Zero-Shot Parameter Estimation of Modelica Models using Patch Transformer Networks

Ankush Chakrabarty[1] Marco Forgione[2] Dario Piga[2] Alberto Bemporad[3] Christopher Laughman[1]

[1]Mitsubishi Electric Research Labs (MERL), Cambridge, USA {chakrabarty, laughman}@merl.com
[2]SUPSI, IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland. {marco.forgione, dario.piga}@supsi.ch
[3]IMT School for Advanced Studies Lucca, Lucca, Italy. {alberto.bemporad}@imtlucca.it

Abstract

This paper introduces a transformer-based generative network for rapid parameter estimation of Modelica building models using simulation data from a Functional Mock-up Unit (FMU). Utilizing the `MixedAirCO2` model from the Modelica Buildings library, we simulate a single-zone mixed-air volume with detailed thermal and CO_2 dynamics. By varying eight physical parameters and randomizing occupancy profiles across 100 simulated systems, we generate a comprehensive dataset. The transformer encoder, informed by room temperature and CO_2 concentration outputs, predicts the underlying physical parameters with high accuracy and without re-tuning (hence, "zero-shot"). This approach eliminates the need for iterative optimization or can be used to warm-start such optimization-based approaches, enabling real-time control, monitoring, and fault detection in FMU-based workflows.

Keywords: transformer networks, dynamic simulation, system identification, model calibration, functional mockup interface, machine learning.

1 Introduction

Accurate modeling of building energy systems is critical for the design and operation of advanced control strategies such as model predictive control (MPC), fault detection, and digital twins (Zhan, Chakrabarty, et al. 2023). Physics-based modeling tools, such as those based on the Modelica language, offer a powerful framework for capturing the thermal, ventilation, and air quality dynamics of buildings through component-based system representations. However, the reliability of these models depends heavily on accurate parameter values, which can be challenging to determine in practice due to limited access to detailed building specifications, aging infrastructure, or changes in operational conditions over time (Chakrabarty et al. 2021; Wüllhorst et al. 2022).

Parameter estimation methods bridge this gap by calibrating model parameters to match observed output data. In the context of Modelica models exported as Functional Mock-up Units (FMUs) (Blochwitz et al. 2011), parameter estimation typically involves executing repeated simulations while adjusting model parameters to minimize the mismatch between simulation outputs and measured signals, such as temperature or CO_2 concentration. Classical approaches—including gradient-based optimization (Cañas et al. 2023), Bayesian calibration (Li et al. 2025), and black-box global optimization methods—have been successfully applied in this domain (Chakrabarty et al. 2021). Tools like RAPID (Vanfretti et al. 2016) provide modular ways to link external optimization routines to FMU simulations, supporting a range of problem formulations from gray-box modeling to model order reduction (Bres 2021). Despite their success, these methods can be computationally expensive, particularly when applied to large-scale simulation studies or real-time control tasks, due to their reliance on iterative optimization loops (Balali et al. 2023). Additionally, the authors of (Zhang and Mikelsons 2023) present a sensitivity-guided, iterative framework that combines BayesFlow and a physics-enhanced variational auto-encoder to improve parameter identification and data generation for model calibration. Their method enhances efficiency and accuracy by focusing on the most sensitive parameters during each iteration.

Herein, we study a fundamentally different paradigm: using transformer networks to learn the mapping from system outputs to model parameters directly from Modelica simulations. The idea is rooted in the principle that, for a well-defined family of parameterized models, there exists a functional relationship between the parameters and the corresponding output trajectories under typical excitation. By simulating a large number of such model instances offline, varying both physical parameters and disturbance conditions, we can train a transformer network to approximate the inverse mapping: from observed outputs back to the parameters that generated them. This multi-source parameter estimation has been investigated in the context of Modelica-based dynamical simulation only recently (Zhan, Wichern, et al. 2022; Li et al. 2025).

We focus on the use of transformer architectures, which have recently emerged as state-of-the-art tools for modeling sequential data in natural language processing, time-series forecasting, and symbolic regression (Vaswani et al. 2017). Transformer networks are a generative machine learning tool that excel at capturing long-range dependencies and temporal patterns, making them well-suited for analyzing

building simulation data, which often involves complex interactions between building materials and fluid flows, occupancy, and external weather conditions over extended time periods. Moreover, transformers have demonstrated impressive "zero-shot" capabilities: they often generalize well to entirely new forecasting or control tasks even without any additional fine-tuning. By leveraging pretraining on diverse datasets, they can transfer learned representations of temporal structure and physical dynamics to novel scenarios (such as predicting thermal loads in previously unseen building types or adapting to different climate conditions) simply by framing the problem in a compatible input format. This flexibility not only accelerates deployment in new environments but also reduces the need for costly labeled data when extending models to emerging use cases.

To evaluate our transformer-based approach to parameter estimation, we consider a simplified room-level building model implemented in Modelica and exported as an FMU. The model simulates the dynamics of air temperature and CO_2 concentration in a mixed-air volume, subject to external disturbances such as solar radiation and time-varying occupancy. Eight key physical parameters are varied across 2000 simulations: floor area, room height, room volume, external wall thermal resistance, window U-factor, window solar absorption fraction, exhaust air mass flow rate, and CO_2 emission rate per person. In each simulation, occupant presence schedules are also randomized to introduce realistic variability in internal gains. The output signals from each run—specifically, the room air temperature and CO_2 concentration—are collected at five-minute intervals for a full week, resulting in multichannel time series of high resolution. These serve as inputs to a patch-wise transformer encoder, followed by a multi-layer regression head that predicts the underlying parameter vector.

