

Lawrence Berkeley National Laboratory

LBL Publications

Title

An online machine learning multi-resolution optimization framework for energy system design limit of performance analysis

Permalink

<https://escholarship.org/uc/item/6x07v3jc>

Journal

Engineering Applications of Artificial Intelligence, 182

ISSN

0952-1976

Authors

Amusat, Oluwamayowa O

Grbic, Luka

Patureau, Remi

et al.

Publication Date

2026-10-01

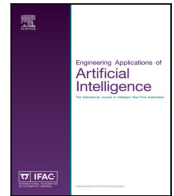
DOI

10.1016/j.engappai.2026.116080

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>

Peer reviewed



Research paper

An online machine learning multi-resolution optimization framework for energy system design limit of performance analysis

Oluwamayowa O. Amusat^{a,1}, Luka Grbcic^{b,*,1}, Remi Patureau^{c,2}, M. Jibran S. Zuberi^d, Dan Gunter^a, Michael Wetter^c

^a Scientific Data Division, Computing Sciences, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA 94720, United States of America

^b Applied Mathematics & Computational Research Division (AMCR), Computing Sciences, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA 94720, United States of America

^c Building & Industrial Energy Systems (BIES) Division, Energy Technologies Area, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA 94720, United States of America

^d Energy Analysis Division, Energy Technologies Area, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA 94720, United States of America



ARTICLE INFO

Dataset link: <https://github.com/lbl-srg/pbd-industrial-limit-of-performance>

Keywords:

Online machine learning
Multi-resolution optimization
Energy systems
Platform-based design
Process control
Limit of performance

ABSTRACT

Designing reliable integrated energy systems for industrial processes requires optimization and verification models across multiple fidelities, from architecture-level sizing to high-fidelity dynamic operation. However, model mismatch across fidelities obscures the sources of performance loss and complicates the quantification of architecture-to-operation performance gaps. We propose an online, machine-learning-accelerated multi-resolution optimization framework that estimates an architecture-specific upper bound on achievable performance while minimizing expensive high-fidelity model evaluations. We demonstrate the approach on a pilot energy system supplying a 1 megawatt industrial heat load. First, we solve a multi-objective architecture optimization to select the system configuration and component capacities. We then develop an machine learning (ML)-accelerated multi-resolution, receding-horizon optimal control strategy that approaches the achievable-performance bound for the specified architecture, given the additional controls and dynamics not captured by the architectural optimization model. The ML-guided controller adaptively schedules the optimization resolution based on predictive uncertainty and warm-starts high-fidelity solves using elite low-fidelity solutions. Our results on the pilot case study show that the proposed multi-resolution strategy reduces the architecture-to-operation performance gap by up to 42% relative to a rule-based controller, reduces the required high-fidelity model evaluations by 34% relative to the same multi-fidelity approach without ML guidance, and improves on direct derivative-free optimization by up to 24%, enabling faster and more reliable design verification. Together, these gains make high-fidelity verification tractable, providing a practical upper bound on achievable operational performance prior to industrial deployment.

1. Introduction

Integration across mechanical and electrical systems is a key enabler to achieve cost-effective, grid-flexible energy systems as such integration allows optimizing the size of equipment for storing and converting energy across energy carriers under consideration of dynamic energy rates. For example, for given utility rates and load profiles, one could trade off investments in batteries and thermal storage tanks for energy storage, and heat pumps and gas boilers for heat production. The overall techno-economic performance depends on both how equipment

is sized and how it is operated, requiring a co-optimization across electrical, mechanical, and controls domains.

There are two general approaches to this co-optimization challenge. Either one uses simplified process and control models to formulate an optimization problem that can be solved efficiently and robustly, such as a linear programming problem or a nonlinear programming problem. These types of models assume perfect knowledge of the system and the future, and ignore fast transients and control switches that would lead to non-differentiable functions. We will call these types of models *optimization models*. Alternatively, one may build detailed process and control models, such as by formulating a differential algebraic system

* Corresponding author.

E-mail address: lgrbcic@lbl.gov (L. Grbcic).

