SimBuild
2024

Eleventh National Conference of IBPSA-USA
Denver, Colorado
May 21 – 23, 2024

Simplifying Modeling for Building and District Energy Systems with Large Language Models

Saman Mostafavi, John T. Maxwell, Maksym Zhenirovskyy and Ion Matei
Palo Alto Research Center (PARC), part of SRI International
Palo Alto, CA 94304 USA

Abstract

We present a modeler-assistant tool designed to facilitate the creation and validation of building and district energy models. Our objective is not to develop a new modeling library but rather to enhance the usability of open-source modeling libraries and the efficiency of the model creation and simulation process. This methodology significantly reduces the manual effort required in model validation. It leverages Large Language Models (LLMs) and Python programming to transform textual description of requirements into Modelica models, complete with graphical annotations for thorough inspection. A key innovation is our API, which leverages syntactically constrained LLMs to streamline user interaction with the Python-based Modelica model generator, ensuring the consistent creation of valid models. Our approach has the potential to reduce manual effort for model validation from hours to minutes, significantly streamlining the process. We provide examples of model generation at various levels of input detail and showcase the integration of weather and operational data to calibrate the models, aligning it more accurately with real-world scenarios.

Introduction

Building Energy models (BEM) are crucial for designing energy-conserving structures in urban sectors, being key components of smart city digital twins. During the design phase, they guide energy-saving infrastructure planning, and during operation, they are integrated into AI-driven analytics for optimal energy management (Hong et al. 2020). As BEMs pave the way for innovative, energy-efficient urban designs, their integration into model-based methods emerges as a critical step for practical, scalable applications.

Model-based methods' capacity to quantify the likelihood and impact of potential risks allow for easier identification and mitigation of risks, thus improving project safety and efficiency. For instance, a designer can use BEM to simulate a design sketch, conducting parameter sweeps to assess various scenarios. This process allows for the examination of how changes in design elements impact energy efficiency. Additionally, BEMs facilitate streamlined refinements, resulting in time and cost savings. For example, if a new cooling/heating technology becomes available, BEMs can quickly incorporate the new technology via updated component models to assess its impact on energy efficiency without the need to reconstruct the physical model from scratch. Nevertheless, the most significant impact of these models is upon system-level optimization and planning. A comprehensive and dynamic system model facilitates the development of energy management and retrofit strategies on a district scale, enabling a synergistic effect that enhances overall system performance beyond the capabilities of individual components (Fuchs et al. 2016).

State of the Art. Modern tools like EnergyPlus and Modelica are established energy modeling benchmarks, known for their detailed simulation capabilities. However, applying them to real world examples can be labor-intensive, specifically when integrating numerous individual buildings and other energy component models into a cohesive simulation that accurately reflects the intricate dynamics of urban districts. Automation and adaptability are two approaches for reducing the usage of EnergyPlus and Modelica. Tools such as the Modelica Buildings Library and OpenIDEAS have been developed to enable the integration of various data streams into complex models (Wetter et al. 2014; Remmen et al. 2018; Baetens et al. 2015). These tools are pivotal in automating workchains, crucial ingenerating complex energy models. More recently, BESMod (Wüllhorst et al. 2022) distinguishes itself by addressing two key challenges: it provides a standardized approach for integrating diverse domains within a Building Energy System (BES), and streamlines system-level analysis of complex models, reducing time and error-prone processes signifi-

cantly.

Furthermore, the integration of Large Language Models (LLMs) into design and control domains marks a significant trend. For instance, in high-level synthesis, tools like VeriGen (Thakur et al. 2023a) utilize LLMs to translate natural language into Verilog, thereby enhancing Verilog code generation tasks. AutoChip (Thakur et al. 2023b) merges the interactive capabilities of LLMs with Verilog simulation outputs to create modules. In robotics, LLMs generate low-level control commands, acting as efficient feedback controllers in complex systems (Brohan et al. 2022). These developments in LLMs are not only pivotal in simplifying the complexities of simulation setup and process control but also hold potential for innovative applications in energy modeling, bridging gaps between complex modeling tasks and user-friendly interfaces.