Compared to traditional parameter estimation methods, our approach offers two key advantages. First, it decouples the expensive optimization loop from the online estimation process. Once trained, the model can infer parameters from new output signals in a single forward pass, enabling rapid estimation that is suitable for time-sensitive applications such as adaptive control or live performance monitoring. Second, the neural network can leverage the full temporal structure of the input signals, including cross-channel correlations and delayed dependencies, which may be difficult to exploit effectively with handcrafted objective functions or scalar error metrics used in classical methods. While this approach does not replace traditional calibration frameworks, particularly in applications requiring safety-critical design, it offers a complementary tool for use cases where speed and scalability are paramount. Moreover, since the method is data-driven but trained entirely on simulated data, it mitigates the need for large amounts of real-world measurement data, making it practical in warm-starting optimization-based parameter estimation algorithms.

By integrating modern machine learning with established Modelica modeling workflows, this work aims to expand the toolkit available for building energy analysis, enabling faster, scalable, and data-informed modeling workflows for real-world systems. Specific contributions of this paper include: (i) proposing a transformer-based neural architecture for estimating physical parameters from building simulation output time series; (ii) demonstrating the effectiveness of zero-shot parameter estimation on a dataset generated using a Modelica FMU of a mixed-air room model, with randomized physical parameters and occupancy profiles, and (iii) empirical validation of the trained model, showing that it can accurately recover eight key physical building parameters that may vary as the building ages, or are difficult to measure, despite significant climate and occupant-induced variation.

The remainder of this paper is organized as follows. Section 2 formulates the parameter estimation problem and describes our FMU-based simulation model, including the `MixedAirCO2` benchmark, the procedure for exporting it as an FMU, and the generation of a large, randomized dataset for pretraining. Section 3 details the design of our patch-wise transformer encoder and regression head, including embedding, positional encoding, attention layers, and pooling strategy. Section 4 presents the experimental setup and results: we begin by assessing zero-shot estimation accuracy across a range of context lengths (:= sequences of measurement data collected for parameter estimation, to be used as contextual inputs for the generative network), then benchmark our transformer against non-transformer baselines to highlight the superiority of attention-based models in achieving robust zero-shot performance under varying data quantities. Section 5 summarizes our conclusions and outlines directions for future work, including extensions to probabilistic transformers for uncertainty quantification.

2 Motivation

2.1 Problem Statement

We consider a parameterized dynamical system represented by a Modelica-based Functional Mock-up Unit (FMU):

$$\mathbf{y}(t) = f(\mathbf{p}, \mathbf{d}(t), t), \tag{1}$$

where $\mathbf{p} \in \mathbb{R}^m$ is a vector of unknown physical parameters, $\mathbf{d}(t)$ denotes time-varying disturbances (e.g., occupancy-driven internal gains), and $\mathbf{y}(t) \in \mathbb{R}^n$ are the observable outputs over time such as indoor air temperature and CO_2 concentration. Our goal is to learn a parameter estimator $\hat{g}_\theta : \mathbb{R}^{T\times n} \to \mathbb{R}^m$ that maps output trajectories $\mathbf{y}_{1:T} = \{\mathbf{y}(t_1), \ldots, \mathbf{y}(t_T)\}$ to estimates of the underlying parameters. Formally, we write this as

$$\hat{\mathbf{p}} = \hat{g}_\theta(\mathbf{y}_{1:T}) \approx \mathbf{p}, \tag{2}$$

where θ denotes some parameterization of the map $\hat{g}$, e.g., weights of a deep neural network. We reiterate that unlike traditional optimization-based parameter estimation methods which solve an inverse problem for each new instance

DOI
10.3384/ecp218469

of contextual data $\mathbf{y}_{1:T}$, we propose to learn this mapping directly from synthetic data generated offline.

2.2 FMU-based Multi-Source Data Collection

We use an FMU extracted from the `Examples.MixedAirCO2` model from the Modelica `Buildings 11.0` library (Wetter et al. 2014) as the simulation model with default TMY3 file for meteorological data; see Figure 1 for the graphical layer representation in Dymola. This model represents a single thermal zone with a detailed energy and CO_2 balance, incorporating contributions from internal gains, air infiltration, ventilation, and weather-driven boundary conditions. The model includes thermal mass, surface convection and radiation, and moisture dynamics, making it: (i) representative of building simulation use cases, and (ii) easily extendable to multi-zone office buildings.

The zone is served by a mixing box that supplies outdoor air and recirculated return air based on a CO_2-based control strategy. To perform parameter estimation under realistic operating conditions, we utilize closed-loop data, where the PI controller remains active with gain constants carefully tuned to balance tracking performance and disturbance rejection. We also add small-gain sinusoids to the control action in order to provide some excitation for the inputs. We already know that the outputs are sufficiently sensitive to the persistently exciting inputs and will induce persistently excited outputs that we will use for learning. This configuration preserves feedback actuation, as evident from the Fig. 1, and allows the system's response to exogenous inputs (e.g., occupancy profiles, weather disturbances, and internal loads) to be interpreted in the context of both the building's physical properties and the control dynamics. This setup is not only realistic, but the identification problem is challenging because of varying occupancy profiles and weather conditions.

The physical parameters varied in each simulation are summarized in Table 1. Each parameter is sampled uniformly within the specified range. Each simulation spans 1 year of operation, sampled every 60 minutes (3600 seconds), resulting in $T = 8760$ time steps per simulation. The output signal at each time step is the room air temperature T and the CO_2 concentration: $\mathbf{y}(t) = \begin{bmatrix} T(t) & CO_2\,(t) \end{bmatrix}$.

