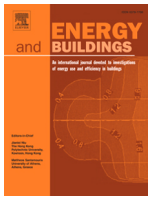
Energy & Buildings 369 (2026) 118025



Contents lists available at ScienceDirect

Energy & Buildings

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





AutoControl-MPC: An automated model predictive control development framework based on large language models for HVAC systems

Ziqi Hu [a,b], Mingchen Li [a,b], Xu Han [c], Lunlong Li [a,b], Zhe Wang [a,b,*]

[a] State Key Laboratory of Climate Resilience for Coastal Cities (SKLCRCC), Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Hong Kong, China
[b] HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen, China
[c] School of Architecture, University of Notre Dame, Notre Dame, 46556, Indiana, USA

ARTICLE INFO

Keywords:
Large language models
Model predictive control
Prompt engineering
In-context learning
Heating, ventilation, and air conditioning system
Generative artificial intelligence

ABSTRACT

Heating, Ventilation, and Air Conditioning (HVAC) systems are responsible for a significant portion of global energy consumption, making their efficient management crucial for reducing energy use and carbon emissions, as well as maintaining indoor comfort. Model Predictive Control (MPC) is widely recognized as an effective approach for HVAC control due to its superior ability to handle the dynamics and uncertainty of building energy systems. However, traditional development of an MPC controller is labor-intensive and requires specialized domain knowledge. Different building environments require customized model identification and code implementation, which limits its wide adoption at scale in practice. To address this challenge, this paper proposes AutoControl-MPC, an end-to-end automated workflow for MPC development using large language models (LLMs). At each stage of the MPC development process, AutoControl-MPC establishes a fully automated workflow. This workflow employs a template-driven automation pipeline where LLMs follow deterministic decision rules to parse semantic information, identify system models, and generate control code. The proposed AutoControl-MPC framework is evaluated in two test cases with different building configurations (i.e., a residential building and a commercial building) on the BOPTest virtual testbed. The experimental results show that AutoControl-MPC achieves a comparable and even better control performance compared to expert-developed MPC controller, demonstrating the effectiveness of the proposed framework and shedding light on the potential of LLMs in automating control development for complex building energy systems.

1. Introduction

Building energy consumption accounts for a significant portion of global total energy demand and addressing the energy efficiency issues in buildings has attracted extensive attention for achieving carbon neutrality and sustainable building environments [1]. Heating, Ventilation, and Air Conditioning (HVAC) systems are major energy consumers [2], where the effective control strategies are urgent for reducing the energy usage and maintaining the occupant comfort. Traditional Rule-Based Control (RBC) and Proportional-Integral (PI) Control are widely employed in the industry due to their simplicity and ease of implementation. However, these methods are often insufficient to address the complexity and uncertainties exhibited by modern building energy systems [3]. Controllers such as RBC and PI control adopt passive control strategies without the capability to predict future system states, often resulting in suboptimal performance. On the other hand, these traditional control methods are typically designed for single-objective optimization, such as minimizing energy consumption or maintaining thermal comfort [4]. However, the trade-off between multiple objectives is one of the tasks that must be considered in real-world building energy systems [5].

To address these challenges, Model Predictive Control (MPC) has been widely recognized as a superior approach compared to the traditional ones. Unlike reactive controllers, MPC leverages a dynamic system model—derived from either physics-based principles or data-driven identification—to explicitly predict future system evolution over a receding horizon. At each control horizon, a constrained optimization problem is formulated and solved by the MPC, solving the optimal control action for the next time step, which minimizes a predefined cost objective [3]. This optimization process is iteratively repeated at each control step with updated state feedback. This predict-then-optimize framework inherently addresses the multi-objective optimization problems and constraints that traditional methods cannot effectively manage [6].

* Corresponding author.
E-mail addresses: zhubg@connect.ust.hk (Z. Hu), mingchen.li@connect.ust.hk (M. Li), xhan7@nd.edu (X. Han), llidx@connect.ust.hk (L. Li), cezhewang@ust.hk (Z. Wang).

https://doi.org/10.1016/j.enbuild.2026.118025
Received 6 May 2026; Received in revised form 12 July 2026; Accepted 28 July 2026
Available online 29 July 2026
0378-7788/© 2026 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

However, the practical deployment of MPC faces significant challenges, particularly regarding its dependence on accurate system modeling and identification [7]. Establishing system models is expertise-demanding and requires extensive labor efforts, which becomes the major bottleneck for the widespread application of MPC in real-world scenarios [8].

Recent advances in Large Language Models (LLMs) have provided a potential solution to address the challenges of expertise dependency and labor intensity in development of MPC controllers. The embedded domain knowledge, semantic parsing, and code-generation proficiency of LLMs facilitates seamless integration of them into the control design process for building energy systems. Recent studies, such as AutoControl [9], have demonstrated that an end-to-end workflow for PI controller design can be fully automated. By leveraging building semantic information, such approaches successfully lower the development barrier and validate the feasibility of automatic controller generation for HVAC systems. However, previous efforts have been primarily limited to PI controllers, which are restricted to simple, low-level control tasks. A significant research gap remains in utilizing LLMs to construct more advanced control strategies, such as MPC. Unlike PI control design, MPC involves the complex challenges of automated physical modeling, specifically regarding actuator mechanisms and the thermal dynamics of operational zones. Furthermore, it requires the adaptive selection of forecasting models by the LLM—a decision-making process that has traditionally relied on the implicit heuristics of human experts.

To address the research gaps mentioned above, this paper proposes AutoControl-MPC, an end-to-end workflow that can automatically generate MPC for HVAC system control. AutoControl-MPC utilizes the Brick model of the building energy system as the semantic knowledge base, which contains key information of system variables for MPC design. The semantic information is automatically interpreted and processed using few-shot and chain-of-thought techniques. In the traditional design process, the relationship and properties of system variables are manually processed by human experts from building documents which requires significant time and effort, especially those that are not well organized. In contrast, the proposed workflow achieves automated data processing without human intervention. After obtaining the system information, AutoControl-MPC leverages LLMs to automatically model the essential actuator (e.g., heat pump) of the HVAC system and the thermal dynamics of the operational zone. For the thermodynamic model, an auto-selection mechanism driven by a multi-criteria decision-making protocol (balancing model fidelity and complexity) is proposed to determine the optimal model for MPC to achieve the best control results. Finally, a rule-based diagnostic self-validation loop is introduced to verify the control performance of AutoControl-MPC on the specific operational zone based on LLMs. The main contributions of this work are summarized as follows:

- An end-to-end workflow AutoControl-MPC is proposed to automatically design the MPC controller for HVAC systems based on LLMs. The building semantic metadata is automatically translated into fully formulated MPC optimization problems, which provides a generalized approach to mitigate the labor-intensive work, enabling the rapid and automated development of advanced control algorithms in highly heterogeneous building environments.
- An LLM-based system identification framework is formalized to capture the thermodynamic behavior of operational zones. Driven by a deterministic multi-criteria evaluation protocol, this approach systematically balances predictive accuracy with structural complexity, yielding an optimal low-order Resistor-Capacitor (RC) thermal network tailored to specific environments.
- A systematic methodology is established to derive the physical dynamics of HVAC actuators utilizing lagged cross-correlation for dominant variable identification and multi-hypothesis symbolic regression for non-linear modeling. By integrating these derived actuator models with the optimized RC model, the framework autonomously constructs rigorous MPC optimization problems, followed by a deterministic self-verification protocol to strictly bound the control logic within real-world safety limits.
- Experiments conducted on the Building Optimization Performance Test Framework (BOPTest) validate the feasibility of AutoControl-MPC with different building and HVAC system configurations. The results demonstrate comparable control performance of the auto-generated MPC controller compared with the expert-designed MPC controller.

2. Related work

2.1. Model predictive control in HVAC systems

Traditional RBC and PI control rely on reactive feedback mechanisms, which often lead to sub-optimal solutions when facing dynamic environment changes or conflicting control objectives [10]. To address these limitations, MPC has been widely adopted as a superior solution that solves an optimization problem at each control horizon based on forecasting system states. This predictive capability enables MPC to effectively handle the system constraints and interrelated variables, achieving a better balance between energy minimization and occupant thermal comfort compared to conventional methods [11]. Zheng et al. [12] utilized data-driven and physics-based MPC to manage the operations of the heat pump and battery energy storage system in residential buildings, which achieved a well-balanced trade-off between thermal comfort and energy consumption. In this work, the forecasting models in two MPC need to be pre-trained and calibrated in the simulation testbed, which requires extensive manual and expertise-demanding effort before extending the controller to other building environments or real-world scenarios.

In the traditional MPC of HVAC control, in order to take account of the multiple objectives, weighted objective function and Pareto front are usually applied as a practical solution. Yang et al. [13] proposed a scalable multi-objective MPC framework that utilized a genetic algorithm (NSGA-II) to dynamically generate a Pareto front representing the trade-off between occupant thermal comfort and grid-based CO_2 intensity. The results demonstrated a significant reduction in CO_2 emission by 26.63% while maintaining indoor comfort. However, such multi-objective optimization lacks flexibility handling different building environments. Once the terminal and equipment have changed, the objectives need to be re-designed to adopt the new system configurations. In some existing studies, data-driven MPC has emerged as an effective approach to address the problem of manually calibrating the physical models in HVAC systems. Zhao et al. [14] proposed a data-driven MPC framework that employed symbolic regression to model the thermal dynamics using built-in sensors, thereby effectively reducing the complexity of calibrating parameters required by traditional models. The results showed that the proposed framework can reduce the total energy consumption by up to 30.2% and peak power demand by up to 35.1% compared to traditional rule-based methods. However, the data-driven MPC is applied as a black-box method which lacks physical interpretability and requires significant historical data to train the model.

2.2. Semantic modeling in smart buildings

The labor-intensive process of parsing the system configuration is one of the major bottlenecks of the development of advanced building control, where engineers must rely on segmented technical manuals to identify specific sensor and actuator properties [15,16]. To address this problem, semantic modeling methods, such as Building Information Modeling (BIM) [17] and the Brick Schema [18], have been introduced to provide a standardized, machine-readable representation of building assets and their topological relationships. By structuring this metadata, semantic models serve as a digital knowledge base that resolves data heterogeneity, providing the interpreted component information for control systems without manual intervention. Furthermore, recent research

has expanded the scope of the schema beyond physical assets. Luo et al. [19] introduced an occupant-centric extension to Brick that standardizes behavioral patterns and preferences. This method enabled the semantic representation of complex human-building interactions, further enriching the knowledge base available for control strategies.

To further address the challenges of using semantic models for building automation, semantic web technologies have served as an effective approach for knowledge representation and automated reasoning. Delgoshaei et al. [20] proposed a framework that combines domain ontologies with rule-based inference to derive operational insights, such as identifying system states and diagnosing faults. The results of real-world case studies showed that this extension effectively captures granular occupant metadata—such as movement trajectories and thermal preferences—which were previously unrepresentable in standard schema. This framework demonstrated the capability of semantic models to support actionable decision-making beyond mere metadata representation.

In addition to the operational applications, significant progress has been made in bridging semantic models with physics-based simulation. Chen et al. [21] developed the AutoBPS-BIM toolkit, which automates the translation of BIM data (IFC) into Building Energy Models (BEM) for rapid load calculation and chiller optimization. Experiments in an eight-story office building showed that this toolkit reduced the entire modeling and calculation process to under 10 minutes and facilitated a chiller design optimization that achieved 15.3% energy savings compared to the baseline. This work illustrated the potential of leveraging standardized semantic information to drastically reduce the manual effort required for constructing dynamic simulation environments. While current methods can successfully map semantic data to high-fidelity Building Energy Models (BEM) for performance assessment, the existing semantic schema fail to explicitly characterize the control-oriented dynamic properties (e.g., simplified state-space representations) and the closed-loop operational logic required by control algorithms (e.g., MPC) [22]. Consequently, the lack of an automated mechanism to translate static metadata directly into executable control strategies remains the primary bottleneck impeding the scalable deployment of advanced building automation.

2.3. Large language model for building energy applications

With the deeper research investigated in LLMs, substantial advancements have been made in the intelligent management of building environments [23], including knowledge management and retrieval [24], modeling and prediction [25], as well as fault detection and diagnostics [26]. Existing studies have revealed that LLMs possess the inherent versatility to execute complex technical workflows, extending their utility from text-based synthesis to domain-specific engineering challenges [27,28]. Regarding knowledge management, LLMs have been instrumental in automating semantic ontology alignment and facilitating natural language retrieval within Building Information Models (BIM), effectively bridging the gap between unstructured human queries and structured building data [29,30]. Li and Wang [31] proposed the BuildingGPT framework, which integrated a hybrid Vector-Graph Retrieval-Augmented Generation (VG-RAG) mechanism for building information retrieval. This approach utilized vector similarity search with knowledge graph traversal to accurately map natural language queries to formal SPARQL syntax, thereby enabling precise retrieval of semantic building data without requiring users to possess specialized query language expertise.

Advancing into predictive analytics, existing studies have leveraged the data mining and pattern recognition capabilities of LLMs to facilitate building load forecasting workflows. Zhang et al. [25] proposed a GPT-based framework that automates the entire data-driven modeling pipeline. Their approach incorporated an automated prompt generation mechanism and a code self-correction module, enabling the autonomous execution of feature engineering and model optimization. This innovation significantly lowers the technical barrier, allowing non-experts to achieve high-accuracy energy predictions without extensive manual coding. More recently, the focus has expanded to generative control and modeling, where LLMs are utilized to automatically generate executable code for Building Energy Models (BEM) and generate intelligent control logic for HVAC systems. Hu et al. [9] proposed AutoControl which developed an end-to-end automated workflow for designing PI controllers. By combining the semantic information extracted from Brick Models with the logic mapping and code generation capabilities of LLMs, AutoControl successfully translated high-level control objectives into executable control code without human intervention. However, this work is limited to generating model-free control algorithms (i.e., PI controllers), which fail to accommodate the multi-objective optimization and physical constraints inherent in complex building environments.

3. Methodology

3.1. Overview of the workflow

The overview of the proposed workflow AutoControl-MPC is illustrated in Fig. 1. The workflow is conducted through five LLM-based operations from semantic parsing of HVAC system to MPC generation. Firstly, the semantic information specifically the properties and relationships of control and measurement variables of the system is extracted and interpreted using the LLM-based agent. To ensure an end-to-end automated workflow, the Brick Model [18] is utilized as the source database storing the information of physical and logical assets in the HVAC system. This standardized data storage format allows AutoControl-MPC to be directly applied to the buildings that adopt such data format. Additionally, the control requirements of building users are also considered as the inputs to the LLM agent, where the requirements will further provide subjective suggestions for the LLM-based MPC generation in the sequential steps. After obtaining the variable information of the system, the heat pump and the thermal dynamics are automatically modeled using LLMs. These models are critical components within the MPC controller, therefore need to be fitted ahead. The resistance-capacitor (RC) model is utilized as the basic model for thermal dynamics modeling, and an auto-selection mechanism is developed to determine the optimal RC model for the MPC. The auto-fitted parameters of the heat pump and the thermal dynamics of building are embedded into the MPC formulation, and the MPC code is auto-generated by LLM-based agent. Then the control results are validated by a self-verification LLM agent which will conduct error detection and reasoning. If the control performance does not meet the user requirements, the iterative correction and refinement will be applied to model fitting and code generation.