Motivation and Design Considerations. Unleashing the full potential of BEMs for urban areas hinges on ensuring user-friendliness, reusability, and precision. To illustrate, a recent case study (Hinkelman et al. 2022) underscores the benefits of using a model-based approach, particularly in evaluating energy and financial impacts of actions like replacing chillers or adding thermal storage, and in conducting system flexibility analysis, which leverages the thermal inertia of distribution piping. However, despite these advantages, it is evident that the process still involves numerous manual stages. These stages, which are crucial in developing and validating models based on field data, can be labor-intensive and error-prone. Our work, therefore, focuses on developing a modeler-assistant tool that aims to minimize these manual efforts. In designing such a tool, we consider the following features:

- Users interact with the tool by providing text inputs, the specificity of which can vary. In cases where the descriptions are less detailed, the tool employs inference mechanisms to deduce and supplement the missing information. Additionally, it utilizes side-channel information, where applicable, to further enrich and complement the user-provided data.

- User inputs are converted into an intermediate data structure to maintain consistency across the model generation process.

- We employ the Modelica language for creating models, adopting a reduced-order framework that builds upon the FastBuildings library, a method elaborated in (Remmen et al. 2018). This approach facilitates efficient and streamlined model development.

Detailed explanations of these mechanisms and their implementation will be further elaborated in the Methodology section.

Contributions. We leverage advancements in LLMs to develop an intermediary API between the user and a Python program synthesizing the model to reduce manual labor in energy modeling. Our tool processes model descriptions expressed in natural language, converting them into structured formats. We ensure the accuracy of these data structures using syntactically constrained LLMs, and their generative capabilities fill gaps in the textual inputs. These LLMs are specifically designed to adhere to syntactic rules. In our application, they generate outputs that conform to the syntactic structures of the Modelica modeling language, thereby ensuring that the models produced are not only semantically accurate but also syntactically correct. Additionally, we employ a Python program to take this structured representation and instantiate models. This program not only generates models but also annotates them for graphical representation, enhancing their clarity and usability. By integrating weather and operational data, we refine and calibrate the models, ensuring their relevance and accuracy with respect to real world examples. This approach simplifies the modeling process, making it more accessible to a broader range of practitioners, and bridges the gap between complex energy modeling concepts and their practical application.

Methodology

Our proposed methodology, as depicted in Figure 1, begins with an 'unstructured' text description of a building or district energy system. This description can vary in detail, ranging from a basic outline, such as a 'model of a five-zone building with central HVAC and individual VAV reheat boxes,' to more intricate specifics like building geometry, windows, and doors. An LLM processes this initial input, and creates a structured JSON file that outlines the components and their interactions within the model. This file is then parsed by a Python module to generate initial Modelica *.mo* files, representing the energy system's model. The raw model undergoes further refinement and optimization, incorporating additional user-provided data to mirror the real-world behavior of the energy system accurately. This step-by-step process results in a comprehensive model that aligns precisely with user specifications and real-world data, which we will discuss in detail in the following sections.

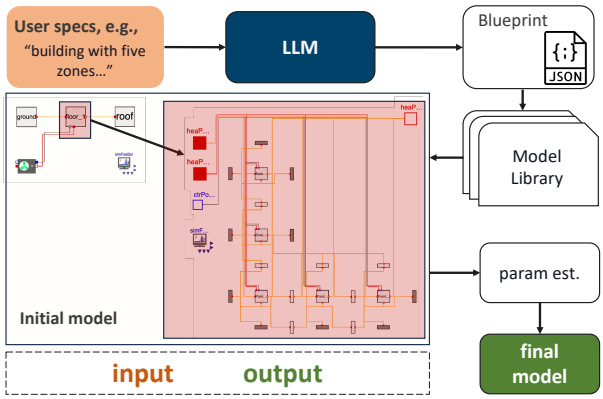


Figure 1: The overall approach: the figure illustrates an automatically generated example model diagram for a five-zone building.

LLM as a Modeling User Interface

We employ LLMs to transform textual descriptions into executable Modelica models, capitalizing on their proficiency in processing natural language inputs. However, LLMs, trained on extensive internet data, can sometimes 'hallucinate' or generate misleading information. This, coupled with their broad capabilities and limited controllability, poses challenges in creating complex tasks (Wu, Terry, and Cai 2022) like consistent Modelica models. To counter these issues, we adopt a structured form-filling approach, where the LLM fills out a predefined template. This strategy not only mitigates the risk of inaccuracies but also improves the conformity of the output with the Modelica model structure.