¹ Authors contributed equally to this work.

² Work done while employed at Lawrence Berkeley National Laboratory.

of equations with continuous-time and discrete event dynamics. Such models serve well for verification of a design, but determining equipment sizes and control schedules can only be optimized heuristically, and their simulation can be computationally expensive, sometimes making even heuristic optimizations impractical. We will call these models *verification models*.

For complex energy systems such as a power plant that provides heat for an industrial process or cooling for a building, district, or data center, there is no model that can serve as both optimization and for verification (Wetter and Sulzer, 2024). Platform-Based Design (PBD) is a process that formalizes how to transition between models of different fidelity, allowing holistic design space exploration and de-risking implementation of integrated systems (Sangiovanni-Vincentelli, 2007; Sulzer et al., 2023). However, as the optimization model and the verification model are different, they have an inherent model mismatch and corresponding performance gap. The different models will have different approaches and assumptions, such as perfect knowledge of the process and the future in optimization models compared to only measurable quantities and imperfect predictions with closed loop stability constraints and operational constraints for verification models. As one progresses in the design from the optimization to the implementation with verification, it is not clear what part of the performance gap is due to mismatch of the process models and due to implementation of the control. However, knowing the performance gap due to control implementation is important, as such knowledge allows designers to judge whether the control is near-optimal, and further attempts to improving it would be fruitless. While engineers and designers are aware of this performance gap between optimization model and verification model, quantification of this gap, which is critical for decision making, is typically impractical.

Addressing this knowledge gap is the core contribution of the paper. One potential strategy involves a direct brute-force optimization of the verification model. However, the optimal control problem is a constrained, non-differentiable black-box of high dimensionality, which makes the problem computationally expensive and achieving a competitive result difficult at best. An exhaustive search is typically required to close the performance gap with optimal solutions.

The limit of performance estimation problem we address is a novel problem in itself, but it involves solving optimal control problems for the energy system (a well-established problem). Thus, a natural question is whether established methods for energy system control suffice. The defining requirement of the limit of performance analysis is that it be posed on the verification model itself, so that the control-implementation gap can be isolated from process-model mismatch. This immediately rules out model predictive control (MPC) and other gradient-based optimal control methods: because the verification model is non-differentiable, they would require a simplified surrogate model, reintroducing the very mismatch that limit of performance analysis is designed to isolate. Standard ML-MPC inherits the same flaw, and is further impractical here because the year-long horizon with season-varying dynamics makes a single global surrogate prohibitively expensive to train to sufficient accuracy, while the hard operational constraints are difficult to enforce through a learned surrogate (Hewing et al., 2020). Reinforcement learning (RL) avoids an explicit surrogate but scales poorly to our high-dimensional state and control space; and while safe RL can accommodate constraints, it remains sensitive to distribution shift and to reward design, both of which are problematic over a year-long horizon with changing seasonal conditions (Su et al., 2025). Surrogate-assisted optimization, such as Bayesian optimization, is highly effective for the low-dimensional design and sizing problem (Booker et al., 1999), but suffers from the curse of dimensionality and does not scale to the high-dimensional control problem (Shahriari et al., 2015), where constraint enforcement and the cost of generating a year of training data compound the difficulty. Limited work exists on the development of methods that are simultaneously able to operate on the non-differentiable verification model, enforce

hard constraints, and remain computationally tractable at this scale. To overcome these challenges, we propose an integrated framework that leverages multi-resolution optimal control and machine learning (ML) for further acceleration. This approach aims to improve both the tractability and computational efficiency of limit of performance estimation problems.

Multi-resolution optimal control approaches have been shown to be beneficial for improving the tractability of general optimal control problems in power and energy storage systems (Abdulla et al., 2017; Jain and Tsiotras, 2008; Zhang et al., 2023; Zou et al., 2018). The fundamental principle of multi-resolution optimal control is to leverage optimal control sequences from a coarse-grained discretization to inform the optimization process at a finer-grained level (Polak, 1997; Jain and Tsiotras, 2008). This strategy is designed to converge towards a solution with fewer evaluations of the computationally expensive function.

