cupeiro Figueroa, D. Blum, K. Arendt, D. Kim, E. P. Ollé, J. Oravec, M. Wetter, D. L. Vrabie, and L. Helsen (2020, January). All you need to know about model predictive control for buildings. *Annual Reviews in Control 50*, 190–232.

Futurehome (2021). Local API access over MQTT (Beta). `https://support.futurehome.no/hc/en-no/articles/360033256491-Local-API-access-over-MQTT-Beta-`. [Online; accessed 4-May-2021].

Grinberg, M. (2018). *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Hensen, J. and R. Lamberts (edited by) (2019). *Building performance simulation for design and operation* (Second edition ed.). Abingdon, Oxon ; New York, NY: Routledge.

Hunkeler, U., H. L. Truong, and A. Stanford-Clark (2008, January). MQTT-S - A publish/subscribe protocol for Wireless Sensor Networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Bangalore, India, pp. 791–798. IEEE.

Inc., P. T. (2015). Collaborative data science.

Lu, Q., A. K. Parlikad, P. Woodall, X. Xie, Z. Liang, E. Konstantinou, J. Heaton, and J. Schooling (2019, October). Developing a dynamic digital twin at building and city levels: A case study of the West Cambridge campus. *Journal of Management in Engineering 36*.

Maree, J. P., S. Gros, and H. T. Walnum (2021). Adaptive control and identification for heating demand-response in buildings. In *2021 European Control Conference (ECC)*, pp. 1931–1936. IEEE.

Microsoft (2022). Visual Studio Code - Code Editing. Redefined.

Mohammed, C. M., S. R. Zeebaree, et al. (2021). Sufficient comparison among cloud computing services: Iaas, paas, and saas: A review. *International Journal of Science and Business 5*(2), 17–30.

Qin, S. J. and T. A. Badgwell (2003). A survey of industrial model predictive control technology. *Control engineering practice 11*(7), 733–764.

Raissi, M., P. Perdikaris, and G. E. Karniadakis (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics 378*, 686–707.

Rasheed, A., O. San, and T. Kvamsdal (2020). Digital Twin: Values, Challenges and Enablers From a Modeling Perspective. *IEEE Access 8*, 21980–22012. Conference Name: IEEE Access.