3.2. Extraction and interpretation of semantic information

Acquiring accurate and complete information of actuators and sensors in the HVAC system is essential for designing the MPC controller. Traditionally, this process is accomplished by manually looking up the configuration handbooks of the systems, which is labor-intensive and time-demanding. To address this problem, the semantic web technology provides a potential solution for constructing a standardized data storage format that can describe the properties and relationships between physical and logic assets in the HVAC system. The Brick Model is one of the models that are built upon the semantic web technology. In this study, Brick Model is utilized as the data source for acquiring essential semantic information of actuators and sensors in the HVAC system.

The semantic parsing and interpreting is implemented based on semantic web technology and LLMs, which includes three steps [9], as shown in Fig. 2. In the first step, the variables in the Brick Model are identified and classified using SPARQL query based on their superclass types. The variable whose type is a subclass of `brick:Sensor` is classified as a measurement variable, while those falling in `brick:Setpoint` or `brick:Command` are categorized as control variables. Additionally,

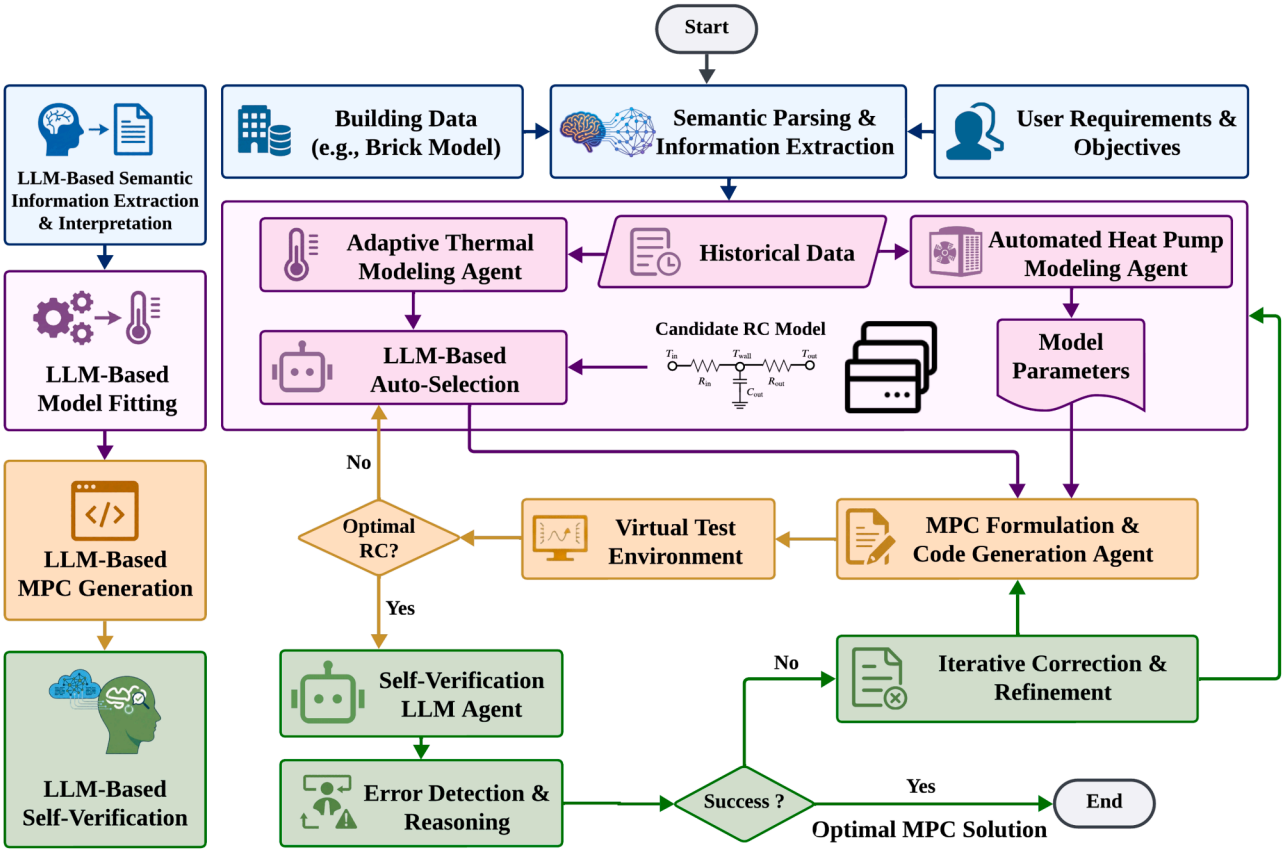


Fig. 1. The overview of the workflow of AutoControl-MPC.

the associated attributes of the entities are also extracted along with the classification. While the current implementation applies this extraction to single-zone operational environment, the SPARQL query logic is inherently scalable. By traversing topological predicates (e.g., `brick:hasPart` and `brick:isAdjacentTo`), the workflow establishes a foundation for identifying interconnected multi-zone variables in future complex system deployments.

In the second step, additional properties of variables are extracted, including maximum boundary, minimum boundary, measurement unit and activation condition. The querying is conducted based on the external reference linked via `ref:hasExternalReference`. The extracted properties provide rich semantic information for the reference of the subsequent LLM-based description generation. In the final step, the structured prompt is constructed using prompt engineering [9] to generate detailed descriptions for the control and measurement variables. The prompt is fed to LLMs, which then returns coherent and contextually precise variable descriptions explaining how the variables work in the HVAC system. The final output of the semantic parsing is generated in natural language, providing human-readable definitions for the variables and key information for the subsequent MPC generation.

3.3. Adaptive thermal modeling and auto-selection

The forecasting model is one of the critical components in the MPC providing future information of system states for control optimization. In HVAC system control, the thermal dynamics of the operational zone is modeled as the forecasting model. There are diverse approaches to model the thermal dynamics according to the existing studies ranging from physics-based methods to black-box methods [32,33]. To enhance the integration of LLMs within automated modeling workflows, we formulate the thermal dynamics based on the RC model. To clearly illustrate the physical foundation of these grey-box formulations without loss of generality, we take the widely-adopted 3R2C model [34–36] as an example, which follows the topology as shown in Fig. 3. The continuous-time heat balance equations for the 3R2C structure are governed by the following differential equations:

$$C_e \frac{dT_e}{dt} = \frac{T_a - T_e}{R_{ea}} + \frac{T_i - T_e}{R_{ie}} + A_e I_{sol}, \tag{1a}$$

$$C_i \frac{dT_i}{dt} = \frac{T_e - T_i}{R_{ie}} + \frac{T_a - T_i}{R_{win}} + A_i I_{sol} + \dot{Q}_h + a\dot{Q}_{int}, \tag{1b}$$

where T_i, T_e, and T_a denote the indoor air temperature, building envelope temperature, and outdoor air temperature, respectively. C_i and C_e represent the thermal capacitances of the indoor air node and the building envelope node. R_{ie}, R_{ea}, and R_{win} denote the thermal resistances between the indoor air and the envelope, the envelope and the outdoor environment, and the window thermal resistance, respectively. I_{sol} is the global horizontal irradiance, with A_i and A_e being the effective solar aperture areas. $\dot{Q}_h$ represents the thermal power provided by the HVAC system, and $a\dot{Q}_{int}$ accounts for the internal heat gains.

To seamlessly integrate these physical equations into the MPC framework, the continuous-time differential equations are discretized into a unified linear state-space representation using the zero-order hold (ZOH) method:

$$x_{k+1} = A_d x_k + B_d u_k + E_d d_k \tag{2}$$

where x_k represents the state vector (e.g., $x_k = [T_i, T_e]^{\mathrm{T}}$ for the 3R2C model), u_k is the control input, and d_k represents the disturbance vector. This discretized state-space representation directly serves as the thermal evolution model $f(\cdot)$ within the multi-objective MPC optimization problem formulated in Eq. (9c).

The prompt engineering and chain-of-thought technology is utilized to generate code for auto-modeling of thermal dynamics. The logical structure of the prompt is conceptually illustrated in Fig. 4, detailing the

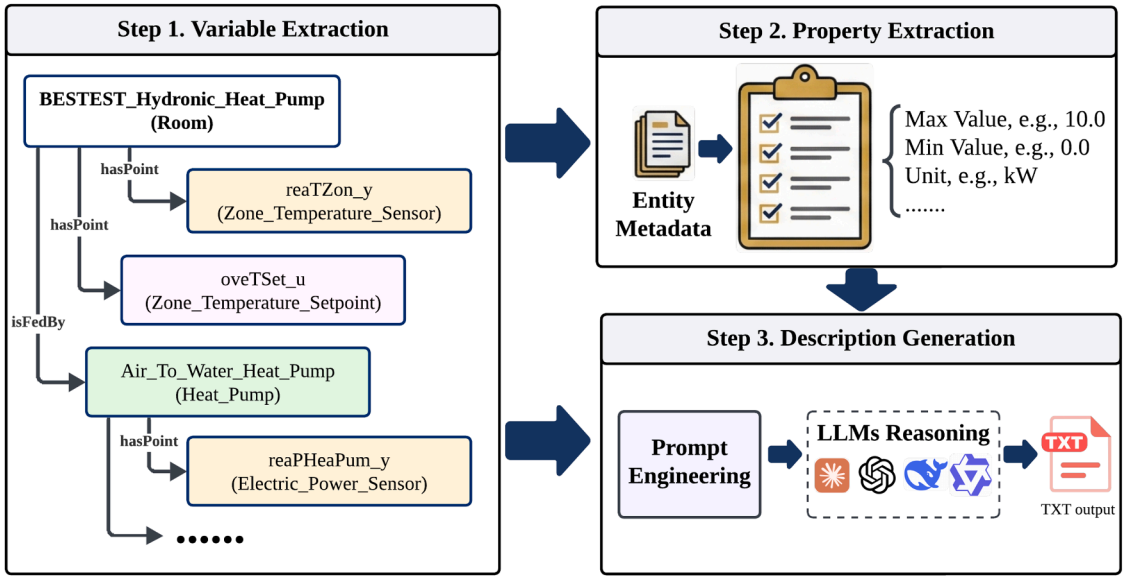


Fig. 2. The semantic information is extracted and translated into variable descriptions in natural language.

Table 1
The definitions of the inputs x_1 to x_6.

Input	Description
x_1	The detailed descriptions of system variables parsed from Brick Model
x_2	The notation of different RC models (e.g., 1R1C, 3R2C, etc.).
x_3	The variable mapping between physical concept and variable names from step 1.
x_4	The Model Interface Definition (MID) from step 2.
x_5	The code template for RC-model parameter fitting (i.e., using 1R1C as the reference).
x_6	The revision instruction from self-verification agent. Set to 'None' in the first trial.

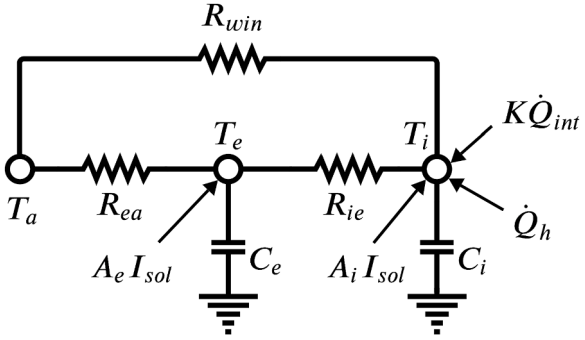


Fig. 3. The topology of the 3R2C thermal model.

data flow and core modules. The complete and detailed prompt text is provided in Appendix A.1. There are six slots ($[x_1]$ to $[x_6]$) for the inputs (x_1 to x_6). Table 1 shows the detailed information of these six inputs. The auto-modeling of thermal dynamics is divided into three steps. Firstly, to better understand the system variables and handle their relationships in the constructing RC model, the variable descriptions generated in Section 3.2 are fed to LLMs to parse unstructured metadata, mapping heterogeneous variable names (e.g., the heat pump compressor speed `oveHeaPumY_u`) to standard control theoretic notations based on their linguistic description and properties. This process will generate a JSON object mapping the physical concept to the raw variable name as shown in Fig. 5.

After obtaining the semantic mapping of variables, a physics-informed topological abstraction module is introduced to address the heterogeneity of building thermal dynamics (e.g., varying RC orders). The prompt is constructed based on Chain-of-Thought and fed to LLMs to generate MID. Unlike direct code generation, which often suffers from dimensional mismatch or hallucinations, this module facilitates the construction of RC model through three aspects: i) state vector definition: the module determines the requisite state nodes based on the specified RC model order (e.g., 3R2C) and maps the measured indoor temperature to the corresponding state index; ii) input vector configuration: the module identifies the exact order of physical inputs required to drive the differential equations of thermal dynamics. Additionally, it employs a dynamic mapping strategy to align available BMS tags with theoretical inputs, automatically flagging missing data for zero-padding; iii) sparsity pattern recognition: the topology of the state-space matrices are explicitly defined in the MID, enforcing zero-entries for physically non-interacting nodes. This approach decouples the physics derivation from software implementation, improving the generalization ability when applied to different RC models.

Following the abstraction phase, the parameter identification code is generated using a template-driven dynamic generation approach. The logical blueprint (MID) is translated into executable Python code. We employ a syntax-semantic decoupling strategy where a standardized 1R1C code template provides the syntactic scaffolding, while the LLMs instantiate the specific physics defined in the MID. To ensure the robustness of the generated script, several implementation guidelines are embedded in the prompt as shown in Appendix A.1. Firstly, since different RC models may have different dimensions of state x and input u, the LLM-agent is guided to dynamically constructs the data ingestion matrices based on the dimensions defined in the MID. This design avoids hallucinating fixed dimensions, which commonly occurs in LLMs. Secondly, in automated building modeling, the artificial truncation of parameter bounds often leads to suboptimal local minima for heavy-mass buildings. To address this problem, the prompt explicitly enforces a

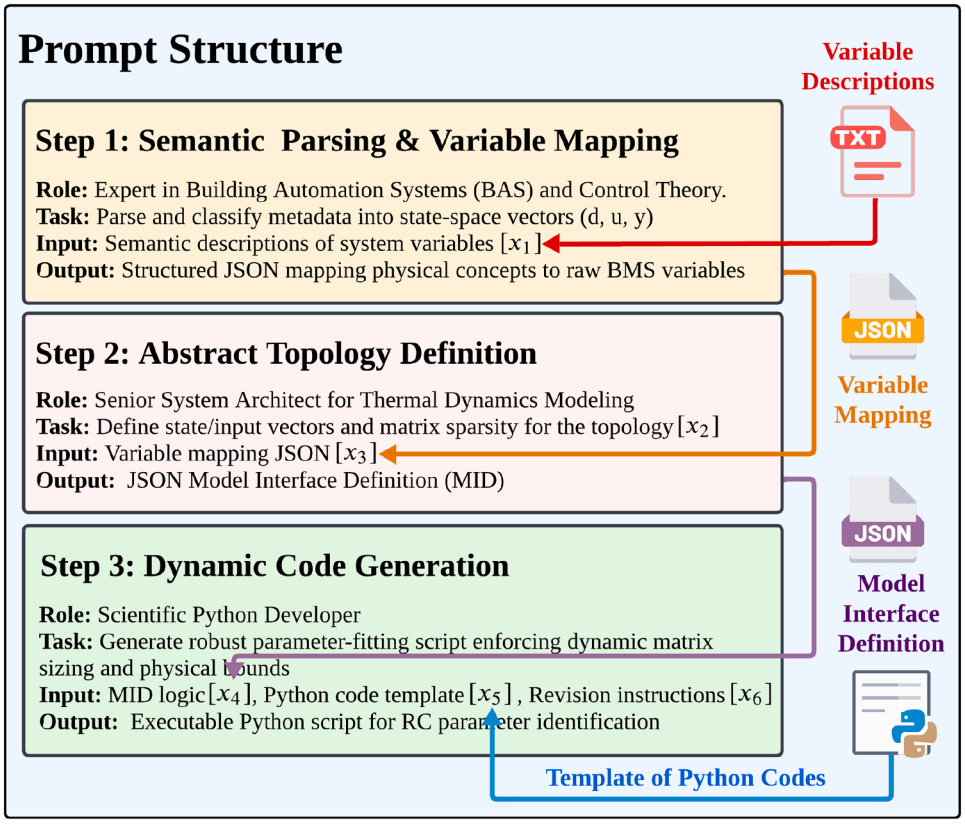


Fig. 4. The prompt structure for auto-modeling of thermal dynamics of the building.

```
{
  "Disturbance": {
    "outdoor_air_temperature": "weaSta_reaWeaTDryBul_y",
    "global_horizontal_irradiance": "weaSta_reaWeaHGloHor_y",
    "outdoor_relative_humidity": "weaSta_reaWeaRelHum_y"
  },
  "Control": {
    "temperature_setpoint": "oveTSet_u",
    "heat_pump_command": "oveHeaPumY_u",
    ...
  },
  "Observation": {
    "indoor_air_temperature": "reaTZon_y",
    "indoor_co2_concentration": "reaCO2RooAir_y",
    ...
  }
}
```

Fig. 5. The JSON object mapping between variable names and physical concepts.

physically expansive search space for thermal capacitance parameters. This ensures the optimization algorithm can accurately capture the massive thermal inertia characteristic of commercial concrete structures.