Rather than developing a new model from scratch, we guide the generative process of pre-trained models. This approach is advantageous as it circumvents the need for tailoring a specific dataset, which is often challenging to procure. We utilize pre-trained models from the Hugging Face Transformers text generation library (Wolf et al. 2019), leveraging their extensive training on diverse datasets. To further enhance the efficiency and reliability of our system, we have revised our methodology to act as a wrapper around these open-source LLM models. This wrapper is specifically designed for structured data environments, where many tokens are fixed and predictable. During the generation process, our system automatically fills in these fixed tokens and focuses the LLM's generative power on the content tokens that require dynamic generation. This streamlined approach ensures more controlled and precise generation, tailoring the LLM's capabilities to the specific requirements of building Modelica models. The use of these pre-trained models ensures that our methodology remains adaptable for future developments in the field.

Constrained Token Generation

LLMs work by breaking down text into smaller pieces, known as 'tokens', similar to puzzle pieces of a language. Each token represents a word or a part of a word, helping the model to grasp and construct language piece by piece. The model then predicts what comes next in the text. Formally, consider an unstructured textual input. This text is tokenized into a sequence of tokens T via a tokenization function τ:

$$\begin{aligned} T &= \tau_{encoder}(\text{Text}), \\ \alpha &= \mathcal{L}(T_{<i}, \theta), \\ T_i &\sim \text{Categorical}(\alpha), \\ \text{Text}' &= \tau_{decoder}(T), \end{aligned} \tag{1}$$

where $T_{<i}$ denotes the sequence of tokens up to but not including position i, and T_i is the token at position i. The function $\mathcal{L}$ represents the LLM which, given the token sequence and the model parameters θ, computes a probability distribution α over all the possible tokens. The next token T_i is sampled from this distribution. Finally, the sequence of tokens T is converted back into a text string Text$'$ using the tokenization decoder $\tau_{decoder}$. This process iteratively continues, enabling the LLM to generate new tokens at each step.

The common and efficient method for generating compliant sequences involves iteratively evaluating the entire vocabulary to filter out tokens that meet the desired constraints. This is achieved by assigning a probability of zero to invalid tokens, thereby excluding them from the pool of potential tokens for the next stage of generation. One such way to do this is a boolean mask function $\mathbf{m}(T_{<i})$ that imposes specified constraints on the generation process by element-wise multiplication with α, resulting in a modified distribution α'.

$$\begin{aligned} \alpha' &= \mathbf{m}(T_{<i}) \odot \alpha, \\ T_i &\sim \text{Categorical}(\alpha'), \\ \text{Text}' &= \tau_{decoder}(T). \end{aligned} \tag{2}$$

Here, the subsequent token T_i is then sampled from this updated distribution, ensuring compliance with the predefined constraints.

The sequential implementation ensures compatibility with off-the-shelf models that provide access to α, such

as OpenAI's GPT series and HuggingFace Transformers [1].

Consider the following example illustrating the inner workings of this mechanism: Suppose we have a JSON schema requiring a mix of string inputs (for zone names) and floating-point numbers (for aspect ratios of building zones). Let the entire vocabulary of the LLM consist of the strings `"Zone1"`, `"Zone2"`, `"5.7"`, `"8.3"`, and `"Zone3"`.

```
{"zoneName": "String",
 "aspectRatio": "FloatingPoint"}
```

When generation begins, non-string values for the `"zoneName"` field are masked, meaning `"5.7"` and `"8.3"` are masked, leaving only `"Zone1"`, `"Zone2"`, and `"Zone3"` as valid options. If `"Zone1"` is selected, the focus shifts to the `"aspectRatio"` field. Here, only floating-point values are valid, so strings like `"Zone2"` and `"Zone3"` are masked, leaving `"5.7"` and `"8.3"` as the remaining choices.

This process ensures that every element of the generated structure, whether it be a key, value, or data type, aligns precisely with the specified schema.

Model Blueprint

We use JSON as our representation for model blueprints since JSON is both human and machine readable. This allows users to check and correct the input before it is processed by our system.

The blueprint for a building consists of a set of floor plans. The blueprint for a floor plan consists of a set of zones and their relationships. Each zone has a set of properties such as name, area, and aspect ratio. Each zone also defines its topological relationship to other zones in the cardinal directions (north, south, east, and west). We can extract most of this information from an image of the floor plan if it exists. We extract the information using OpenCV that processes floor plans in the form of images to determine where the zones are based on contiguous regions of the same color. This lets us estimate the size, aspect ratio, and topology of the zones.