Occupant-driven internal gains are also randomized by varying: (i) occupant count (persons), (ii) start time and duration of occupancy, (iii) daily schedules. However, occupant information is not provided to the model at inference time, reflecting practical limitations.

2.3 FMU Export Procedure

To enable parameter estimation using external (to Modelica) optimization tools, we export the `MixedAirCO2` model as a functional mock-up unit (FMU) using Dymola 2024 (Dassault Systèmes 2023). The following steps summarize the FMU export process:

1. CONTROL CONFIGURATION. We disable the internal PI CO_2 controller.

2. PARAMETER EXPOSURE. Key physical parameters targeted for estimation are re-declared as `input Real` variables within the Modelica model. This allows external modification during estimation without having to continuously recompile the FMU.

3. SOLVER AND TIME SETTINGS. Simulation is configured for a 1-year period with a storage step size of 1 hr to match the measurement data resolution. The CVODE solver (Cohen, Hindmarsh, Dubois, et al. 1996) is selected for robust handling of stiff system dynamics during FMU export and tolerances and solver step-sizes are set to default.

4. FMU GENERATION. The FMU is generated using Dymola's `Advanced` menu with the following settings:

   - FMI version: `2.0`
   - FMI type: `Model Exchange`
   - Solver: `CVODE`

5. VERIFICATION. The resulting FMU is validated using `fmpy`[1] in Python 3.11 to confirm correct loading, access to exposed parameters, and consistency of simulated outputs with the original Modelica model under default disturbance input profiles.

In summary, this export procedure ensures that the physics-oriented Modelica model is encapsulated in a reusable and differentiable FMU format, suitable for integration with external state-of-the-art deep learning tools for which widespread Modelica support at the time of writing remains limited.

2.4 Multi-Source Dataset and Dataloader

Figure 2 presents multiscale building simulation outputs: daily temperature cycles with subtle day-to-day shifts, sharp occupancy-driven internal-gain spikes, multi-day ambient-temperature fronts, and a nonlinear, saturating heating response. This highlights the complex disturbances and actuator dynamics that make parameter estimation challenging.

The collected dataset is split into training (80 systems with uniquely sampled parameters), validation (10 systems), and test sets (10 systems). Channel-wise normalization is applied based on the training data statistics. During evaluation, model accuracy is assessed by comparing predicted parameter vectors $\hat{\mathbf{p}}$ to ground truth vectors $\mathbf{p}$ over the test set. We found it easier to work with logarithmic transforms of the parameters than the parameters themselves due to their values showing varying orders of magnitude.

[1] https://github.com/CATIA-Systems/FMPy

Table 1. Physical parameters varied in simulation experiments. Each parameter is sampled uniformly within the specified range.

Parameter Name	Description	Unit	Range
`gaiCO2.k`	CO_2 emission rate per person	kg/s	$[6 \times 10^{-6},\ 1.2 \times 10^{-5}]$
`roo.AFlo`	Floor area	m^2	[60.0, 200.0]
`roo.hRoo`	Room height	m	[2.5, 4.0]
`roo.V`	Room volume	m^3	[40.0, 300.0]
`matExtWal.R`	External wall thermal resistance	$K \cdot m^2/W$	[0.5, 3.0]
`glaSys.UFra`	Window U-value	$W/m^2 \cdot K$	[1.0, 3.0]
`glaSys.absSolFra`	Window solar absorptance	–	[0.3, 0.7]
`mOut_flow`	Exhaust air mass flow rate	kg/s	[0.05, 0.5]

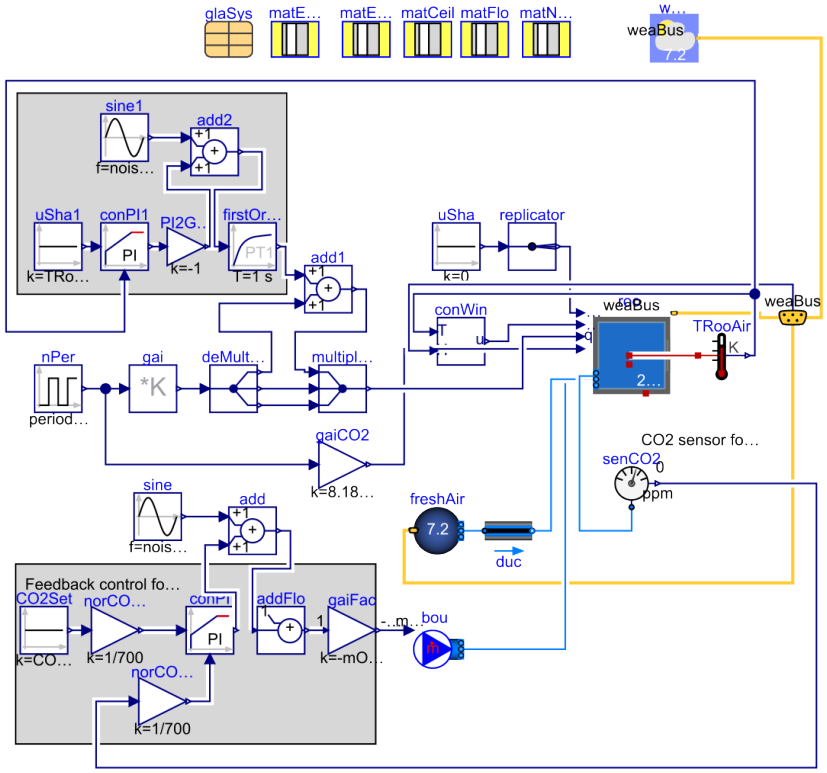

Figure 1. Graphical representation of adapted `MixedAirCO2.mo` model from Modelica `Buildings` Library, with additional PI control loop closed on room air temperature.