To formulate the parameter identification process, the objective function for the RC model is defined to minimize the Root Mean Square Error (RMSE) between the simulated indoor temperature $\hat{T}_i$ and the actual measured temperature T_i over a historical operational dataset of length N:

$$\min_{\theta} J_{RC} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\hat{T}_{i,k}(\theta) - T_{i,k}\right)^2} \tag{3}$$

where θ represents the vector of physical parameters to be identified (e.g., $\theta = [R_{ea}, R_{ie}, R_{win}, C_e, C_i]^T$ for the 3R2C model). The LLM does not perform numerical iterations directly. Instead, it acts as an intelligent constraint configurator and code generator. Based on the semantic abstraction (MID), the LLM dynamically generates the executable Python scripts based on the Nelder-Mead algorithm [37]. To ensure the physically feasible parameters and prevent the solver from converging to meaningless results (e.g., negative mass), the LLM strictly integrates domain-knowledge constraints into the script. For instance, the thermal resistance and capacitance are bounded strictly positive. Furthermore, as artificially narrow bounds often lead to suboptimal local minima for

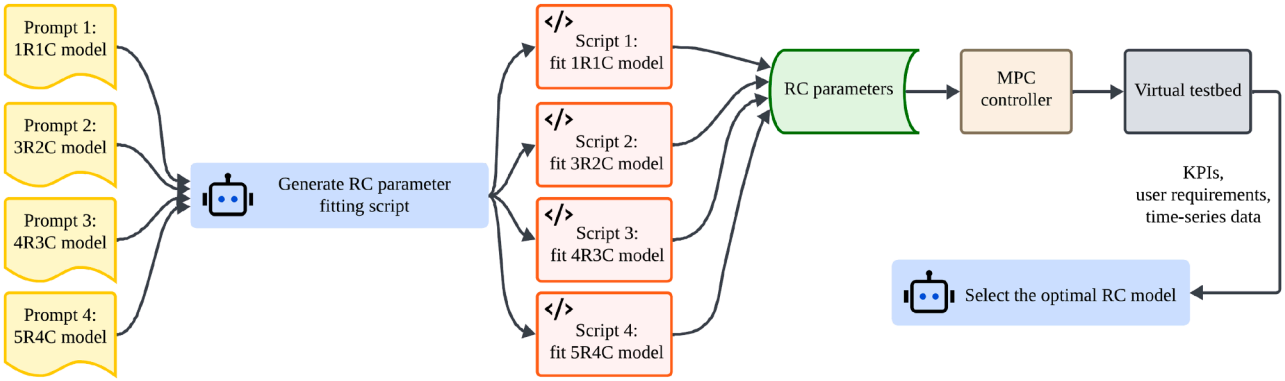


Fig. 6. The optimal RC model is auto-selected from four different RC models (i.e., 1R1C, 3R2C, 4R3C, and 5R4C).

heavy-mass buildings, the prompt explicitly guides the LLM to enforce a wider search space for thermal capacitance parameters.

Finally, an auto-selection mechanism is developed to determine the optimal RC model utilized in the MPC controller, as illustrated in Fig. 6. The workflow iteratively generates execution scripts to fit a candidate pool of RC models with varying structural complexities (e.g., 1R1C, 3R2C, 4R3C, and 5R4C). These models possess different state orders and capabilities for capturing the thermal dynamics of diverse building environments. To autonomously identify the most suitable topology, the LLM configures a quantitative multi-criteria protocol that explicitly balances prediction accuracy with structural complexity. First, the generated script calculates the RMSE for each candidate RC model on a validation dataset to evaluate the prediction accuracy. Subsequently, to prevent over-parameterization and minimize the computational overhead of the downstream MPC optimization, a marginal improvement threshold (set to 5% in this study) is enforced as a complexity penalty. The algorithm systematically evaluates the candidate models in increasing order of complexity. A higher-order model is only selected as the optimal structure if it reduces the validation RMSE by at least 5% compared to the best-performing lower-order model. Through this deterministic penalty mechanism, the AutoControl-MPC framework intelligently identifies and adopts the most accurate yet simple RC model for the specific building environment.

3.4. Automated physical modeling of the heat pump

Accurately modeling the behavior of terminal actuators and primary equipment (e.g., heat pumps, AHU cooling coils, or VAV boxes) is essential for MPC. This section introduces an automated workflow as shown in Fig. 7, generating the modeling scripts and fitting the physical parameters of the heat pump based on the LLMs. The complete prompt guiding this process is detailed in Appendix A.2. There are two critical steps to achieve this purpose. Firstly, the dominant control variable is auto-selected from the control variable list using lagged cross-correlation analysis. Secondly, a multi-hypothesis symbolic regression is developed to fit the physical model of the heat pump, especially the thermal power and electricity power in this study. The final outputs are the fitted parameters of the heat pump in the JSON format for the subsequent generation of the MPC controller. Here is the detailed explanation of two steps.

Step 1: Auto-selection of dominant control variable

Instead of relying on manual variable selection based on expert experience, a lagged cross-correlation analysis (LCCA) is employed in this workflow to robustly identify the dominant control from the control variable list. Despite the indoor temperature is usually influenced by extensive control variables (e.g., supply air temperature, supply air flow, etc.), we want to find the most relevant variable which can enforce a low-latency and high-excitation control over the indoor temperature.

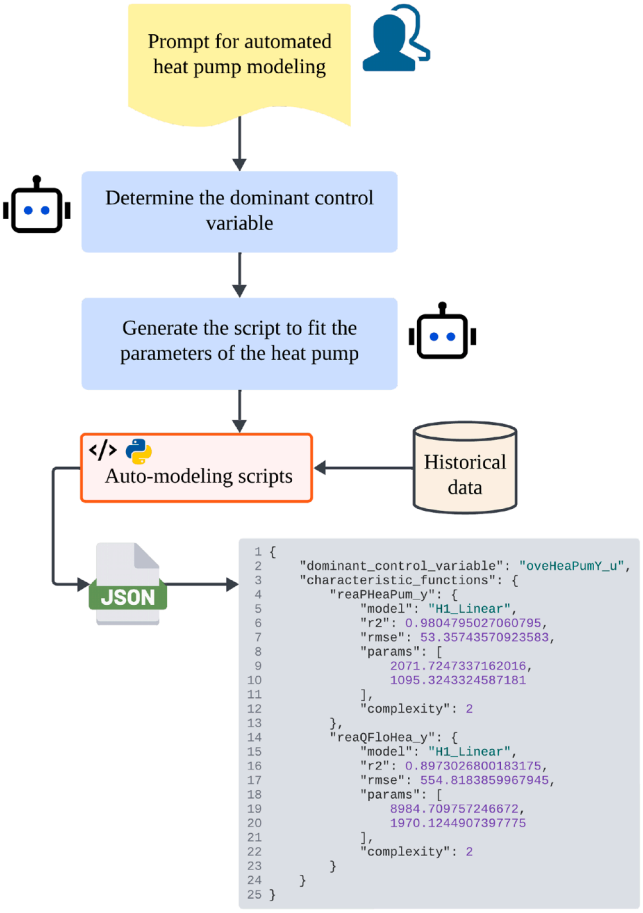


Fig. 7. The workflow of auto-modeling of the heat pump in the HVAC system. The output are the fitted parameters of the thermal power model and the electricity power model of the heat pump in JSON format.

In the first step, the LLM first acts as a semantic filter to identify potential control candidates (e.g., setpoints, valve positions, etc.). Then it generates a statistical verification script that sweeps through time lags (e.g., 0–60 minutes), identifying the control variable u that exhibits the highest peak correlation over the indoor temperature dynamics. Unlike standard correlation analysis, LCCA explicitly accounts for the thermal inertia and transport delays inherent in thermal dynamics processes. The algorithm scans a predefined time-lag window for each candidate control variable and solves a bi-level optimization problem given by

$$(u^*, \tau^*) = \underset{u^i \in \mathcal{U}, \tau \in \mathcal{T}}{\arg\max} |\rho_i(\tau)|, \tag{4}$$

$$\rho_i(\tau) = \frac{\sum_t \left(u^i_{t-\tau} - \bar{u}^i\right)\left(y_t - \bar{y}\right)}{\sqrt{\sum_t \left(u^i_{t-\tau} - \bar{u}^i\right)^2}\sqrt{\sum_t \left(y_t - \bar{y}\right)^2}}, \tag{5}$$

where, y_t is the indoor temperature at time step t, u is the control variable, τ is the time lag, $\mathcal{U}$ is the control variable candidate set, $\mathcal{T} = \{0, \Delta t, 2\Delta t, \ldots, \tau_{\max}\}$, $\bar{y}$ and $\bar{u}^i$ are the average values of the samples in the time window, respectively. The optimization process can be divided into two steps. Firstly, for each candidate control variable u^i, the local optimal time lag τ^* is solved by iterating over the discrete time lag set $\mathcal{T}$:

$$\tau_i^* = \arg\max_{\tau \in \mathcal{T}} |\rho_i(\tau)|, \tag{6}$$

Then, the variable exhibiting the highest absolute Pearson correlation coefficient (PCC) at a physically consistent time lag τ_i^* is identified as the dominant control input u^*:

$$u^* = \arg\max_{u^i \in \mathcal{U}} |\rho_i(\tau_i^*)|. \tag{7}$$

Note that the binary variables and the variables labeled as setpoint are autonomously filtered by the LLM before the correlation analysis, strictly following the heuristic filtering criteria defined in the prompt Appendix A.2. In the HVAC system, the performance curve of the heat pump often exhibits significant non-linear characteristic. The binary variable cannot provide sufficient amplitude excitation and granularity to capture the fine-grained partial-load behavior of the system by only offering input information at two extreme operating points (i.e., On/Off). Setpoint variables represent control intentions rather than physical energy injection. Using setpoints as model inputs creates a semantic gap, failing to capture the physical constitutive laws required for accurate thermal modeling. Furthermore, mapping direct actuator positions enables the MPC to learn true physical bounds and saturation characteristics, ensuring accurate control under capacity constraints.

Additionally, to prevent the selection of physically invalid control variables, the AutoControl-MPC framework embeds the lagged cross-correlation within a rigorous physical verification mechanism. First, semantic pruning is applied utilizing the building metadata (i.e., Brick Schema). The framework strictly limits the candidate pool to variables semantically defined as active actuators structurally supplying the target zone, thereby eliminating spurious correlations from non-causal sensors. Second, the use of a lagged metric inherently searches for temporal precedence, acting as an initial filter for causality. Finally, the physical validity of the selected control variable is explicitly verified during the multi-hypothesis symbolic regression and the final closed-loop MPC self-verification phase. If a selected variable lacks true thermodynamic causality, the generated physical equations will fail to achieve the required predictive fidelity, triggering the self-verification agent to reject the hypothesis and iterate.

Step 2: Physical parameter fitting of the actuator

To capture the potential non-linear characteristic of the heat pump performance, the workflow implements a multi-hypothesis testing strategy. It formulates a library of candidate functions from linear models to non-linear ones as shown in Table 2, which is also the inputs of slot [x_7] in the prompt. The agent fits these candidate functions to the heat pump data and autonomously selects the optimal one by balancing accuracy (R^2 score and RMSE) and complexity. Similar to the RC modeling process, the LLM integrates into this pipeline by generating the optimization scripts based on the Non-linear Least Squares method. The objective function is to minimize the sum of squared residuals between the actual power outputs and the hypothesized function predictions:

$$\min_{\beta} \mathcal{J}_{actuator} = \sum_{k=1}^{N} \left(y_k - f(u_k, \beta)\right)^2 \tag{8}$$

where y_k is the measured electricity or thermal power, u_k is the dominant control variable, $f(\cdot)$ is the specific hypothesis function from Table 2, and β represents the curve coefficients.

To ensure mathematical stability and physical feasibility, the LLM is guided to handle domain constraints strictly during code generation. For instance, when fitting the logarithmic function, the LLM automatically configures the parameter bounds to ensure the terms inside the functions are strictly non-negative (e.g., $x + c > 0$). The initial guesses β_0 are dynamically set to 1.0 by the LLM to ensure solver convergence, and default optimization tolerances are maintained. This structured generation ensures that the resulting model effectively captures physical behaviors without overfitting or crashing due to numerical singularities.