Additionally, the schema encompasses HVAC details for each zone and the overall building, enabling precise environmental control and energy management simulations. Each zone can specify whether it has an HVAC system, along with the type of system installed. Furthermore, the overall building is characterized by its unique name and an overarching HVAC system type, providing a comprehensive view of the building's climate control architecture. Figure 2 depicts a graph for the building's JSON schema. The blueprint for a district extends the building schema and the topology of their connections. We currently allow for recursive star and branch connections.

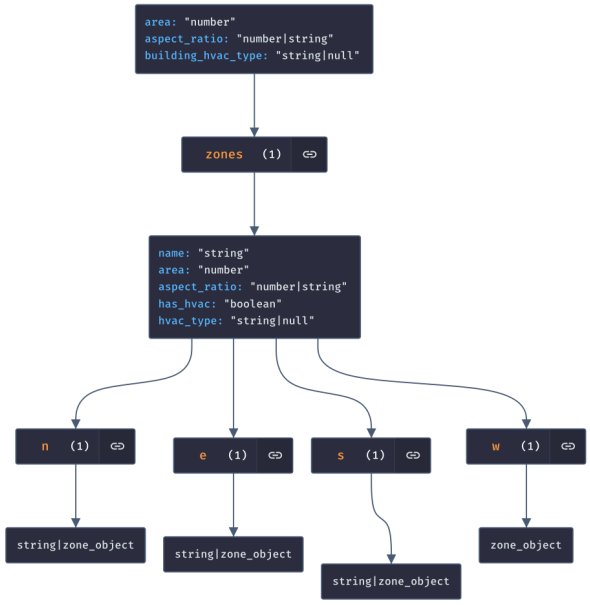
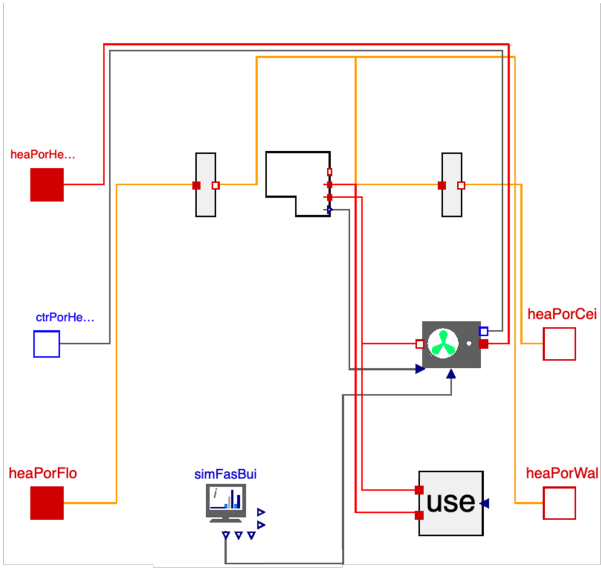


Figure 2: Graph representation of JSON schema for building energy model.

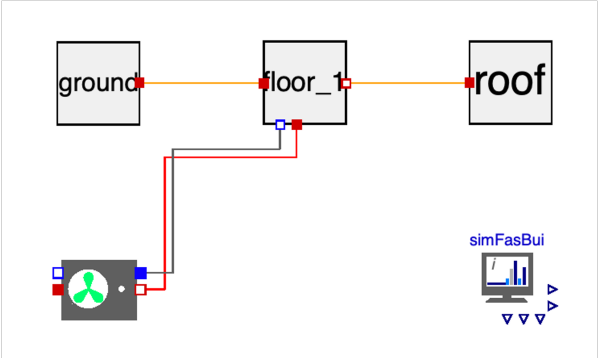
Model Creation

We utilize Python code to convert a JSON blueprint into a Modelica model, employing components from FastBuildings with some custom adjustments. This model represents the building's structure, including floors, zones, walls, and an HVAC system tailored for each zone, designed for ease of use and suitable for various scenarios. Figure 3a illustrates the Modelica model of an individual zone. Central to the model is the FastBuildings zone component, simulating zone capacitance and solar irradiance. The zone's floor and ceiling are depicted by grey rectangles, and it includes a thermostat along with environmental and building usage inputs. Figure 3b displays the complete Modelica model of the building. This models the vertical part of the building and the central HVAC component. The ground is connected to the first floor and the roof is connected to the ceiling of the last floor. If the building has more than one floor, then the ceiling of one floor is connected to the

[1] Practically speaking, APIs like Hugging Face's Logit Processor and OpenAI's Logit Bias offer the capability to algorithmically manipulate logits.