Data loading is implemented via `PyTorch 2.0`'s custom `Dataset` class that wraps the full collection of multivariate time-series records and their corresponding target vectors. Rather than returning entire sequences, the `Dataset`'s length is defined by a "virtual epoch" size, and each call to `getitem` randomly selects one series and then extracts a contiguous subsequence of fixed length from it. This subsequence is returned together with the associated target vector. By sampling both the series index and the time-window start position at each access, the `Dataset` generates a diverse set of training examples each epoch without explicit shuffling. We then wrap this `Dataset` in a standard `DataLoader` with a chosen batch size, yielding batches of subsequences and targets for efficient mini-batch training. Randomization inside the `Dataset` ensures variety in each batch.

More concretely, given contextualizing measurement inputs $\mathbf{Y} \in \mathbb{R}^{Q \times T \times n}$ and static outputs $\mathbf{P} \in \mathbb{R}^{Q \times m}$ across Q systems, each training sample is constructed as follows:

1. A system index $q \in \{1, \ldots, Q\}$ is drawn uniformly at random.
2. A start index $t_0 \in \{0, \ldots, T - L\}$ is sampled uniformly.
3. A temporal window $\mathbf{y}_{q,t_0:t_0+L-1} \in \mathbb{R}^{L \times n}$ is extracted.
4. The target is the corresponding output vector $\mathbf{p}_q \in \mathbb{R}^m$.

The dataset returns tuples $(\mathbf{Y}, \mathbf{P})$ with fixed window length L and configurable epoch size M.

DOI
10.3384/ecp218469

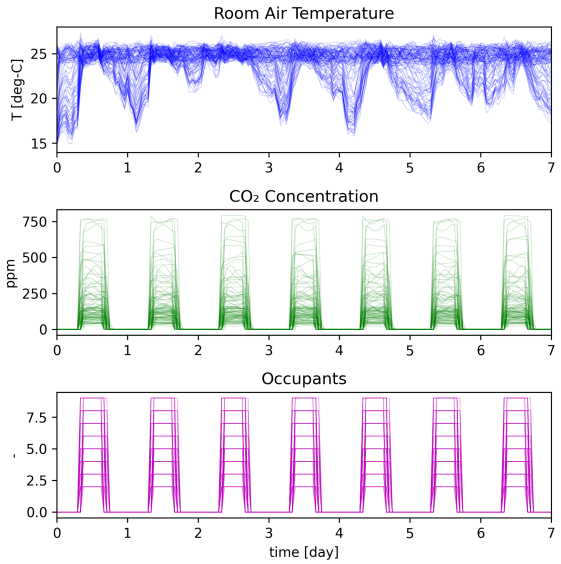

Figure 2. Example multi-system simulation outputs from FMU over one week; with loops closed on carbon dioxide and room air temperature. The outputs have been induced by time-varying disturbances, including fluctuating internal gains, multi-day ambient temperature fronts, and heating power responses.

3 Transformer-Based Parameter Estimator

We employ a patch-wise transformer encoder for parameter estimation from multivariate time series, followed by a deep multilayer perceptron (MLP) head for final prediction. The full architecture is referred to as the *patch transformer*, and is composed of five main components:

1. PATCH EMBEDDING. The input time series $\mathbf{Y} \in \mathbb{R}^{B\times T\times n}$ is separated into patches by a 1-D convolution: $\mathbf{z}_0 = \text{Conv1D}_{\text{stride}=\ell_p,\text{kernel}=\ell_p}(\mathbf{Y}^\top)$, where ℓ_p is the patch size and T is the sequence length. This transforms the input to $B \times (T/\ell_p) \times d$.

2. POSITIONAL ENCODING. A sinusoidal positional encoding $\mathbf{e} \in \mathbb{R}^{T/\ell_p \times d}$ is added to the patch embeddings, providing temporal awareness.

3. TRANSFORMER ENCODER BACKBONE. A stack of L standard transformer encoder layers as described in (Vaswani et al. 2017), each with H attention heads and feedforward width d_{ff}, processes the sequence $\mathbf{z}_L = \text{TransformerEnc}(\mathbf{z}_0 + \mathbf{e})$.

4. GLOBAL POOLING. A global average pooling layer aggregates the token outputs:

$$\mathbf{h} = \frac{1}{T/\ell_p} \sum_{i=1}^{T/\ell_p} \mathbf{z}_L^{(i)}.$$

5. REGRESSION HEAD. A MLP with LeakyReLU activations and LayerNorm at the input maps the pooled embedding to the target vector $\hat{\mathbf{p}} \in \mathbb{R}^{B\times m}$.

Our patch transformer is inspired by the Patcher model introduced in (Ou et al. 2022), which employs a patch-wise transformer encoder. Unlike the Patcher model, we replaced the mixture-of-experts decoder with a regression head, which is simply composed of fully connected layers.

The hyperparameters used in the patch transformer estimator are listed in Table 2. We did not perform extensive hyperparameter search, and expect that a smaller network will likely suffice if optimized by Bayesian neural architecture search (Nasrin et al. 2022) or an automated tool like Optuna (Akiba et al. 2019). All linear and convolutional layers are initialized using Xavier normal initialization (Glorot and Bengio 2010). Biases are zero-initialized.