3.5. LLM-based automated MPC generation

Unlike the PI controller, MPC is an optimization framework that can adapt to multi-objective optimization problems. In this study, MPC controller code is auto-generated using LLMs, governing the indoor thermal comfort and energy consumption. At each forecasting horizon, the future indoor temperatures are predicted using the RC model (auto-fitted using the method in Section 3.3) and the physical model of the heat pump (auto-estimated using the method in Section 3.4), then the optimal control sequences are solved by minimizing the objective function given by:

$$\min_{u \in \mathbb{R}^m} \mathcal{J} = \sum_{k=n}^{n+m} \left(\pi^k \cdot p_h^k + \omega \cdot \delta^k\right) \tag{9a}$$

s.t.

$$\widetilde{T}_i^{n-1} = T_i^{n-1} \tag{9b}$$

$$\widetilde{T}_i^{k+1} = f\left(\widetilde{T}_i^k, T_a^k, \hat{Q}_h^k, \hat{\boldsymbol{\xi}}^k\right), \quad k = n-1, \ldots, n+m-1 \tag{9c}$$

$$P_h^k = f_e(u^k), \quad k = n, \ldots, n+m \tag{9d}$$

$$\hat{Q}_h^k = f_a(u^k), \quad k = n, \ldots, n+m \tag{9e}$$

$$(T_l^k + \epsilon) - \delta^k \le \widetilde{T}_i^k \le (T_u^k - \epsilon) + \delta^k, \quad k = n, \ldots, n+m \tag{9f}$$

$$\delta^k \ge 0, \quad k = n, \ldots, n+m \tag{9g}$$

$$0 \le u^k \le 1, \quad k = n, \ldots, n+m \tag{9h}$$

where, u is the control signal of the heat pump, m is the control horizon, π is the dynamic electricity price, p_h is the electricity power consumption of the HVAC system, ω is the penalty weight for comfort violation, δ is the temperature comfort deviation, $\widetilde{T}_i$ is the predicted indoor air temperature, T_i is the measured indoor air temperature, T_a is the outdoor air temperature, $\hat{Q}_h$ is the estimated thermal power output of the HVAC system, $\hat{\boldsymbol{\xi}}$ is the vector of other disturbances (e.g., solar radiation, internal heat gain, etc.), $f(\cdot)$ is the thermal dynamics model auto-fitted in Section 3.3, $f_e(\cdot)$ and $f_a(\cdot)$ are the physical models of electricity power and thermal power of the heat pump auto-modeled in Section 3.4, T_u and T_l are the upper and lower bounds of indoor air temperature set by building users, and ϵ is the relaxation coefficient of the indoor air temperature, which is set to 0.5 °C in this study.

Then, the MPC formulation is embedded into the template code, which serves as one of the key inputs for the prompt. The complete prompt is provided in Appendix A.3 for auto-generation of MPC controller code. The prompt takes four inputs from x_8 to x_{11}, which are defined in Table 3. Note that the prompt is not designed for a specific RC model, but can be adapted to different ones by integrating the auto-selected RC model structure and the fitted parameters from Section 3.3. Additionally, directly embedded non-convex functions of the actuator model into the MPC formulation may lead to a non-linear programming (NLP) problem, which is computationally expensive and prone to getting stuck in local optimum. To address this problem, the prompt explicitly guides the LLM-agent to approximate the non-convex functions into piece-wise linear (PWL) functions. This will transfer the original NLP problem into a mixed-integer linear programming (MILP) problem, which can significantly improve the computational efficiency using the branch-and-cut algorithms in widely-used optimization solvers (e.g., CPLEX [38], etc.).

Table 2
Functional hypotheses for system modeling.

Hypothesis	Mathematical Form	Description
Linear	$y = ax + b$	Baseline assuming constant proportional control response
Polynomial-2	$y = ax^2 + bx + c$	Quadratic approximation for variable efficiency curves
Polynomial-3	$y = ax^3 + bx^2 + cx + d$	Higher-order fit capturing complex curvature and inflection points
Saturation Exp	$y = a(1 - e^{-bx}) + c$	Models physical capacity limits and asymptotic saturation behavior
Logarithmic	$y = a \cdot \ln(x + c) + b$	Captures rapid initial response with diminishing marginal output
Power	$y = a \cdot x^n + b$	General power law suitable for fan and pump affinity physics
Square Root	$y = a\sqrt{x} + b$	Characteristic flow response of fluid through valves or dampers

Table 3
The definitions of the inputs x_8 to x_{11}.

Input	Description
x_8	The optimization solver, e.g., CPLEX [38], etc.
x_9	The parameters of the RC model.
x_{10}	The physical parameters of the actuator with linearity indication.
x_{11}	The code template with MPC formulation embedded inside.

3.6. Self-reflective verification and iteration

To enhance the reliability and robustness of the auto-generated MPC code without human intervention, a self-reflective verification mechanism is developed based on the LLMs as shown in Fig. 8. The complete prompt is provided in Appendix A.4, which takes input x_{12} from the system logs based on the MPC control performance in the virtual testbed. The self-verification agent evaluates the generated MPC against deterministic criteria by analyzing the control logs through a multi-tiered diagnostic protocol. Specifically, the agent conducts four critical checks: 1) verifying that the optimization solver status is strictly optimal to prevent the execution of infeasible constraints; 2) identifying whether thermal comfort bounds are violated while the actuator remains unsaturated. Such a condition indicates that the internal RC model is hallucinating and underestimating the thermal load, triggering a relaxation of thermal resistance bounds in the RC model fitter; 3) ensuring absolute thermal discomfort and energy costs fall within the user-specified absolute thresholds; and 4) confirming that actuator limits, anti-freezing thresholds, and equipment cycling rates strictly adhere to physical safety bounds. If all absolute criteria pass, the MPC code is verified and the workflow terminates successfully. Otherwise, the LLM-agent triggers the rule-based diagnostic routine to find the primary faulty module and generate targeted revision instructions. The iteration will be executed until the control performance meets the user requirements or reaches the maximum iteration number.

To mitigate the inherent probabilistic nature and potential hallucinations of LLMs, the auto-generated MPC code does not interface with the physical plant directly. While the MPC optimization problem inherently contains state and input constraints, these programmatic constraints are formulated and generated by the LLM, where generative errors or logical flaws may exist. Therefore, a Deterministic Safety Filter is implemented as an independent, outermost strict protection layer. This layer enforces hard hardware constraints (e.g., maximum compressor frequency limits, minimum chilled water temperature to prevent freezing, and maximum valve actuation rates) completely independent of the output of the LLMs.

4. Results

4.1. Setup and evaluation metrics

4.1.1. Virtual testbed

BOPTest was used as the virtual testbed for evaluating the control performance of the proposed approach in this study. BOPTest is an open-source platform [39] for benchmarking building energy optimization algorithms. Two different test cases were selected from BOPTest to evaluate the proposed approach under different building configurations. The building construction configurations of the two test cases are summarized in Table 4. The HVAC system structures of the two test cases are demonstrated in Fig. 9. The BESTEST_Hydronic_Heat_Pump is a small-scale residential building equipped with a hydronic air-to-water heat pump system, with a nominal thermal power of 15 kW for the heat pump, and a nominal mass flow rate of 0.5 kg/s for the floor heating system. In contrast, the Single_Zone_Commercial_Hydronic is a large-scale commercial building equipped with a more complex hydronic system with the nominal capacities of the district heating system, the Air Handling Unit (AHU) coil, and the radiator system being 500 kW, 250 kW and 255 kW, respectively. The weather data used in the simulations are from Brussels, Belgium and Copenhagen, Denmark for the BESTEST_Hydronic_Heat_Pump and Single_Zone_Commercial_Hydronic test cases, respectively.

4.1.2. Evaluation metrics and benchmark algorithms

To quantitatively evaluate the control performance of the proposed approach, thermal discomfort and energy consumption were used as the evaluation metrics in this study. The thermal discomfort was utilized to evaluate the indoor comfort, which is given by

$$\Delta T_d = \frac{1}{N} \sum_{z=1}^{N} \int_{t_s}^{t_e} \left(\max\left(T_i'(t) - T_l(t), 0\right) + \max\left(T_u(t) - T_i'(t), 0\right) \right) dt, \tag{10}$$

where, N is the number of zones in the building, t_s and t_e are the start and end time of the evaluation period, respectively, $T_i'(t)$ is the indoor temperature at time t, and $T_l(t)$ and $T_u(t)$ are the lower and upper bounds of the desired temperature range at time t, respectively. The energy consumption E_t [\$/m^2] is measured as the total cost of energy consumed by the HVAC system during the evaluation period.

Two benchmark algorithms were selected to establish a comprehensive comparative baseline. The default baseline controller embedded in BOPTest operates on standard reactive PI tracking. It serves as the lower-bound baseline representing legacy building operations. In contrast, the manually designed MPC, which was rigorously formulated and tuned by human experts utilizing perfect knowledge of the system parameters, serves as the upper-bound benchmark representing theoretical optimal performance.

Regarding the manually designed MPC controller, it was carefully designed to serve as a standard conventional baseline representing typical manual MPC implementation. A 3R2C model-based MPC was developed for the two test cases, since the 3R2C model is widely regarded as a practical trade-off between model accuracy and complexity [34–36]. The structure of the 3R2C model is illustrated in Fig. 3. To construct this manual baseline, the parameters of the 3R2C thermal network and the actuator models were identified using established grey-box system identification techniques and non-linear least squares regression based on a two-week historical operational dataset collected from the simulation test cases BESTEST_Hydronic_Heat_Pump and Single_Zone_Commercial_Hydronic in BOPTest. The MPC was formulated as a linear programming problem as described in Section 3.5. The prediction horizon and control horizon of the MPC controller were set to 12 hours and 15 minutes, respectively. The weighting factors in the objective function

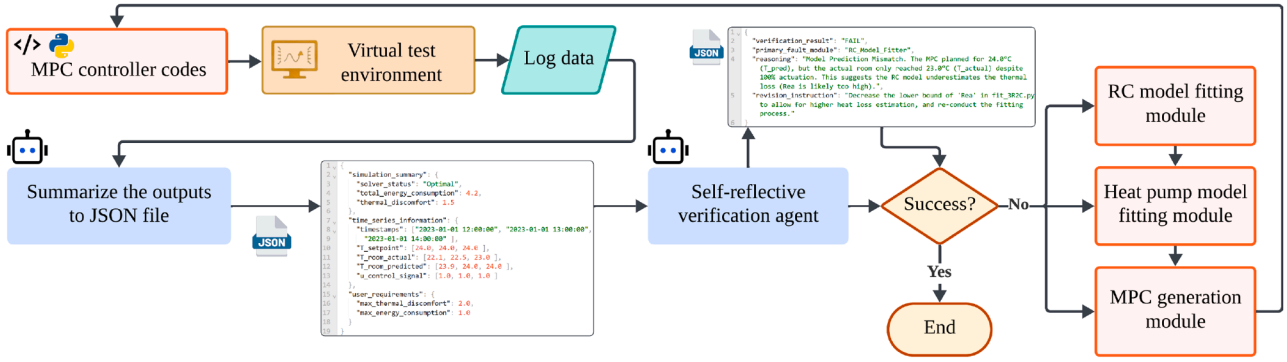


Fig. 8. The workflow of the self-reflective verification and iteration of the auto-generated MPC controller.

Table 4
Building construction configurations of the two test cases in BOPTest.

Component	BESTEST_Hydronic_Heat_Pump	Single_Zone_Commercial_Hydronic
Exterior Wall	0.90 cm wood siding + 6.15 cm insulation + 10 cm concrete block	20.00 cm concrete + 15.00 cm insulation
Roof	1.9 cm roof deck + 11.18 cm fiberglass + 1 cm plaster board	27.00 cm concrete + 52.00 cm insulation
Floor	15 cm concrete + 20 cm insulation + 5 cm screed + 1 cm tile	20.00 cm concrete + 15.00 cm insulation
Window	24 m^2	Triple-glass window model with 0.30 cm clear glass and 1.27 cm air
Area	12 x 16 m^2	8500 m^2

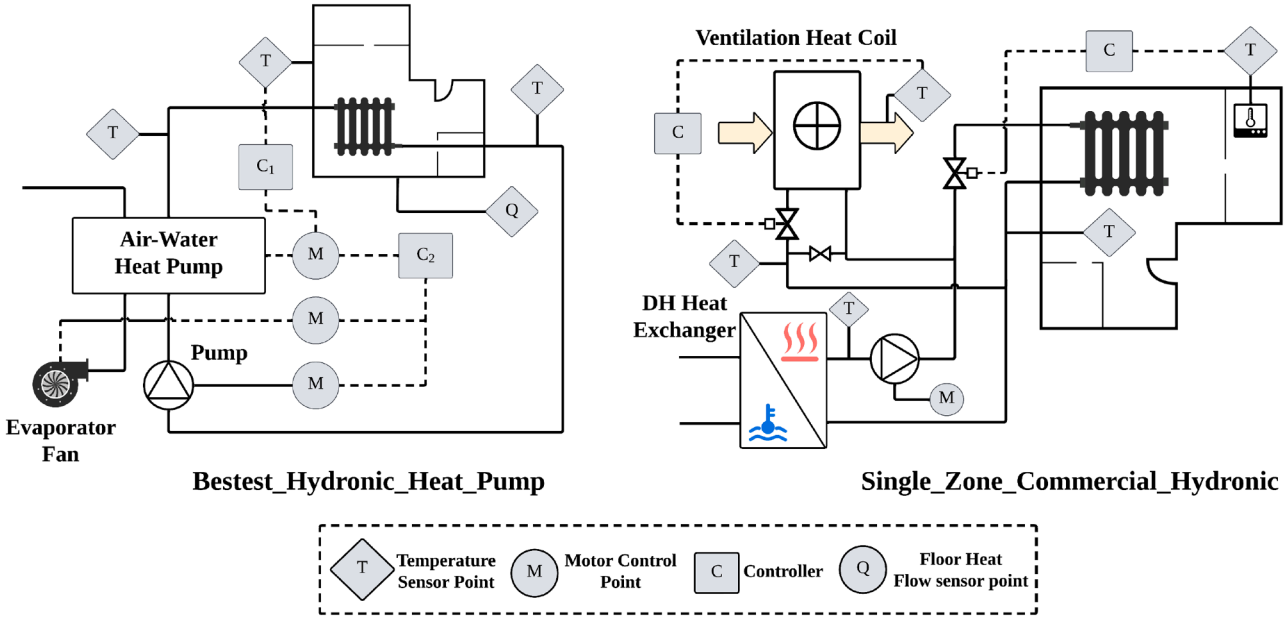


Fig. 9. The overview of the HVAC system structure of the two test cases in BOPTest.

are tuned to achieve a good balance between thermal comfort and energy consumption.

4.2. *BESTEST_hydronic_heat_pump test case*

We first evaluated the control performance of the proposed approach in the BESTEST_Hydronic_Heat_Pump test case. The control results of AutoControl-MPC compared with the baseline controller and manual-MPC are illustrated in Fig. 10. AutoControl-MPC shows superior control results maintaining the indoor temperature within the comfort band for the entire test period, while requiring a similar energy consumption compared to that of the baseline and manual-MPC controller. The quantitative evaluation results are summarized in Table 5. Compared to the baseline controller embedded in BOPTest—which relies on standard reactive Rule-Based logic and static schedules—AutoControl-MPC effectively eliminated thermal comfort violations (ΔT_d reduced from 2.96 to 0.00 Kh/zone). As expected, overcoming the limitations of passive RBC requires a slight increase in energy compensation (i.e., 7.3% cost increase). More importantly, when compared to the manual-designed MPC, AutoControl-MPC achieved a comparable control performance with respect to thermal comfort and energy cost. This control result was achieved by optimizing the operations of the heat pump based on the accurate forecasting of the building thermal dynamics, rather than simply following a predefined schedule as the baseline controller.

The optimal RC model auto-determined by AutoControl-MPC is the 3R2C model, which aligns with the structural choice of the manual-MPC controller. The prediction error of the auto-fitted RC model is shown in Fig. 11. To rigorously justify the optimal RC structure selected by the autonomous agent, the temperature prediction performance of all candidate RC topologies generated by AutoControl-MPC was evaluated

Table 5
The thermal discomfort, energy cost and average indoor temperature using the baseline controller, manual-MPC and AutoControl-MPC in the BESTEST_Hydronic_Heat_Pump test case.

	Baseline	Manual-MPC	AutoControl-MPC
Thermal discomfort [Kh/zone]	2.96	0.11	0.00
Energy cost [$/m^2]	0.41	0.31	0.38
Avg. temperature [°C]	21.04	21.12	21.75

Table 6
Temperature prediction performance of candidate RC models and the manually fitted model in the BESTEST_Hydronic_Heat_Pump test case.