(a) Modelica model of the zone.



(b) Modelica model of the building.

Figure 3: Overview of Modelica models. A detailed floor model is demonstrated in Figure 6.

floor of the next. A floor model is demonstrated in Figure 6. Similarly, this process applies to a district model, with specific variations regarding the details of the base models to accommodate district-scale simulations. One example can be seen in Figure 5.

Parameter estimation

The behavior of the Modelica-based energy system model is defined by components parameters. We fine-tune these parameters such that the simulated output closely resembles the observed data. We first define the loss function

$$f(\mathbf{p}) = \sum_{i=0}^{N} \left(y_{\text{sim}}(t_i, \mathbf{p}) - y_{\text{meas}}(t_i)\right)^2, \tag{3}$$

where $\mathbf{p} \in \mathbb{R}^n$ represents the vector of parameters to be estimated, n is the number of parameters, and N is the number of measurement time points. The optimization problem can be formally expressed as:

$$\min_{\mathbf{p}\in\mathbb{R}^n} f(\mathbf{p}). \tag{4}$$

To simulate the response of a system, we compile the Modelica model into a Functional Mock-up Unit (FMU) (Blochwitz et al. 2011), and then import the FMU into Python. Following the approach of (Gedda et al. 2012; Matei et al. 2023), we adopt a gradient-free optimization technique, namely the Powell algorithm from SciPy (Virtanen et al. 2020), to avoid the complexities of derivative computation[2]. Detailed examples pertaining to the observed data are discussed in the examples section.

Examples

We provide two examples to showcase the proof of concept: one for creating a building energy model, and the other for depicting a district heating system. Additionally, we construct a five-zone model and match its performance using data from an EnergyPlus model. This model, accessible from Energy Plus Prototype Building Models, is used to replicate the thermal dynamics of a real building, thereby confirming our model's ability to perform realistic simulations.

Modeling Generation

We evaluated several Large Language Models (LLMs) and determined that OpenAI's ChatGPT-4 was the most effective for our application. However, as ChatGPT-4 is not open-source, we turned to the HuggingFace open-source model repository for a suitable alternative. We selected Dolly-v2-3b, a model developed by Databricks, considering our computational resource constraints. This 2.8 billion parameter causal language model excels in various capabilities, including brainstorming, classification, and summarization, as outlined in related literature (Ouyang et al. 2022).

For each LLM, we configured the temperature to 1 and limited the token length to 100. Figure 4 shows an example of model generation at two detail levels. In the first scenario, Prompt 1, the LLM uses minimal text information to generate tokens compliant with the JSON schema's type requirements. The model accurately pre-

[2]Although, it is noteworthy that numerical Jacobians may be obtained when exporting Model Exchange FMUs with certain tools, such as Dymola.

Prompt 1: Please provide a typical retail store layout for two zones

Answer:

```
{"building_name": "Downtown Fashion
    Plaza",
  "area": 10000,
  "aspect_ratio": 1.5,
  "building_hvac_type": "Central Air
     Conditioning",
  "zones": [
    {"name": "Fashion and Apparel",
      "area": 6000,
      "aspect_ratio": 1.2,
      "has_hvac": true,
      "hvac_type": "Ducted Split System
          "},
    {"name": "Home Decor and Furnishings
        ",
      "area": 4000,
      "aspect_ratio": 1.8,
      "has_hvac": true,
      "hvac_type": "Ducted Split System
          "}
  ]}
```

Prompt 2: Please provide a typical retail store layout for two zones with a total area of 8178 ft^2 and equal-size zones

```
    { "building_name": "City Center
       Retail Complex"
  "area": 8178,
  "aspect_ratio": 3.87,
  "zones": [
    {"name": "Zone1",
      "area": 4089,
      "aspect_ratio": 7.74,
      "n": ["Zone2"],
      "s": ["exterior"],
      "e": [{"zone2": "exterior", "
          window_area": 0}],
      "w": [{"zone2": "exterior", "
          window_area": 0}]},
    {"name": "Zone2",
      "area": 4089,
      "aspect_ratio": 7.74,
      "n": [{"zone2": "exterior", "
          window_area": 0}],
      "s": ["Zone1"],
      "e": [{"zone2": "exterior", "
          window_area": 0}],
      "w": [{"zone2": "exterior", "
          window_area": 0}]}
  ]}
```