Table 2. Patch Transformer Hyperparameters

Hyperparameter	**Value**
Input channels C	2
Patch size ℓ_p	12
Embedding dimension d	64
Number of Transformer layers L	4
Number of attention heads H	8
Feedforward dimension d_{ff}	256
Dropout	0.2
Output dimension d_{out}	8
Activation	LeakyReLU(α=0.2)
Positional Encoding	Sinusoidal (fixed)
Pooling	Global average pooling

The patch transformer parameter estimator combines localized feature extraction and global sequence modeling, which is critical for capturing the complex temporal dynamics inherent in multivariate time series from building systems. Patch embedding via 1D convolution reduces sequence length and emphasizes local temporal correlations, while the transformer encoder captures long-range dependencies using self-attention, avoiding the limitations of recurrence-based methods. The use of sinusoidal positional encoding preserves temporal order, and global average pooling enables efficient dimensionality reduction and regularization. A deep MLP regressor maps the aggregated latent representation to parameter estimates, allowing flexible modeling of non-linear, non-convex mappings typical in grey-box system identification.

Compared to alternatives, this architecture provides several advantages. Unlike RNN-based models (e.g., LSTM, GRU), it enables full parallelization during training and better handles long-range dependencies without gradient degradation. CNN-only approaches lack adaptability to temporal variation, and MLPs trained on raw sequences fail to encode temporal structure. The transformer's ability to dynamically attend across patches, combined with the efficiency of patch-wise tokenization, makes it more expressive and scalable. This inductive bias, along with its modular structure, is expected to offer improved accuracy and robustness over conventional baselines for zero-shot parameter estimation in building energy models.

4 Results and Discussion

All experiments were conducted on a MacBook Pro equipped with an Apple M3-Max system-on-a-chip featuring 16 cores (12 performance cores and 4 efficiency cores) and 64 GB of unified memory. Training on CPU for the patch transformer required between 1-2h. The Adamax optimizer is used for training with a learning rate of 0.001 for 2000 training iterations, the batch size is $B = 16$.

4.1 Test Setup

We evaluate the trained transformer regressor on held-out time–series snippets drawn from the same one-year simulation of 100 independent systems. Each snippet spans a fixed "context window" of length L (in hours), chosen from $\{24, 48, 96, 168, 360, 720\}$. For each L:

1. We randomly sample $N_{\text{win}} = 1000$ disjoint windows of length L from the full dataset, forming a *test* subset.

2. The network (whose inputs had been normalized and whose outputs were predicted in base-10 logarithmic scale) is applied in inference mode to each window, yielding $\widehat{\mathbf{p}}_{\log 10}$. We recover physical parameter estimates by
$$\widehat{\mathbf{p}} = 10^{\widehat{\mathbf{p}}_{\log 10}}.$$

3. The true parameter vector $\mathbf{p}$ for each system is likewise converted from $\log_{10}$ back to the original units.

4. We compute two metrics over the test set:

   - The coefficient of determination for each parameter
   $$\mathsf{R}^2 = 1 - \frac{\sum_{i=1}^{N_{\text{win}}} \left(p_i - \widehat{p}_i\right)^2}{\sum_{i=1}^{N_{\text{win}}} \left(p_i - \bar{p}\right)^2},$$
   where $\bar{p}$ is the sample mean of the true values.
   - The relative accuracy for each estimate,
   $$\mathsf{A}_i = 1 - \frac{\left|p_i - \widehat{p}_i\right|}{p_i}, \tag{3}$$
   which is 1 for a perfect match and is smaller as the absolute relative error grows.

4.2 Patch Transformer Results

In Fig. 3, we test the quality of the outputs generated by the FMU over 21 days. We first use 7 days as an input to the transformer model, as illustrated by the blue trace, and from this infer a set of predicted parameters for the model. Note that for this test we switch off persistently exciting noise on the controls to promote realism. By simulating with the predicted parameters for the 14 days after the 7 days of context input, we see that the error in the predictions is small, as is evident by the overlap between the orange dashed predicted trajectory and the blue solid true trajectory. Figure 4 presents scatter plots comparing predicted versus true values for eight key parameters of a thermal building model under varied contextual inputs. Each subplot corresponds to a distinct physical parameter. Red dots denote individual predictions, while the solid black line indicates the ideal identity mapping (i.e., perfect prediction). Each subplot reports the coefficient of determination (R^2) to quantify regression accuracy. The model demonstrates excellent predictive performance ($\mathsf{R}^2 \geq 0.95$) across most parameters, with only a modest drop in performance for the room height. The relatively lower score likely arises from its weaker influence on the observable outputs used for inference. Unlike parameters such as floor area or ventilation mass flow rate, which directly impact air volume exchange and thermal load profiles, room height has a more indirect effect, primarily through its contribution to overall room volume and stratification effects, which may not be strongly expressed in the measured outputs (e.g., temperature, CO_2 levels) under the simulation conditions. Additionally, identifiability may be confounded when both area and height jointly influence volume, making it difficult for the model to disentangle their individual contributions. This leads to a higher spread in predicted values for room height and hence a lower R^2 score. Overall, these results validate the model's high-fidelity parameter inference capability across diverse building configurations.