Model Configuration	RMSE (°C)	MAE (°C)
AutoControl-MPC: 1R1C	0.99	0.91
AutoControl-MPC: 3R2C	**0.21**	**0.17**
AutoControl-MPC: 4R3C	0.31	0.26
AutoControl-MPC: 5R4C	1.08	0.90
Manual-MPC: 3R2C	0.36	0.30

and compared against the manually fitted model. As demonstrated in Table 6, the simple 1R1C model fails to adequately capture the complex thermal dynamics of the building envelope, resulting in significant prediction errors with RMSE up to 0.99 °C due to underfitting. The optimal prediction accuracy is achieved at the 3R2C configuration, where the RMSE drops substantially to 0.21 °C. Notably, as the model order further increases to 4R3C and 5R4C, the prediction performance does not improve but rather deteriorates significantly, with the 5R4C model exhibiting an RMSE of 1.08 °C. The over-parameterization inherent in higher-order models creates a highly non-convex optimization space, making the numerical solver highly susceptible to converging to sub-optimal local minima. Furthermore, the auto-fitted 3R2C model outperforms the manually fitted 3R2C baseline with RMSE of 0.36 °C. This superior accuracy indicates that the physical constraint boundaries and heuristic initializations dynamically configured by the LLM in the optimization scripts allow the solver to locate a better global optimum than traditional manual tuning based on human expertise.

The identified parameters from both AutoControl-MPC and the manual identification process are presented and compared in Table 7. The parameters identified by the auto-generated scripts are highly consistent with those obtained manually. The only notable discrepancy occurs in the envelope thermal capacitance C_e and envelope solar aperture A_e. Such divergences are mainly caused by the different initial searching points of the model parameters and the optimization boundaries. In the multi-order RC networks, different parameter combinations can yield highly similar dynamic responses due to numerical compensation. This comparison validates that the LLM-configured optimization scripts can autonomously extract physically meaningful parameters that perform reliably in the MPC framework. The estimated error of the auto-fitted physical model of the actuator is illustrated in Fig. 12, which showed that the physical model can accurately capture the dynamics of the heat pump with $R^2 = 0.96$ for electricity power model and $R^2 = 0.95$ for the thermal power model.

4.3. Single_zone_commercial_hydronic test case

In order to further validate the effectiveness and generalization ability of the proposed approach, we evaluated the control performance of AutoControl-MPC in another completely different test case—Single_Zone_Commercial_Hydronic. The control results are illustrated in Fig. 13. It can be observed that AutoControl-MPC successfully maintained the indoor temperature within the comfort band for most of the test period and achieved a high energy efficiency. The quantitative evaluation results are summarized in Table 8. AutoControl-MPC achieved the lowest thermal discomfort and the lowest energy cost among all the three controllers. It improved the thermal discomfort by 8.3% over the baseline controller and by 31.3% over the manual-MPC. On the other hand, while achieving a 7.1% reduction in energy cost compared to the baseline controller, AutoControl-MPC maintained the same energy cost as the manual-MPC.

Table 7
Comparison of identified 3R2C model parameters between manual-MPC and AutoControl-MPC in the BESTEST_hydronic_heat_pump test case.

Parameter	Unit	Manual-MPC	AutoControl-MPC
R_{ea}	K/W	3.40×10^{-3}	9.59×10^{-3}
R_{ie}	K/W	9.75×10^{-5}	1.52×10^{-4}
R_{win}	K/W	2.12×10^{-3}	1.80×10^{-3}
C_e	J/K	4.74×10^{11}	1.71×10^{8}
C_i	J/K	1.09×10^{8}	1.99×10^{7}
A_i	m^2	70.76	44.75
A_e	m^2	186.51	50.01
a	–	30.95	10.00

Table 8
The thermal discomfort, energy cost and average indoor temperature using the baseline controller, manual-MPC and AutoControl-MPC in the single_zone_commercial_hydronic test case.

	Baseline	Manual-MPC	AutoControl-MPC
Thermal discomfort [Kh/zone]	0.12	0.16	0.11
Energy cost [$/m^2]	0.14	0.13	0.13
Avg. temperature [°C]	20.42	20.16	20.18

Table 9
Temperature prediction performance of candidate RC models and the manually fitted model in the single_zone_commercial_hydronic test case.

Model Configuration	RMSE (°C)	MAE (°C)
AutoControl-MPC: 1R1C	0.39	0.31
AutoControl-MPC: 3R2C	0.67	0.54
AutoControl-MPC: 4R3C	0.46	0.36
AutoControl-MPC: 5R4C	**0.25**	**0.21**
Manual-MPC: 3R2C	0.35	0.28

The Single_Zone_Commercial_Hydronic test case represents a large-scale commercial building with highly complex thermal dynamics and massive thermal inertia. The temperature prediction performance of all candidate RC topologies generated by AutoControl-MPC was thoroughly evaluated, as presented in Table 9. Unlike the residential environment where the 3R2C model was optimal, lower-order models such as 1R1C (RMSE 0.39 °C) and 3R2C (RMSE 0.67 °C) failed to adequately capture the complex heat transfer behaviors in this test case. However, driven by the auto-selection mechanism, AutoControl-MPC intelligently scaled up the structural complexity and determined that the 5R4C model was the optimal topology, achieving the highest prediction accuracy with an RMSE of 0.25 °C and an MAE of 0.21 °C. Furthermore, by intelligently expanding the optimization boundaries of thermal capacities to accommodate the massive thermal inertia of the commercial building, the auto-fitted 5R4C model successfully outperformed the manually tuned 3R2C baseline (RMSE 0.35 °C). The prediction error trajectory of this optimal auto-fitted 5R4C model is illustrated in Fig. 14.

The actuator in this test case is the radiator system, where the physical model describes the relationship between the radiator valve position and the thermal and electricity power. Different from the heat pump in the previous test case, the actuator model here cannot be simply fitted by a linear regression model. Therefore, AutoControl-MPC tried several non-linear regression models and selected the cubic polynomial model as the optimal electricity power model with $R^2 = 0.99$, and selected the exponential saturation model as the optimal thermal power model with $R^2 = 0.77$. The estimated errors of all the auto-fitted physical models

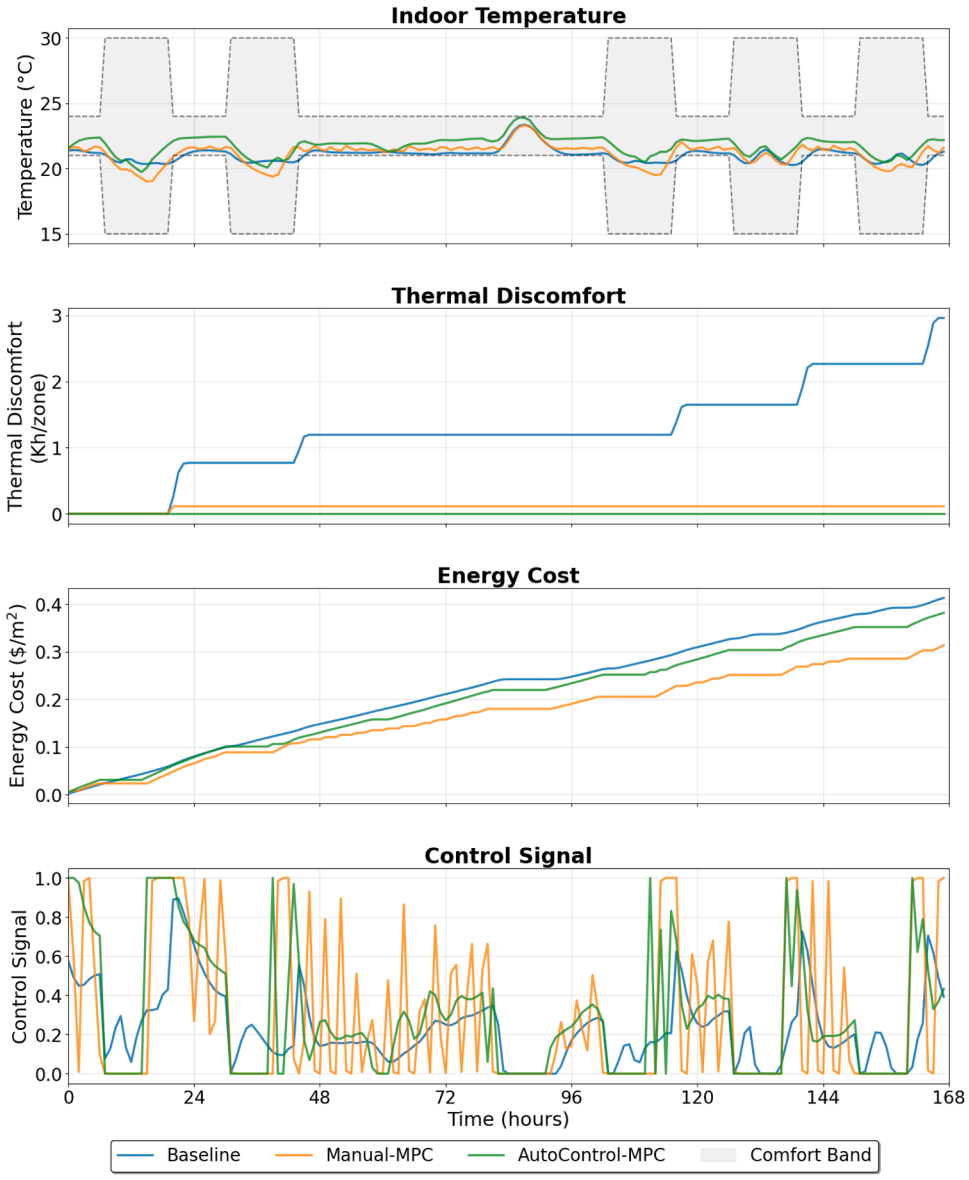


Fig. 10. The control results of AutoControl-MPC compared with the baseline controller and manual-MPC in BESTEST_hydronic_heat_pump test case.

of the heat pump are illustrated in Figs. 15 and 16. The control results demonstrated in the Single_Zone_Commercial_Hydronic test case indicate that AutoControl-MPC can handle different building types and HVAC system configurations based on the reasoning of LLMs to achieve a comparable and even better control performance than the manual-MPC controller.

4.4. Evaluation of success rate and cost

This section evaluates the success rate and cost of AutoControl-MPC under different configurations of the self-verification agent. To statistically evaluate the overall framework in this section, we define an externally successful trial as one where the final auto-generated MPC outperforms the baseline controller in both thermal discomfort and energy cost. It is important to note that this relative metric is strictly used for our external statistical evaluation. The internal AutoControl-MPC self-verification agent operates entirely independently using the absolute deterministic criteria defined in Section 3.6. A total of 50 independent trials per test case were conducted under identical initialization conditions. The success rates are illustrated in Fig. 17. It can be observed that by enabling the self-verification agent, AutoControl-MPC can achieve a higher success rate in both test cases, which reached an 82% and 66% success rate in the two test cases, respectively. In the self-verification process, the LLM agent helps to identify and correct potential errors in the auto-generated modules based on the log information. For example, the self-verification agent identified that the weighting factor in the objective function of the MPC was set too low which led to poor control performance, then suggested increasing the weighting factor to improve the penalty on thermal discomfort, as shown in Fig. 18.

To quantitatively evaluate the risk of LLM hallucinations, we analyzed the failed trials (18% in the residential case and 34% in the commercial case). In these instances, the LLM exhibited hallucinations such as defining physically impossible actuator boundaries or omitting critical equipment capacity constraints in the MILP formulation. However, rather than causing system failure, these erratic outputs were successfully caught by the syntax parsers and the pre-defined hard safety constraints within the self-verification agent (e.g., throwing a solver status: SAFETY_VIOLATION). This demonstrates that coupling LLM generation with deterministic safety bounds

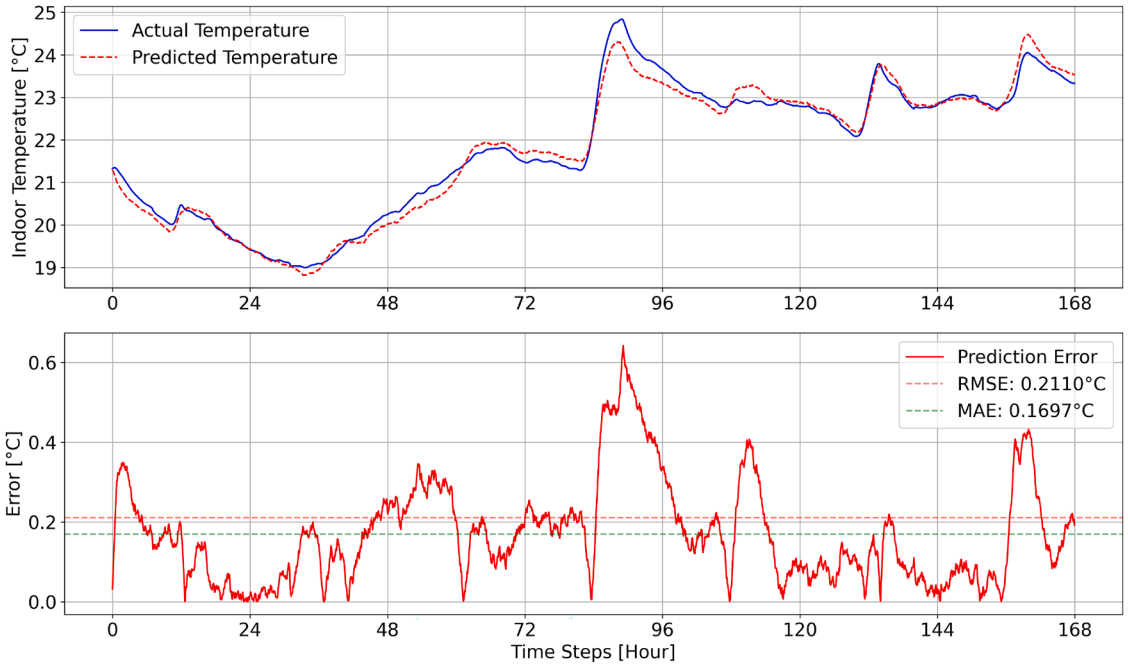


Fig. 11. The prediction error of auto-fitted RC model in BESTEST_hydronic_heat_pump test case.

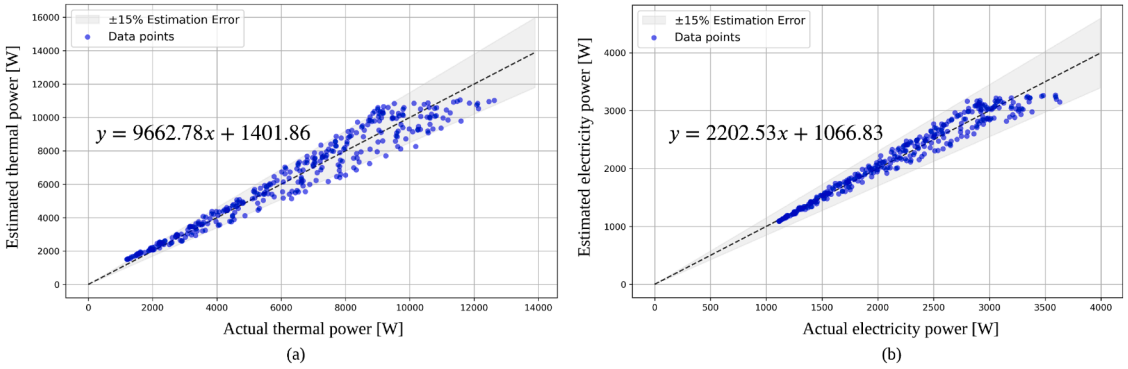


Fig. 12. The estimated error of auto-fitted physical model of the heat pump in BESTEST_hydronic_heat_pump test case: a) the thermal power model; b) the electricity power model.

effectively confines the damage of hallucinations to the software compilation phase, preventing them from propagating into physical hardware.