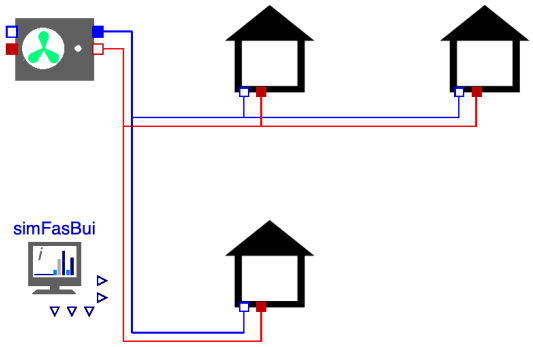
Figure 4: Examples of building energy model blueprint generation from user text-prompt.

Prompt: Please provide a typical district heating system layout for 3 buildings with cool names with a star topology and central HVAC unit

Answer:

```
{name='Downtown Heating District',
    topology=['star',
              {'name': 'hvac', '
                  type_name': '
                  FastBuildingsExtras.
                  HVAC_Central'},
              ['branch',
               {'name': 'SkylineVista',
                   'floor_plans': [
                   one_zone_floor_plan
                   ]},
                {'name': 'UrbanOasis', '
                   floor_plans': [
                   one_zone_floor_plan
                   ]}],
              {'name': 'EcoHaven', '
                  floor_plans': [
                  one_zone_floor_plan
                  ]}]}
```

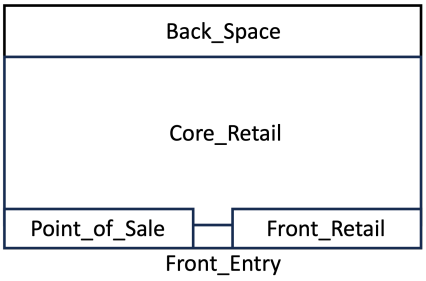
(a) District model blueprint generated from user text-prompt.



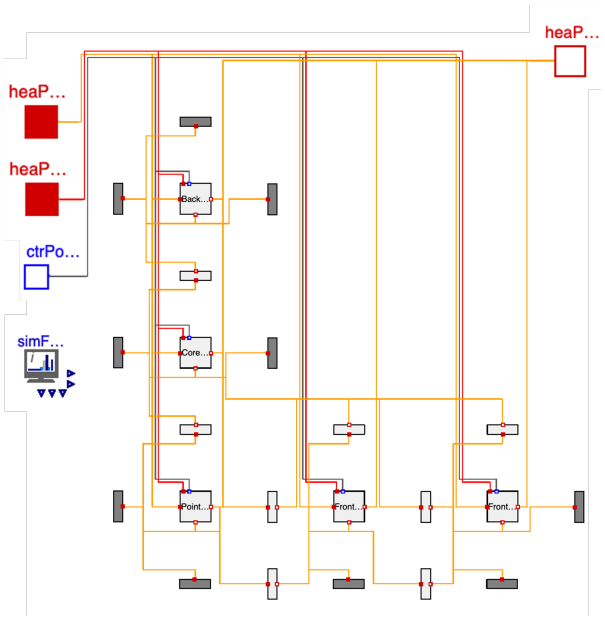
(b) Corresponding Modelica model.

Figure 5: Example of district model development process from user text-prompt.

dicts the overall area and sizes of two zones, with naming conventions possibly reflecting the LLM's training data. In Prompt 2, with more detailed text, the model precisely extracts additional information. For this particular model, the response includes an augmentation of the LLM's output. This augmentation is an image of a two-zone layout, processed using the OpenCV library as de-



(a) Layout of five thermal zones.



(b) Corresponding Modelica model.

Figure 6: Example of Modelica model development for a five-zone system utilizing topological relationships derived from the architectural layout.

scribed in the methodology section. Figure 5 illustrates this process applied to a district model. It demonstrates that the HVAC system is directly connected to home1 and home3, with home1 being daisy-chained to home2.