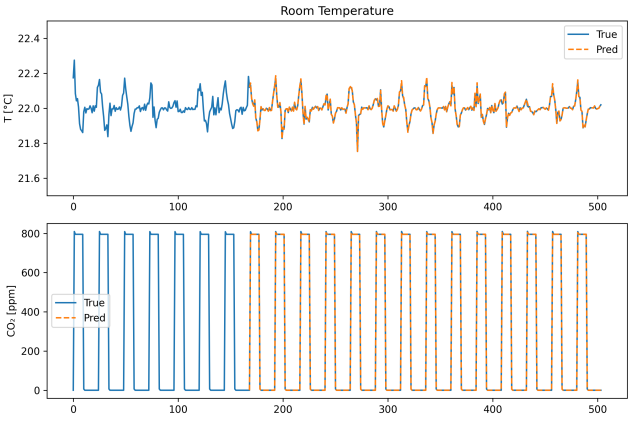

Figure 3. Predicted (orange) vs. true (blue) simulated outputs with a 7-day context set of measurements, and 14-day predictions.

Figure 5 presents boxplots of relative accuracy (3) for each parameter, evaluated over context windows of 1, 2, 4, 7, 15, and 30 days. Each box illustrates the interquartile range of relative accuracy across all test samples, with the median indicated by a horizontal line. As context length increases, inference accuracy improves consistently across all parameters, with median values approaching 1.0 and reduced variability. This trend is especially pronounced between 1 and 7 days, after which gains begin to saturate. Parameters such as exhaust mass flow rate (`mout.flow`) and CO_2 generation rate (`gaiCO2`) exhibit high identifiability even with short contexts, reflecting their strong observability through direct influence on air quality dynamics. In contrast, geometric and envelope-related parame-

DOI
10.3384/ecp218469

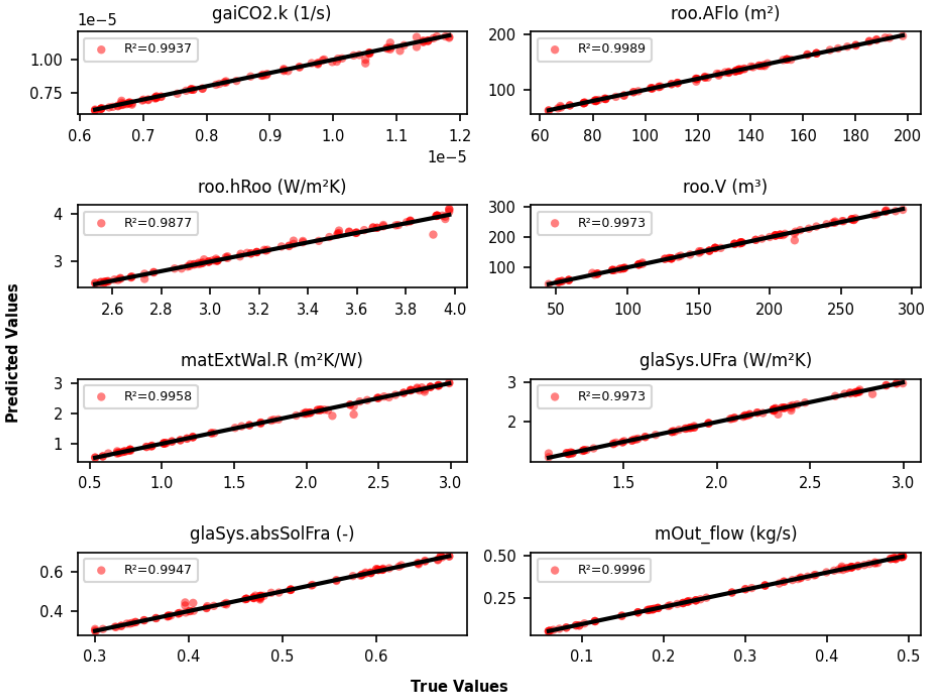

Figure 4. R^2 scatter plots (true vs. predicted) for each parameter, with 48 hr context windows.