The average time and API cost of AutoControl-MPC were also evaluated and summarized in Table 10. It can be observed that the total time cost of AutoControl-MPC was around 24 minutes and 36 minutes for the two test cases, respectively. The major time cost came from the auto-fitting of the RC model and the heat pump model, which involves complex reasoning, such as the determination of the dominant control variable and the inferring of the optimal RC model structure. The total API cost of AutoControl-MPC was around $0.682 and $0.904 for the two test cases, respectively. In this study, the GPT-5 model is used as the LLM for code generation and reasoning, whose API cost is $1.25 and $10.0 per one million tokens for input and output, respectively. In practice, the time cost and API cost demonstrated in this study are acceptable, which indicates the feasibility of AutoControl-MPC for real-world applications.

5. Discussion

5.1. Contributions

This study proposed a novel approach AutoControl-MPC to address the challenges of automatically designing the MPC controller for HVAC systems using LLMs. Unlike traditional methods that rely heavily on expert knowledge and manual design, our approach achieves a fully automated workflow from extracting and interpreting semantic information to generating executable code of MPC controllers. The end-to-end AutoControl-MPC framework was evaluated on the standardized BOPTest HVAC control testbed. It is important to emphasize that the primary objective of AutoControl-MPC is not to significantly outperform state-of-the-art MPC control algorithms in terms of energy savings and maintaining thermal comfort, but rather to significantly reduce human effort by fully automating the labor-intensive controller development pipeline. The experimental results demonstrate that the auto-generated

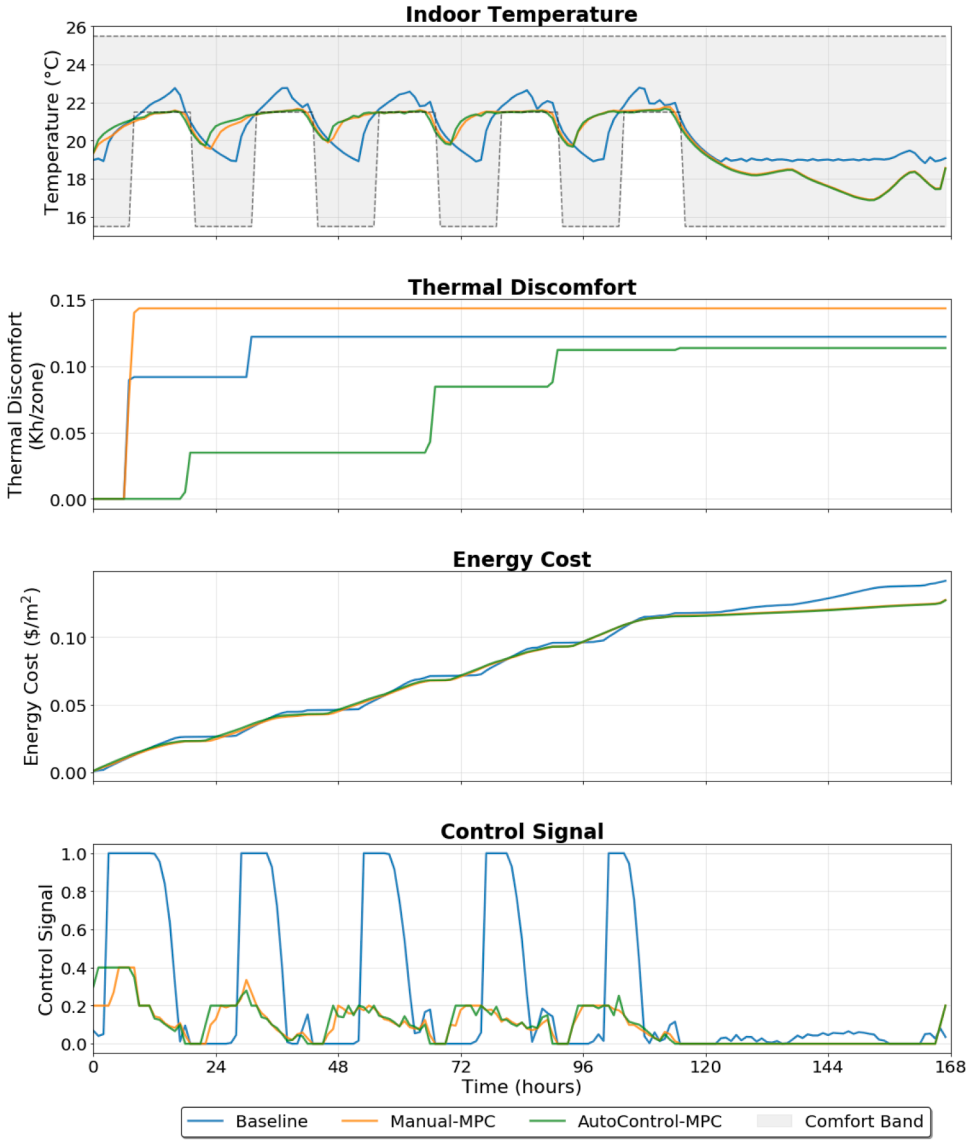


Fig. 13. The control results of AutoControl-MPC compared with the baseline controller and manual-MPC in single_zone_commercial_hydronic test case.

Table 10
The time cost and API cost of AutoControl-MPC for two test cases.

	BESTEST_Hydronic_Heat_Pump		Single_Zone_Commercial_Hydronic	
	Time cost [s]	API cost [$]	Time cost [s]	API cost [$]
Semantic parsing	16.60	0.019	123.53	0.034
Auto-fitting of the RC model	582.52	0.292	1223.76	0.396
Auto-modeling of the heat pump	411.10	0.096	370.84	0.105
Auto-generation of MPC code	99.12	0.050	132.13	0.065
Self-verification	302.04	0.225	337.24	0.304
Total	1411.38	0.682	2187.5	0.904

controller autonomously achieved a control performance comparable to, and even slightly better than, the conventional manually-tuned MPC controller. Specifically, AutoControl-MPC has several advantages over traditional methods. Firstly, the auto-selection of the optimal RC model provides significant flexibility and adaptability to different building configurations. In conventional practice, engineers often need to manually select and tune the RC model based on their expertise, which can be time-consuming and error-prone. Secondly, during the auto-fitting of the physical model of the actuator in the HVAC system, AutoControl-MPC achieved a high degree of accuracy in capturing the system dynamics, by first recognizing the dominant control variable and then fitting the optimal physical model to balance the complexity and accuracy. The accurate physical model of the heat pump is crucial for the MPC controller to make precise predictions and decisions.

Thirdly, the hallucination of LLMs is one of the major challenges in applying LLMs to safety-critical systems like HVAC control. AutoControl-MPC effectively addressed this issue by implementing a self-verification mechanism that cross-checks the generated modules based on the running log information. This mechanism significantly improves the reliability and robustness of the proposed framework by

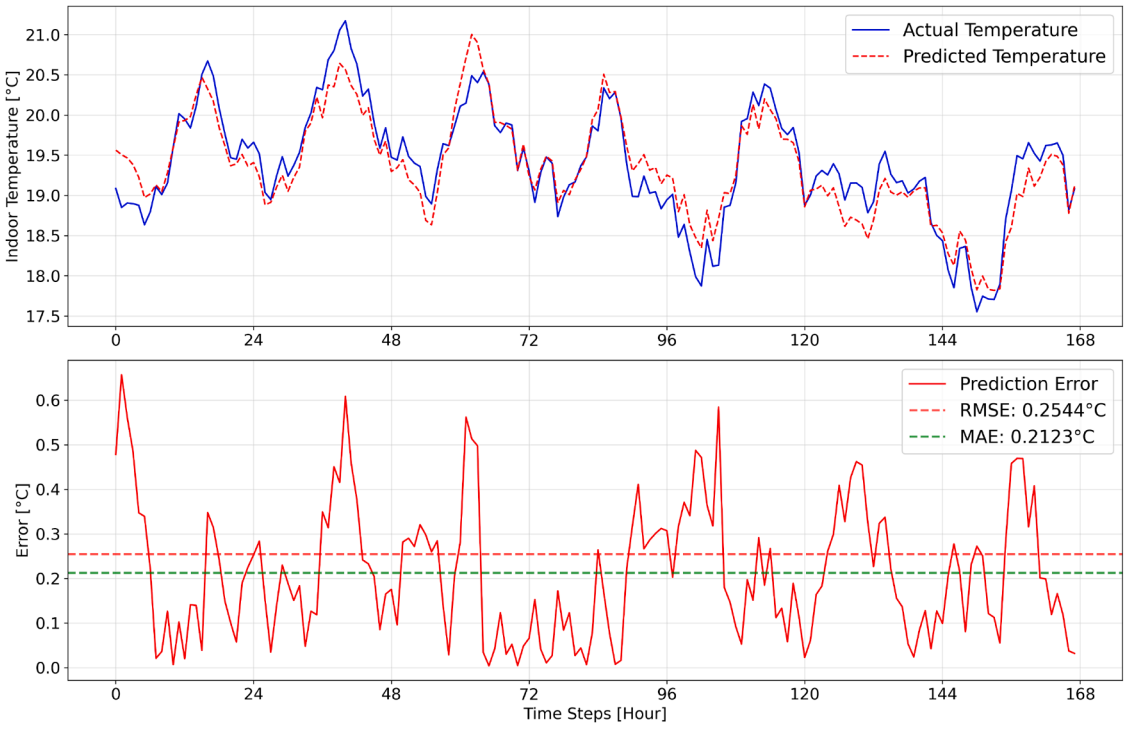


Fig. 14. The prediction error of auto-fitted RC model in single_zone_commercial_hydronic test case.

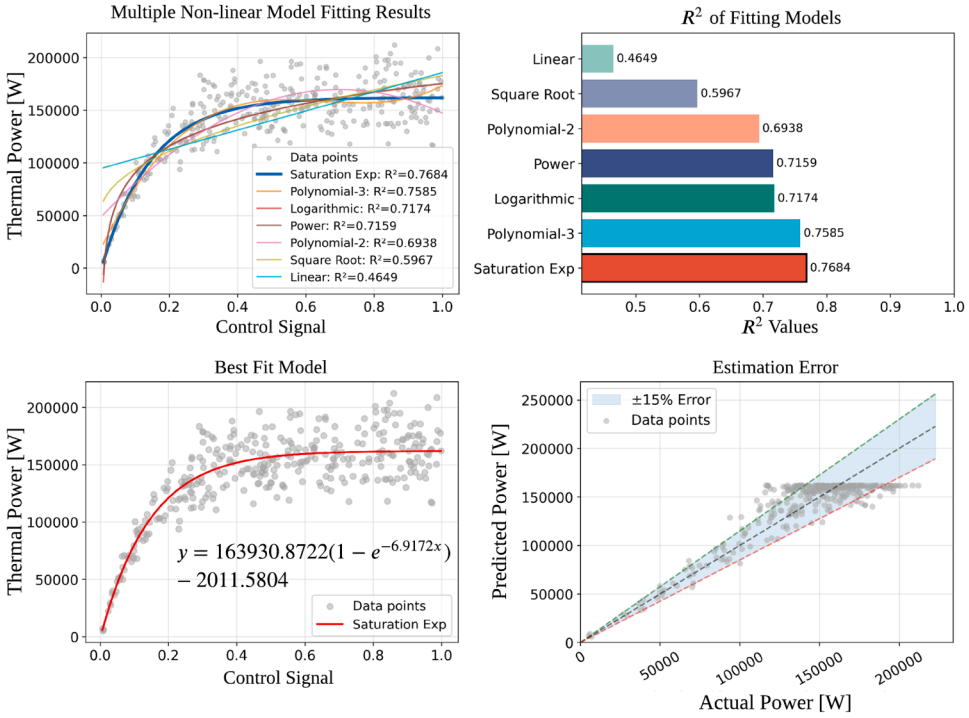


Fig. 15. The estimated error of auto-fitted thermal power model of the actuator in single_zone_commercial_hydronic test case.

recognizing and correcting potential bugs and improper parameter settings. Experimental results demonstrated that the self-verification mechanism successfully improved the success rate from 58% to 82% in the simple scenario and from 40% to 66% in the complex scenario. Lastly, from the perspective of economic cost and scalability, AutoControl-MPC significantly reduces the time and effort required to design and implement MPC controllers for HVAC systems. Generally, the traditional design of MPC controllers can take weeks of expert labor, while AutoControl-MPC can generate a functional MPC controller within hours with an API cost less than one dollar. This cost-effectiveness makes AutoControl-MPC a viable solution for large-scale deployment across multiple buildings and facilities.

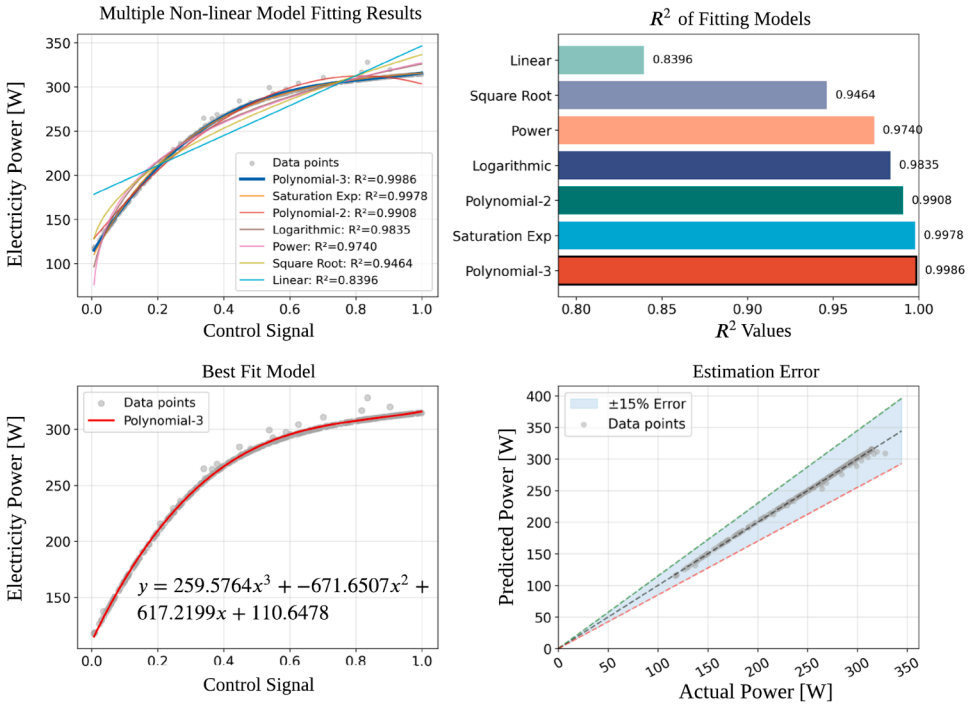


Fig. 16. The estimated error of auto-fitted electricity power model of the actuator in single_zone_commercial_hydronic test case.