Model Calibration

We apply the same process to create a Modelica model for a five-zone system, utilizing the architectural layout depicted in 6a to determine each zone's topological relationship to others in the cardinal directions. Due to space constraints, the JSON file for this example is not included. The resulting Modelica model, shown in Figure 6b, features five zones represented as grey boxes. Dark grey rectangles signify exterior walls, while light grey ones indicate interior walls. Red and yellow lines represent HVAC heat transfer and heat conductance, respectively, and black lines depict control signals. For our data source, we use the EnergyPlus reference model of a standalone retail building, with the HVAC system turned off and San Diego weather data applied as the boundary condition. Parameter estimation is performed using temperature data from the core retail zone of the building, trained on one week of data in November and validated for two weeks in December, with a mean absolute error of 0.292 and 0.282 respectively. A total of 15 free parameters were adjusted during the calibration process, including window sizes, initial zone temperatures, thermal properties of walls and floors, and zone heat capacitance. Figure 7 presents these results, with blue lines ('gt') representing the EnergyPlus model output and green lines ('model') depicting the output of the calibrated Modelica model.

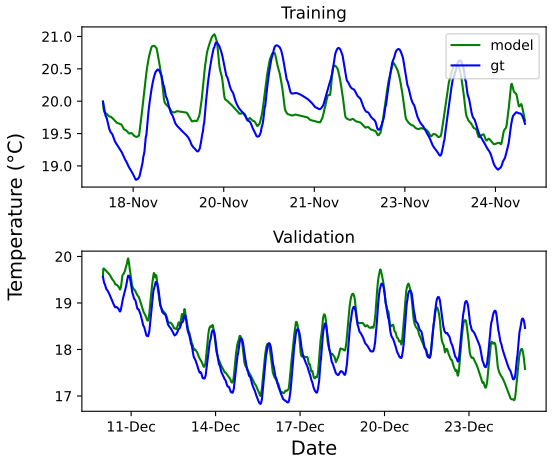


Figure 7: Training and validation using zone temperature for Core Retail zone of five zone reference model. In this representation, blue lines ('gt') signify the output of the EnergyPlus model, while green lines ('model') illustrate the output from the calibrated Modelica model.

Conclusion and Future Work

To bridge the gap between the accessibility of open-source building and district energy modeling libraries and their use by non-modeling expert practitioners, we have presented a methodology using LLMs to parse unstructured user input into simulatable models. To ensure consistency in model generation and to mitigate undesirable artifacts from LLMs, we adopted a grounding approach that constrains the output of LLMs into a schema

with type constraints. The generative nature of LLMs also aids in supplementing missing information in the input. This constrained token generation was applied using pre-trained open-source models, and the output is then parsed into annotated Modelica models through a Python program. Such an approach eliminates the cumbersome intermediate steps typically associated with energy system modeling.

Limitations: While our approach demonstrates the promise of LLMs for building and district energy modeling, it's important to acknowledge the following limitations and areas for ongoing research:

- Geometric constraints: We are currently exploring methods to enforce additional model properties, such as geometrical constraints within LLMs. This will streamline the process and eliminate the need for image-based parsing.
- Contextual understanding: We are working to enhance contextual relevance by using a Retrieval-Augmented Generation (RAG) approach (Lewis et al. 2020). This integrates a database of relevant building documents and metadata, allowing the LLM to reference accurate data and ensure factually grounded outputs.
- Real-World validation: To demonstrate the practical effectiveness of our approach, we plan to develop a Modelica model of a multi-zone commercial building using data from a commercial testbed.

In terms of applications, we aim to further develop this methodology for use in model-based control, diagnosis, and design. In control, this will assist in creating digital twins of building and district models for computing control policies using methods such as Model Predictive Control (MPC) (Mostafavi, Cox, and Futrell 2019; Mostafavi et al. 2022; Mostafavi et al. 2023; Sharma et al. 2023). For diagnostic purposes, the use of a nominal model of a plant facilitates fault detection. In this context, faults can be deliberately introduced into Modelica models, serving as training scenarios for machine learning algorithms to recognize and handle various fault conditions (Matei et al. 2020). Additionally, for design, simulation-based optimization methods will be pivotal in developing sustainable design strategies for building energy systems.

References

Baetens, Ruben, Roel De Coninck, Filip Jorissen, Damien Picard, Lieve Helsen, and Dirk Saelens. 2015. "Openideas-an open framework for integrated district energy simulations." *Building simulation*.

Blochwitz, Torsten, Martin Otter, Martin Arnold, Constanze Bausch, Christoph Clauß, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, Manuel Monteiro, Thomas Neidhold, et al. 2011. "The functional mockup interface for tool independent exchange of simulation models." *Proceedings of the 8th international Modelica conference*. Linköping University Press, 105–114.

Brohan, Anthony, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. "Rt-1: Robotics transformer for real-world control at scale." *arXiv preprint arXiv:2212.06817*.