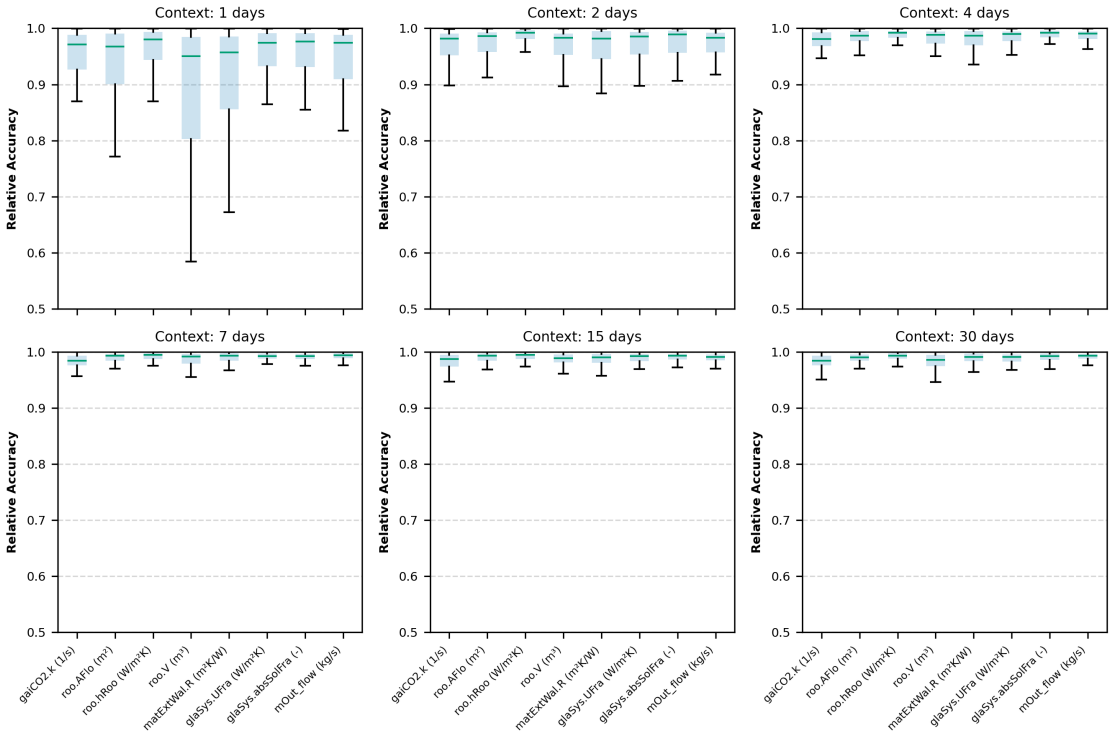

Figure 5. Boxplots of relative accuracy (3) versus context length.

ters (e.g., wall resistance `matExWal.R`, window U-factor `glsys.Ufrc`) benefit more from longer observation periods due to their slower thermal dynamics, indirect influence on measurable outputs, and entanglement with other physical properties. These factors lead to broader variance and reduced accuracy under short contexts. A 7-day context window offers a practical trade-off: it is short enough for deployment during commissioning while long enough to recover key physical parameters with high fidelity. This balance maximizes data efficiency and supports robust ini-

tialization of physics-based optimization routines in real buildings.

Table 3. Per-parameter R^2 on the test set for different transformer patch sizes for $N_{\text{win}} = 100$ parameter estimates.

Parameter	Patch 4	Patch 12	Patch 24
`gaiCO2.k`	0.9977	0.9947	0.9888
`roo.AFlo`	0.9989	0.9985	0.9975
`roo.hRoo`	0.9802	0.9979	0.9698
`roo.V`	0.9966	0.9982	0.9942
`matExtWal.R`	0.9850	0.9991	0.9913
`glaSys.UFra`	0.9950	0.9976	0.9931
`glaSys.absSolFra`	0.9976	0.9983	0.9949
`mOut_flow`	0.9997	0.9997	0.9997

In Table 3, we perform an ablation on the patch size of the proposed patch transformer; 1000 inputs are tested and the corresponding R^2 score is calculated. Overall, the network seems to be quite robust to patch-size selection. All three patch sizes deliver excellent predictive accuracy, with every parameter achieving $R^2 > 0.9$. The best performance in terms of average R^2 score across the paramaters is found with a patch size of 12, indicating it best balances capturing both fine-scale fluctuations and longer-range dependencies. Make the patch size smaller seems to improve estimates on more sensitive parameters such as `mOut_flow` and `gaiCO2.k` but shows poorer performance on the slower thermal dynamics (e.g. `matExtWal.R`). This trade-off suggests that very small patches lack sufficient context for slowly changing parameters, while very large patches dilute high-frequency signals. Note that a patch size of 12/24 corresponds to a 12h/24h window, which neatly aligns with the dominant diurnal cycles in building thermal and occupancy patterns. Smaller patches (4h) capture only very local fluctuations, leading to weaker performance. Consequently, a 12/24 patch size is recommended for uniformly high fidelity across all eight physical parameters.

4.3 Performance Comparison

We compare our patch transformer approach to two well-established time-series models: LSTM (Long Short-Term Memory) and TCN (Temporal Convolutional Network).

LSTM is a recurrent neural network variant designed to capture long-term dependencies in sequential data (Hochreiter and Schmidhuber 1997). It remains a common baseline for temporal modeling tasks due to its ability to handle variable-length inputs and learn from sequences with temporal correlations. We include LSTM as a representative of RNN-based architectures, which are widely adopted in forecasting and control settings.

In comparison, TCN is a convolutional alternative to RNNs that uses dilated causal convolutions to model temporal dependencies over long horizons (Lea et al. 2017). TCNs are attractive for their parallelizability, stable gradients, and competitive performance across time-series benchmarks. Their ability to capture hierarchical temporal structure without recurrence makes them suitable non-transformer baselines.

Table 4 summarizes the R^2 scores (mean and standard deviation) across models for varying context lengths. Overall, the Patch Transformer consistently outperforms both LSTM and TCN at short and medium horizons, validating its architectural design and inductive biases for learning from fine-grained temporal patterns.

Table 4. Performance comparison with TCN and LSTM.

Context	Model	Mean R^2	Stdev R^2
1 day	Our Transformer	0.8694	0.0818
	LSTM	-0.0284	0.4489
	TCN	0.0327	0.9633
3 days	Our Transformer	0.9679	0.0263
	LSTM	0.4266	0.2208
	TCN	0.4784	0.2382
7 days	Our Transformer	0.9669	0.0222
	LSTM	0.9664	0.0221
	TCN	0.6244	0.1410

At a short context length of 1 day, the transformer yields a mean R^2 of 0.8694, significantly outperforming LSTM and TCN, both of which under-perform with lower or even negative R^2 values. This highlights the transformer's superior capacity for modeling local dependencies via learned attention over patched segment: often, neither RNNs nor standard convolutions excel at this without larger receptive fields. At a medium context length (3 days), the performance gap improves, with our patch transformer achieving 0.9679, which is over double that of LSTM and TCN. This suggests that as the context window increases modestly, the transformer is uniquely capable of integrating more history without degradation, due to its global receptive field and ability to capture complex inter-dependencies. For a long context (7 days), both the transformer and LSTM perform similarly well, but the transformer's slightly higher mean and lower standard deviation suggest more stable generalization across samples. Notably, TCN remains behind, despite its design for long-range patterns, likely due to the difficulty of effectively tuning dilation rates and convolutional depths for diverse input dynamics.