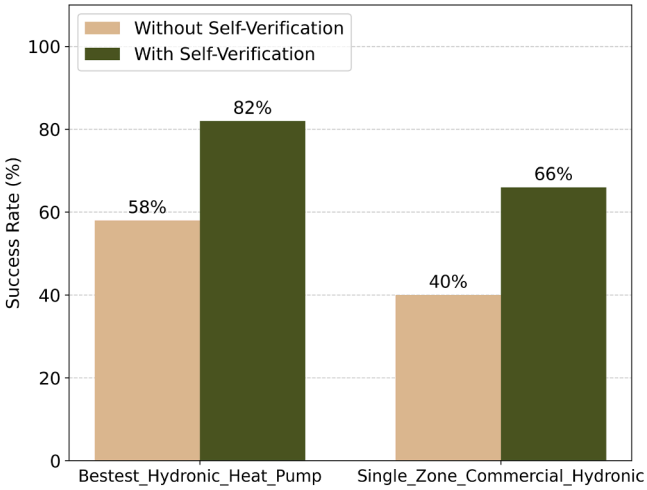


Fig. 17. The success rate of AutoControl-MPC in the two test cases under different configurations of the self-verification agent.

```
{
  "verification_result": "FAIL",
  "primary_fault_module": "General Tuning",
  "revision_instruction": "In MPC_Gen, increase the thermal comfort tracking
  weight in the MPC objective (the 'weight' parameter of MPC_Controller) by 2–3
  orders of magnitude relative to the energy penalty. Concretely: multiply the
  temperature tracking weight (e.g., W_temp_tracking) by 100x, re-run, and
  iteratively increase (up to ~300x) until thermal_discomfort <= 2.96 while
  keeping solver_status Optimal and constraints unchanged.",
  "diagnostic_summary": {
    "solver_health": {
      "status": "Optimal",
      "fault": false,
      "advice": null
    },
    ......
}
```

The revision instruction for MPC controller generation

Fig. 18. The output example of self-verification agent for code revision instruction.

5.2. Limitations

Despite the advantages of AutoControl-MPC, there are still several limitations that warrant further improvement. First, the source data of the workflow mainly relies on the metadata of the building. Although Brick Model provides standardized and structured representation of building information, the missing labels and incorrect points in the Brick Model may lead to the failure of the subsequent steps by misguiding the semantic parsing of LLMs. Secondly, the quality of the auto-fitted physical model of the actuator and RC model currently relies on the historical operational data of the HVAC system. In scenarios where the historical data is limited or contains significant noise, the fitting process may yield suboptimal models that cannot accurately capture the system dynamics. Thirdly, while the current rule-based self-verification mechanism and deterministic safety filters effectively mitigate operational risks, the generated MPC logic still lacks mathematical guarantees of safety. To deploy AutoControl-MPC in highly sensitive real-world HVAC systems, future work will integrate Formal Verification methods (such as reachable set computation or neural barrier certificates). This would provide the basis for a mathematical verification that, regardless of any bounded disturbances, the LLM-derived control policies never encroach upon critical safety zones. Fourth, the current validation scenarios primarily focus on single-zone environments. Real-world large-scale commercial buildings are inherently multi-zone, characterized by coupled Multi-Input Multi-Output (MIMO) dynamics and significant inter-zone heat transfer. Scaling AutoControl-MPC to such environments remains a challenge. To handle multi-zone configurations, the semantic parsing module will be modified to extract spatial adjacency relationships (e.g., `brick:isAdjacentTo`) and hierarchical equipment topologies from the Brick schema. Additionally, the automated thermal modeling agent will be upgraded to construct coupled RC network matrices rather than isolated models, capturing inter-zone thermal dynamics. Furthermore, to reduce the computational complexity in solving large-scale MIMO MPC formulations, the code generation prompt will be enhanced to implement decentralized or distributed MPC architectures, breaking down the global multi-objective optimization into zone-specific comfort subproblems.

6. Conclusion

MPC is considered as a powerful control strategy in HVAC systems, which offers significant advantages over traditional control methods such as PI control or RBC. By utilizing the forecasting of future system behaviors, MPC can optimize the control actions, achieving desired control objectives. However, manually designing and developing an MPC controller is a time-consuming and error-prone task that requires deep expertise in control theory, system modeling, and code implementation. To address the challenges, this paper has proposed an end-to-end fully automated workflow AutoControl-MPC for designing and implementing MPC controllers for HVAC systems using LLMs. Specifically, AutoControl-MPC first extracts the semantic information from the Brick Model and constructs a control-oriented state-space mapping of the HVAC system. Then, it automatically fits the thermal dynamics model and physical model of the heat pump based on the parsed semantic information. Next, the MPC is auto-formulated by integrating the fitted models and control objectives. The code for the MPC controller is then automatically generated and validated in the virtual testbed. Finally, a self-verification mechanism is developed to enhance the generated code and modules if the control results do not meet the control objectives. The experimental results on the BOPTest have demonstrated that AutoControl-MPC can effectively generate the MPC controller and achieve comparable and even better control performance than the expert-designed MPC controller. Specifically, the AutoControl-MPC has improved the thermal comfort by 100%, while increasing the energy cost by 18.4% compared to the manually designed MPC controller in the residential building test case. In the commercial building test case, AutoControl-MPC achieved a comparable energy cost level while reducing thermal discomfort by 31.3% compared to the manually designed MPC controller. Equipped with the self-verification mechanism, the success rate of AutoControl-MPC has reached 82% and 66% in the two test cases, respectively. The results have indicated the potential of leveraging LLMs to automate the design and implementation of advanced control strategies in HVAC systems.

CRediT authorship contribution statement

Ziqi Hu: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Mingchen Li:** Software, Investigation; **Xu Han:** Writing – review & editing, Investigation; **Lunlong Li:** Investigation, Data curation; **Zhe Wang:** Writing – review & editing, Supervision, Project administration, Investigation, Funding acquisition.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used DeepSeek Chat in order to check grammar and language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Declaration of competing interest

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

Acknowledgements

The authors acknowledge the support from the State Key Laboratory of Climate Resilience for Coastal Cities (ITC-SKLCRCC26EG01), National Science Fund for Excellent Young Scholars (Grant number 52322813), and the Shenzhen Science and Technology Innovation Committee (KCXST20221021111403009).

Data availability

No data was used for the research described in the article.

Appendix A. LLM prompt templates

This appendix provides the comprehensive, text-format prompt templates utilized by the AutoControl-MPC framework to autonomously generate the MPC controller. Specifically, the automated workflow is driven by four core LLM modules, which are used to construct the RC thermal models, actuator models, and generate the MPC controller script and the self-verification agent. The four prompts are detailed in the following subsections.

A.1. Prompt for RC thermal modeling

```
================================================================================
Step 1: Semantic Parsing & Variable Mapping
================================================================================
* Role
You are an expert in Building Automation Systems (BAS) and Control Theory.

* Task
Analyze the provided list of raw BMS points (metadata). Your goal is to semantically parse these
     variables and classify them into the standard State-Space formulation vectors:
1. Disturbances (d): Uncontrollable but measurable inputs (e.g., outdoor temp, solar radiation, internal
      loads).
2. Control Inputs (u): Manipulated variables (e.g., heating power, setPoints).
3. States/Outputs (y): Measured system feedback (e.g., indoor temperature).

* Input Data (Raw Metadata)
- [x_1]

* Output Format requirements
Return a structured JSON object mapping the physical concept to the raw variable name.
Example format:
{
  "y": {"indoor_temp": "raw_variable_name"},
  "u": {"heating_power": "raw_variable_name"},
  "d": {"outdoor_temp": "derived_or_assumed", "solar_radiation": "derived_or_assumed"}
}
* Note: If a specific disturbance (like Outdoor Temp) is missing from the list but is essential for
     thermal modeling, mark it as "REQUIRES_EXTERNAL_DATA".

================================================================================
Step 2: Abstract Topology Definition
================================================================================
* Role
You are a Senior System Architect for Thermal Dynamics Modeling.

* Task
Define the Topological Architecture for a [x_2] building thermal model.
You must bridge the gap between the available raw data and the theoretical physics requirements.

* Input Data (Raw Tags)
- [x_3]

* Architecture Definition Requirements
1. State Vector (x) Definition:
```

```
   - List all state nodes required by the physics of the requested model (e.g., Wall Outer, Wall Inner,
       Indoor Air, Internal Mass).
   - Define the Exact Order: x = [x_1, x_2, ..., x_n]^T.
   - Map the measured indoor temperature (`reaTRoo_y`) to the corresponding state index.
2. Input Vector (u) Definition:
   - List all physical inputs required to drive the differential equations.
   - Define the Exact Order: u = [u_1, u_2, ..., u_m]^T.
   - Data Mapping Strategy:
     - If a required physical input (e.g., "Fresh Air Load") exists in the raw tags, map it.
     - If it is MISSING in the data but REQUIRED by physics (e.g., "Solar on Mass"), define a fallback
         strategy (e.g., "Reuse global Solar Radiation" or "Set to Zero").
3. Matrix Sparsity Logic:
   - Describe the B-Matrix (Input-to-State) topology: Which input u_j affects which state x_i?
   - Example: "HVAC Heat (u_3) only affects Indoor Air Node (x_3)."

* Output Format (JSON Contract)
Produce a JSON "Model Interface Definition" (MID). This will be the blueprint for the coder.

================================================================================
Step 3: Dynamic Code Generation
================================================================================
* Role
You are a Scientific Python Developer.

* Task
Generate the `fit_[x_2].py` script based on the Model Interface Definition (MID) provided below.

* Inputs
1. Interface Definition (The Logic):
   - [x_4]
   - Crucial: You must construct the matrices (A, B) and Input Vector (u) strictly according to this
       definition. Do not default to the 1R1C structure.
2. Code Template (The Syntax): [x_5]
   - Instruction: Use this template ONLY for Python class structure (`__init__`, `fit`, `simulate`), `
       scipy.optimize` usage style. Discard the 1R1C physics and vectors.

* Implementation Guidelines
1. Dynamic Vector Construction:
   - Look at `input_u_order` in the MID (e.g., ['Ta', 'Isol', 'Qint', 'Qh', 'Qfr']).
   - In the `fit` method, construct the data matrix matching this EXACT order:
     ```python
     # Example dynamically generated based on MID
     u = data[['Ta', 'Isol', 'Qint', 'Qh', 'Qfr']].values
     ```
   - Self-Correction: If the MID says a column is missing in the CSV (e.g., 'Qfr'), generate code to
       create a zero-column first: `data['Qfr'] = 0.0`.
2. Matrix Dimensions:
   - Matrix A must be (N_x \times N_x) where N_x is `state_x.dim`.
```

```
   - Matrix B must be (N_x \times N_u) where N_u is `input_u.dim`.
   - Do not hardcode 2x2 or 2x4. Use the dimensions defined in the MID.
3. State Initialization (x_0):
   - The 1R1C template assumes a scalar or 2-vector.
   - You must generate an initialization strategy for a vector of size N_x.
   - Heuristic: Initialize T_{indoor} using the measured first point. Initialize other wall/mass
       temperatures to be equal to T_{indoor} or T_{ambient} (steady-state assumption).
4. Scientific Parameter Bounding (Crucial):
   - In the `fit` method, define the bounds for `scipy.optimize.minimize`.
   - Resistance (R): Use standard ranges (e.g., 1e-5 to 1e-1).
   - Capacitance (C): Apply a "Physically Expansive Search Space" strategy.
     - Reasoning: Commercial buildings often exhibit massive thermal inertia. Restrictive upper bounds
         can cause the optimizer to prematurely truncate the search.
     - Instruction: Set the upper bound for C_i and C_e to at least 1e12 (1 Terajoule/K). This ensures
         the optimizer is not artificially constrained for heavy-mass structures.
   - Solar/Internal Gains: Allow broad ranges (e.g., 1 to 500).
5. Optimization Algorithm:
   - Explicitly configure the optimizer to use the 'Nelder-Mead' method.
   - Syntax: `minimize(..., method='Nelder-Mead', ...)`

* Revision instruction: [x_6]

* Output
Generate the full, executable Python script. Only output the Python codes only, without any other text.
```

A.2. Prompt for actuator physical modeling

```
* Role
You are a Cyber-Physical Systems Researcher specializing in HVAC Data Analysis.

* Task
First, please identify the Dominant Control Variable (u) that regulates the Indoor Temperature.
Second, based on the identified control variable, Derive the mathematical characteristic function y = f(
    u) for the Actuator.
- Input (u): The control variable identified in Step 1.
- Outputs (y): Electric Power and Heating Power. Identify the notation of the output variables from the
    input data below.

* Input Data : [x_1]

================================================================================
Step 1. Reasoning the dominant control variable
================================================================================
1. Heuristic Filtering:
- Before calculating correlations, strictly filter the candidate list based on the following System
    Identification Principles:
- Rule A: Exclude 'Setpoints' (tags often containing 'Set', 'Sp', 'cmd').
```

```
  - Reasoning: We need to model the Open-Loop Plant (u -> y). Setpoints are reference signals, not the
      actual physical energy injection.
  - Important Exception: Do NOT exclude variables named 'cmd' or 'enable' purely by name, as they often
      represent continuous modulating signals (e.g., VFD speed 0-100%). Rely on Rule B to filter them if
       they are actually binary.
- Rule B: Exclude Binary/Discrete Variables (variables with only 2 unique values, e.g., 0/1 or On/Off).
  - Reasoning: We aim to fit Non-Linear Actuator Models (Polynomial/Saturation). Binary variables lack
      the amplitude granularity required to characterize partial-load curves. Only keep continuous/
      modulating variables (e.g., valve positions, inverter speeds).

2. Statistical Verification:
- For the remaining Continuous, Non-Setpoint variables, calculate the Lagged Cross-Correlation.
  - Hypothesis: In thermal systems, the effect of a control action is not instantaneous. It exhibits a
      Time Lag (dead time + time constant).
  - Action: Write a Python script to calculate the Lagged Cross-Correlation between each candidate u and
       the target y.
  - Criterion: The variable that shows the highest correlation magnitude at a physically reasonable lag
      (e.g., 10-60 mins) is the true Control Authority.

================================================================================
Step 2. Model the actuator: Multi-Hypothesis Symbolic Regression
================================================================================
Do not assume linearity. Actuators often exhibit saturation or polynomial characteristics.
You must generate a Python script to perform a Competitive Fit:

1. Define Model Library (The Hypotheses): [x_7]

2. Implementation Details:
- Use `scipy.optimize.curve_fit` for non-linear models.
- Handle domain constraints (e.g., x+c > 0 for Log, x >= 0 for Sqrt/Power).
- Set reasonable initial guesses (p0) to ensure convergence.

3. Engineering Best Practices:
- Time-Step Independence: The script must handle physical time units correctly. Note that the input time
     column (timestamp or time) is always in seconds. Always convert user-provided minute parameters to
    array steps based on the actual sampling interval derived from these seconds.
- Numerical Stability: Ensure the code is resilient to edge cases.
  - Handle NaNs/Infs and constant signals (zero std dev) without crashing.
  - Initialization Safety: Initialize correlation maximization variables (e.g., `best_corr`) to `0.0`,
      NEVER use `-inf` or `inf`. (Reason: `abs(corr) > abs(-inf)` evaluates to False, causing valid
      correlations to be ignored).
- Data Adaptability: Do not hardcode sampling rates (e.g., "5 minutes"). The script should strictly
    trust the `timestamp` column to determine the actual delta-T.

4. Evaluation Metric:
- Use R^2 Score and RMSE.
- Parsimony Principle: If R^2 differences between linear and non-linear models are negligible (<0.01),
    prefer the simpler Linear model.
```

```
5. Execution:
- Fit all models using ‘scipy.optimize‘.
- Automatically select the "Best Winner" based on the metrics and save the parameters of the Winner.

6. Strategy:
- If the optimal control variable’s best model has a very low R^2 fitting score (<0.01), then switch to
    the next candidate variable. Repeat this process until a good model is found or all variables are
    tested.

* Revision instruction: [x_6]

* Output
Generate the Python script that implements this logic and saves the best model parameters to a JSON file
    .
```