Fuchs, Marcus, Jens Teichmann, Moritz Lauster, Peter Remmen, Rita Streblow, and Dirk Müller. 2016. "Workflow automation for combined modeling of buildings and district energy systems." *Energy* 117:478–484.

Gedda, Sofia, Christian Andersson, Johan Åkesson, and Stefan Diehl. 2012. "Derivative-free parameter optimization of functional mock-up units." *9th International Modelica Conference. Munich*.

Hinkelman, Kathryn, Jing Wang, Wangda Zuo, Antoine Gautier, Michael Wetter, Chengliang Fan, and Nicholas Long. 2022. "Modelica-based modeling and simulation of district cooling systems: A case study." *Applied Energy* 311:118654.

Hong, Tianzhen, Yixing Chen, Xuan Luo, Na Luo, and Sang Hoon Lee. 2020. "Ten questions on urban building energy modeling." *Building and Environment* 168:106508.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in Neural Information Processing Systems* 33:9459–9474.

Matei, Ion, Alexander Feldman, Johan de Kleer, and Alexandre Perez. 2020. "Real time model-based diagnosis enabled by hybrid modeling." *Annual Conference of the PHM Society*, Volume 12. 10–10.

Matei, Ion, Maksym Zhenirovskyy, John Maxwell, and Johan de Kleer. 2023. "An optimization-based

approach to automated design." *arXiv preprint arXiv:2302.08428*.

Mostafavi, Saman, Robert Cox, and Benjamin Futrell. 2019. "Model Development for Robust Optimal Control of Building HVAC." *Building Simulation 2019*, Volume 16. IBPSA, 3109–3117.

Mostafavi, Saman, Harish Doddi, Krishna Kalyanam, and David Schwartz. 2022. "Nonlinear Moving Horizon Estimation and Model Predictive Control for Buildings with Unknown HVAC Dynamics." *IFAC-PapersOnLine* 55 (41): 71–76.

Mostafavi, Saman, Chihyeon Song, Aayushman Sharma, Raman Goyal, and Alejandro E. Brito. 2023, September. "Benchmarking model predictive control algorithms in Building Optimization Testing Framework (BOPTEST)." *Proceedings of Building Simulation 2023: 18th Conference of IBPSA*, BS 2023. IBPSA.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. "Training language models to follow instructions with human feedback." *Advances in Neural Information Processing Systems* 35:27730–27744.

Remmen, Peter, Moritz Lauster, Michael Mans, Marcus Fuchs, Tanja Osterhage, and Dirk Müller. 2018. "TEASER: an open tool for urban energy modelling of building stocks." *Journal of Building Performance Simulation* 11 (1): 84–98.

Sharma, Aayushman, Raman Goyal, Saman Mostafavi, Shamus Li, Alejandro E Brito, and Eric Bier. 2023, September. "A data-driven method to control buildings sub-zones to spatio-temporal profiles." *Proceedings of Building Simulation 2023: 18th Conference of IBPSA*, Volume 18 of *Building Simulation*. Shanghai, China: IBPSA, 3363–3370.

Thakur, Shailja, Baleegh Ahmad, Hammond Pearce, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri, and Siddharth Garg. 2023a. "Verigen: A large language model for verilog code generation." *arXiv preprint arXiv:2308.00708*.

Thakur, Shailja, Jason Blocklove, Hammond Pearce, Benjamin Tan, Siddharth Garg, and Ramesh Karri. 2023b. "Autochip: Automating hdl generation using llm feedback." *arXiv preprint arXiv:2311.04887*.

Virtanen, Pauli, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. "SciPy 1.0: fundamental algorithms for scientific computing in Python." *Nature methods* 17 (3): 261–272.

Wetter, Michael, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang. 2014. "Modelica buildings library." *Journal of Building Performance Simulation* 7 (4): 253–270.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. "Huggingface's transformers: State-of-the-art natural language processing." *arXiv preprint arXiv:1910.03771*.

Wu, Tongshuang, Michael Terry, and Carrie Jun Cai. 2022. "Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts." *Proceedings of the 2022 CHI conference on human factors in computing systems*. 1–22.

Wüllhorst, Fabian, Laura Maier, David Jansen, Larissa Kühn, Dominik Hering, and Dirk Müller. 2022. "BESMod-A Modelica Library providing Building Energy System Modules." *Modelica Conferences*. 9–18.