Overall, our patch transformer-based regressor consistently outperforms or ties the strongest recurrent baseline across all tested window sizes. Its patch embeddings and self-attention mechanism enable more efficient contextual aggregation, particularly in low-data regimes (short windows), justifying its adoption for parameter regression tasks where context length may vary.

5 Conclusions

In general, the process of calibrating the parameters of physics-based models to experimental data often requires a large number of simulations to iteratively refine the parameters so that the model response matches the observations. Moreover, the number of simulations needed typically increases very quickly with the number of parameters being

DOI
10.3384/ecp218469

estimated, making this process quite time consuming. In comparison, this transformer-based approach enables the generation of parameter estimates directly from a set of experimental data in a zero-shot manner without the need for expensive iteration. This method uses offline simulation effectively in a massively parallel context during the training period to shift the burden of simulation from the time after data is collected to the time after the model is constructed. As engineers often desire information as soon as possible once experimental data is in hand, this is an attractive shift.

Furthermore, the proposed method shows strong promise as a warm-start initialization method for gradient-based parameter calibration algorithms when there is a significant simulation-to-real gap or further refinement of the values generated by the transformer-based method is desired. In many physical modeling settings, accurate initial guesses are critical for successful convergence of non-convex optimization routines. However, deriving such guesses analytically or heuristically is often infeasible. The transformer network helps fill this gap by rapidly providing good initial estimates that improve robustness and reduce computational overhead in subsequent calibration tasks.

We also demonstrate that this transformer-based architectures generalizes more effectively than conventional LSTM and TCN models when estimating system parameters from time-series data, particularly under conditions with limited temporal context. By leveraging the self-attention mechanism, the transformer can infer model parameters using significantly shorter time windows, thereby overcoming a longstanding limitation in traditional calibration tasks which typically require extensive input data for accurate estimation. This is especially beneficial for applications where measurement data is sparse or costly to collect.

As a promising direction for future work, we propose extending this approach to probabilistic transformers. This would allow the model to not only predict point estimates but also provide uncertainty quantification in the form of distributions over possible parameter values. Such capability could be instrumental in probabilistic calibration frameworks, for which prior distributions over parameters are required; e.g. Bayesian calibration.

References

Akiba, Takuya et al. (2019). “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631.

Balali, Yasaman et al. (2023). “Energy modelling and control of building heating and cooling systems with data-driven and hybrid models—A review”. In: *Renewable and Sustainable Energy Reviews* 183, p. 113496.

Blochwitz, Torsten et al. (2011). “The functional mockup interface for tool independent exchange of simulation models”. In: *Proceedings of the 8th international Modelica conference*. Linköping University Press, pp. 105–114.

Bres, Aurelien (2021). “Dimensionality reduction for calibration of dynamic building simulation models”. In: *Building Simulation 2021*. Vol. 17. IBPSA, pp. 1991–1998.

Cañas, Carlos Durán et al. (2023). “Parameter Estimation of Modelica Building Models Using CasADi”. In: *Modelica Conferences*, pp. 301–310.

Chakrabarty, Ankush et al. (2021). “Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics”. In: *Energy and Buildings* 253, p. 111460.

Cohen, Scott D, Alan C Hindmarsh, Paul F Dubois, et al. (1996). “CVODE, a stiff/nonstiff ODE solver in C”. In: *Computers in physics* 10.2, pp. 138–143.

Dassault Systèmes (2023). *Dymola*. Version 2024x. Dassault Systèmes. URL: https://www.3ds.com/products-services/catia/technologies/dymola/.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 249–256.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Lea, Colin et al. (2017). “Temporal convolutional networks for action segmentation and detection”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165.

Li, Yicheng et al. (2025). “A framework for calibrating and validating an HVAC system in Modelica”. In: *Journal of Building Performance Simulation*, pp. 1–21.

Nasrin, Shamma et al. (2022). “ENOS: Energy-aware network operator search in deep neural networks”. In: *IEEE Access* 10, pp. 81447–81457.

Ou, Yanglan et al. (2022). “Patcher: Patch transformers with mixture of experts for precise medical image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 475–484.

Vanfretti, Luigi et al. (2016). “RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models”. In: *SoftwareX* 5, pp. 144–149.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., pp. 6000–6010. ISBN: 9781510860964.

Wetter, Michael et al. (2014). “Modelica Buildings library”. In: *Journal of Building Performance Simulation* 7.4, pp. 253–270.

Wüllhorst, Fabian et al. (2022). “AixCaliBuHA: Automated calibration of building and HVAC systems”. In: *Journal of Open Source Software* 7.72, p. 3861.

Zhan, Sicheng, Ankush Chakrabarty, et al. (2023). “A virtual testbed for robust and reproducible calibration of building energy simulation models”. In: *Building Simulation 2023*. Vol. 18. IBPSA, pp. 3491–3498.

Zhan, Sicheng, Gordon Wichern, et al. (2022). “Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization”. In: *Energy and Buildings* 270, p. 112278.

Zhang, Yi and Lars Mikelsons (2023). “Sensitivity-guided iterative parameter identification and data generation with BayesFlow and PELS-VAE for model calibration”. In: *Advanced Modeling and Simulation in Engineering Sciences* 10.1, p. 9.