A.3. Prompt for MPC controller generation

```
* Role
You are an Optimization Algorithm Engineer specializing in Model Predictive Control (MPC) using the [x_8
    ] library.

* Task
Generate a complete, executable Python script ‘mpc_controller.py‘ defining the class ‘MPC_Controller‘.
Your goal is to replicate the structure of the provided Reference Code but adapt the internal logic
    based on the Actuator Model Profile provided in the Context.

* Context (Input Data)
1. Thermal Model Parameters (To be Hardcoded in ‘__init__‘): [x_9]
2. Actuator Model Profile (JSON): [x_10]
   (Note: If the JSON above indicates a Linear Model, you must use Case A below. If it indicates a Non-
       Linear Model, use Case B.)
3. Model Interface Definition (MID) of the RC model: [x_4]
4. Reference Code Template (with Non-Linear PWL Implementation): [x_11]

* Critical Implementation Instructions
1. Interface Compliance:
   - Strictly maintain the class structure: ‘__init__‘, ‘forecast‘, and (if needed) ‘
       _generate_pwl_breakpoints‘.
   - The ‘forecast‘ arguments and docstring must match the reference exactly.
2. Adaptive Actuator Logic (The Decision):
   - Case A: Linear Model (Detected if ‘model_type‘ is "Linear"):
     - Simplify: Do NOT generate ‘_generate_pwl_breakpoints‘. Do NOT use ‘m.piecewise‘.
     - Direct Substitution: Hardcode the linear parameters (‘slope‘, ‘bias‘) in ‘__init__‘.
     - Constraint: Inside the loop, directly use the linear equation:
       ‘Q_heat[i] == slope_q * u[i] + bias_q‘
       ‘P_elec[i] == slope_p * u[i] + bias_p‘
```

```
   - Case B: Non-Linear Model (Detected if `model_type` is non-linear e.g., Polynomial or Saturation):
     - Strict Replication: You MUST follow the Reference Code Template exactly.
     - Implement `_generate_pwl_breakpoints` in `__init__` using the specific non-linear formula found
         in `Actuator Model Profile`.
     - In `forecast`: Calculate `preslope`/`postslope` -> Define `m.piecewise` object -> Apply
         constraint `Q_heat[i] == func(u[i])`.
3. Thermal Dynamics Replacement:
   - Replace the specific 3R2C equations in the reference with the equations matching the provided
       Thermal Model Parameters (Rie, Rea, Rwin, Ci, Ce, etc.).
   - Ensure the control variable `u` drives the thermal dynamics via `Q_heat` (whether `Q_heat` is
       linear or PWL-derived).

* Revision instruction: [x_6]

* Output
Provide only the Python code. Ensure the logic correctly adapts to the provided JSON profile.
```

A.4. Prompt for the self-verification agent

```
* Role
You are a Senior Control Systems Auditor. Your goal is to verify the performance of an auto-generated
    MPC controller verified by a physics-based simulation testbed.

* Context
We have deployed the generated MPC controller.
- Input Data: Simulation logs containing KPIs and time-series trajectories.
- Modules under Review:
  1. `RC_Model_Fitter`: Responsible for thermal parameters (R, C).
  2. `Actuator_Fitter`: Responsible for heat pump power and electricity power curves.
  3. `MPC_Gen`: Responsible for the optimization code and constraints.

* Diagnostic Logic
Analyze the input JSON strictly following this logic:
1. Check Solver Health:
   - IF `solver_status` is NOT "Optimal":
     - -> FAULT: `MPC_Gen`
     - -> ADVICE: Check constraint feasibility, initial conditions, or variable bounds.
2. Check RC Model Accuracy (Via "Unsaturated Violation" Logic):
   - Condition: IF `T_room_actual` violates the setpoint bounds (e.g., < LowerSetp) AND the control
       signal `u_control_signal` is NOT saturated (e.g., average u < 0.95 during the violation period).
     - -> FAULT: `RC_Model_Fitter`
     - -> REASONING: Model Hallucination. The controller output partial power (e.g., 50%), believing it
         was sufficient to maintain comfort. Since reality violated the setpoint, the internal RC model
         significantly underestimates the thermal load (e.g., R_ea is too high, implying better
         insulation than reality).
     - -> ADVICE: The identified thermal resistance (R) is likely too high. Relax the upper bounds for R
          parameters in `fit_3R2C.py` to allow the fitter to find a "leakier" model.
```

```
3. Check Actuator Behavior:
   - IF control fails managing the indoor temperature within the comfort band for most period (showing
       large thermal discomfort):
     - -> FAULT: ‘Actuator_Fitter‘ OR ‘MPC_Gen‘
     - -> ADVICE: The actuator model might be ill-conditioned. Suggest change the fitted model or
         increasing the weight of the control penalty in the MPC objective function.
4. Check Overall Performance:
   - IF ‘comfort_violation‘ > ‘max_violation‘:
     - -> FAULT: General Tuning.
     - -> ADVICE: Increase the ‘weight‘ parameter in ‘MPC_Controller‘.
5. Hard Safety Constraints:
   - Actuator Limits: Control signal falls outside absolute hardware min/max bounds.
   - Thermal Safety: Room temperature crosses anti-freezing or extreme-heat thresholds.
   - Cycling & Slew Rate: Equipment cycles too rapidly (violating minimum on/off time) or signal changes
       too abruptly.
   - Power Limits: Electrical power draw exceeds the facility’s peak breaker capacity.
   - Fluid Protection: Supply water exceeds max safe temperature, or cooling coil drops below dew point.
     - -> FAULT: SAFETY_VIOLATION.
     - -> ADVICE: Strictly enforce these hardware limits within the optimizer’s bounds. Ensure an
         external deterministic Safety Filter clips unsafe signals before physical deployment.

* Input Data: [x_11]

* Task
Generate a Verification Report (JSON).
- If all metrics pass, set ‘verification_result‘: "PASS".
- If any check fails, set ‘verification_result‘: "FAIL", identify the ‘primary_fault_module‘, and write
    a specific revision_instruction for that module.
```

References

[1] Y. Su, Q. Jin, S. Zhang, S. He, A review on the energy in buildings: current research focus and future development direction, Heliyon 10 (12) (2024) e32869.

[2] N. Asim, M. Badiei, M. Mohammad, H. Razali, A. Rajabi, L. Chin Haw, M. Jameelah Ghazali, Sustainability of heating, ventilation and air-conditioning (HVAC) systems in buildings—An overview, Int. J. Environ. Res. Public Health 19 (2) (2022) 1016.

[3] R. Jin, P. Xu, J. Gu, T. Xiao, C. Li, H. Wang, Review of optimization control methods for HVAC systems in demand response (DR): transition from model-driven to model-free approaches and challenges, Build. Environ. 280 (2025) 113045.

[4] Y. Yang, J. Bjørnskov, M. Jradi, Optimizing HVAC systems with model predictive control: integrating ontology-based semantic models for energy efficiency and comfort, Front. Energy Res. 13 (2025) 1542107.

[5] R. Li, Z. Shari, M.Z.A. Ab Kadir, A review on multi-objective optimization of building performance-Insights from bibliometric analysis, Heliyon 11 (4) (2025) e42480.

[6] X. Sha, Z. Ma, S. Sethuvenkatraman, W. Li, A data-driven model predictive control strategy integrated rule mining to optimize HVAC operating under multiple occupants’ activities, Build. Environ. 287 (2025) 113822.

[7] V. Chinde, Y. Lin, M.J. Ellis, Data-enabled predictive control for building HVAC systems, J. Dyn. Syst. Meas. Control 144 (8) (2022) 081001.

[8] S. woo Ham, L. Paul, A. Casillas, A. Prakash, D. Kim, R. Brown, M. Pritoni, P. Grant, Practical challenges of model predictive control (MPC) for grid interactive small and medium commercial buildings, in: Proceedings of the 2024 Summer Study on Energy Efficiency in Buildings, ACEEE, 2024, pp. 1–15.

[9] Z. Hu, M. Li, H. Tang, Z. Wang, Autocontrol: an end-to-end fully automated workflow for control design of building energy systems, Energy 336 (2025) 138329.

[10] I. Ajifowowe, H. Chang, C.S. Lee, S. Chang, Prospects and challenges of reinforcement learning-based HVAC control, J. Build. Eng. 98 (2024) 111080.

[11] N. He, J. Guo, Y. Li, Y. Quan, R. Li, L. Yang, Stochastic model predictive control for the optimal operation of office buildings, Build. Environ. 267 (2025) 112248.

[12] W. Zheng, Z. Hu, D. Wang, Z. Wang, Optimizing building energy systems for grid-interactivity, comfort and resilience, Energy Convers. Manag. 340 (2025) 119927.

[13] Y. Yang, M. Jradi, Multi-objective optimization for balancing CO2 emissions and thermal comfort in a central heat pump system: a case study of a Danish office building, Energy Build. 342 (2025) 115870.

[14] D. Zhao, D. Watari, Y. Ozawa, I. Taniguchi, T. Suzuki, Y. Shimoda, T. Onoye, Data-driven online energy management framework for HVAC systems: an experimental study, Appl. Energy 352 (2023) 121921.

[15] M. Aniakor, V.V. Cogo, P.M. Ferreira, A survey on semantic modeling for building energy management, (2024) arXiv:2404.11716.

[16] F. Xue, L. Wu, W. Lu, Semantic enrichment of building and city information models: a ten-year review, Adv. Eng. Inform. 47 (2021) 101245.

[17] M. Baghalzadeh Shishehgarkhaneh, A. Keivani, R.C. Moehler, N. Jelodari, S. Roshdi Laleh, Internet of things (IoT), building information modeling (BIM), and digital twin (DT) in construction industry: a review, bibliometric, and network analysis, Buildings 12 (10) (2022) 1503.

[18] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, et al., Brick: towards a unified metadata schema for buildings, in: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, 2016, pp. 41–50.

[19] N. Luo, G. Fierro, Y. Liu, B. Dong, T. Hong, Extending the brick schema to represent metadata of occupants, Autom. Constr. 139 (2022) 104307.

[20] P. Delgoshaei, M. Heidarinejad, M.A. Austin, A semantic approach for building system operations: knowledge representation and reasoning, Sustainability 14 (10) (2022) 5810.

[21] Z. Chen, Z. Deng, A. Chong, Y. Chen, AutoBPS-BIM: a toolkit to transfer BIM to BEM for load calculation and chiller design optimization, in: Building Simulation, 16, Springer, 2023, pp. 1287–1298.

[22] L. Wan, F. Rossa, T. Welfonder, E. Petrova, P. Pauwels, Enabling scalable model predictive control design for building HVAC systems using semantic data modelling, Autom. Constr. 170 (2025) 105929.

[23] M. Liu, L. Zhang, J. Chen, W. Chen, Z. Yang, L.J. Lo, J. Wen, Z. O’Neill, Large language models for building energy applications: opportunities and challenges, in: Building Simulation, 18, Springer, 2025, pp. 225–234.

[24] M. Li, Z. Hu, P. Mohebi, S. Li, Z. Wang, Enhancing LLM-based building data query with chain-of-thought, retrieval-augmented generation, and fine-tuning, Autom. Constr. 182 (2026) 106738.

[25] C. Zhang, J. Zhang, Y. Zhao, J. Lu, Automated data-driven building energy load prediction method based on generative pre-trained transformers (GPT), Energy 318 (2025) 134824.

[26] Q. Zhang, C. Xu, J. Li, Y. Sun, J. Bao, D. Zhang, LLM-TSFD: An industrial time series human-in-the-loop fault diagnosis method based on a large language model, Expert Syst. Appl. 264 (2025) 125861.

[27] Y. Gu, H. You, J. Cao, M. Yu, H. Fan, S. Qian, Large language models for constructing and optimizing machine learning workflows: a survey, ACM Trans. Softw. Eng. Methodol. (2024). https://doi.org/10.1145/3773084

[28] L. Chen, A. Tophel, U. Hettiyadura, J. Kodikara, An investigation into the utility of large language models in geotechnical education and problem solving, Geotechnics 4 (2) (2024) 470–498.
[29] D. Kampelopoulos, A. Tsanousa, S. Vrochidis, I. Kompatsiaris, A review of LLMs and their applications in the architecture, engineering and construction industry, Artif. Intell. Rev. 58 (8) (2025) 250.
[30] O.B. Mulayim, L. Paul, M. Pritoni, A.K. Prakash, M. Sudarshan, G. Fierro, Large language models for the creation and use of semantic ontologies in buildings: requirements and challenges, in: Proceedings of the 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, 2024, pp. 312–317.
[31] M. Li, Z. Wang, BuildingGPT: query building semantic data using large language models and vector-graph retrieval-augmented generation, Build. Environ. 287 (2025) 113855.
[32] F. Jorissen, D. Picard, K. Six, L. Helsen, Detailed white-box non-linear model predictive control for scalable building HVAC control, in: Modelica Conferences, 2021, pp. 315–323.
[33] C.A. Thilker, H.G. Bergsteinsson, P. Bacher, H. Madsen, D. Calì, R.G. Junker, Nonlinear model predictive control for smart heating of buildings, in: E3S Web of Conferences, 246, EDP Sciences, 2021, p. 09005.
[34] X. Lu, A.M. Memari, Evaluation of selected dynamic models in comparison with hot box test results for measurement of building envelope thermal properties, J. Green Build. 15 (2) (2020) 29–43.
[35] A. Boodi, K. Beddiar, Y. Amirat, M. Benbouzid, Model predictive control-based thermal comfort and energy optimization, in: IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society, 1, IEEE, 2019, pp. 5801–5806.
[36] J. González, D. Mora, M.C. Austin, The thermal network approach to model Occupants' heat and CO2 generation interactions: a case study in an office building in panama, in: Journal of Physics: Conference Series, 2385, IOP Publishing, 2022, p. 012015.
[37] S. Singer, J. Nelder, Nelder-mead algorithm, Scholarpedia 4 (7) (2009) 2928.
[38] I.I. Cplex, et al., V12. 1: User's manual for CPLEX, Int. Bus. Mach. Corp. 46 (53) (2009) 157.
[39] D. Blum, J. Arroyo, S. Huang, J. Drgoňa, F. Jorissen, H.T. Walnum, Y. Chen, K. Benne, D. Vrabie, M. Wetter, et al., Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings, J. Build. Perform. Simul. 14 (5) (2021) 586–610.