Similarly, ML techniques have been shown to be able to accelerate the solution of optimal control in energy systems problems, and decision-making in engineering problems in general (Perera et al., 2019; Bre et al., 2020; Prina et al., 2024; Stoffel et al., 2023; Rätz et al., 2024; Wissmann et al., 2026; Xiao et al., 2026; Yin and Xiang, 2025a,b). More specifically, optimal control in energy systems heavily relies on ML methods such as Deep RL for rapid, policy-based decision-making under uncertainty (Li et al., 2023; Michailidis et al., 2025). Furthermore, ML techniques enhance classical control frameworks, such as utilizing algorithms based on Koopman theory to establish accurate, simplified models for MPC in complex components like Pumped Storage Systems (Cai et al., 2021), while hybrid solutions like Reinforced Predictive Control (RL-MPC) are emerging to ensure constraint satisfaction and continuous learning in building energy management (Arroyo et al., 2022). For energy system design problems, ML surrogate models are employed to emulate high-fidelity energy system simulation tools, serving as fast, reliable replacements for costly iterative calculations (Rulff and Evins, 2025; Lédée et al., 2025). To improve the efficiency of black-box optimization, ML surrogate models are deployed in two primary ways: either to provide a good initial guess to warm start the main optimization algorithm (Grbic et al., 2024; Favaro et al., 2024), or as a computationally efficient replacement for the original black-box function, thereby conserving computational resources (Liang et al., 2024; Liu et al., 2023).

Furthermore, in the field of online trained ML models applied to energy system problems, Stoffel et al. (2023) introduce a methodology that integrates online learning and novelty detection in data-driven MPC to enhance the reliability and safety of building energy systems by continuously retraining models and detecting extrapolation events to switch to robust fallback controllers when necessary. Gao et al. (2025) present an online optimization method for integrated energy systems that combines deep learning for accurate forecasting of wind and photovoltaic power generation with an accelerated optimization algorithm. Sha et al. (2025) introduce a data-driven MPC strategy that employs an online learning-enhanced encoder-decoder ML model to optimize HVAC performance in residential buildings.

While existing research has separately applied multi-resolution methods to improve tractability and ML techniques to accelerate optimization, these approaches have not been applied to limit of performance problems in energy system design, or synergistically integrated into a unified framework capable of solving high-dimensional, black-box optimal control problems efficiently. Our work makes four main contributions to the state-of-the-art in energy system and controls co-optimization, and in limit of performance analysis:

- We introduce the limit-of-performance estimation problem for industrial energy system design and, to the best of our knowledge, provide its first formal treatment.

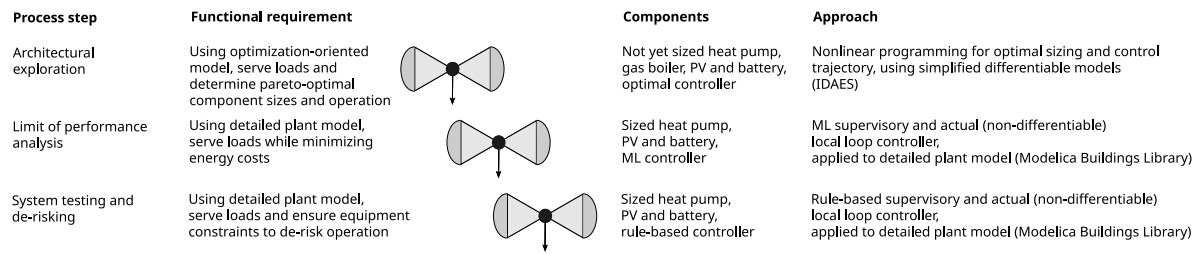


Fig. 1. Three platform layers used in the Platform-Based Design of the industrial plant and its controller.

- We introduce a novel, ML-accelerated, multi-resolution framework for efficient numerical approximation of the limit of performance in energy systems whose transient evolution is specified in differential-algebraic system of equations whose state trajectories are not differentiable in the control schedule. This integrated approach improves tractability over direct optimization and reduces computational cost by intelligently replacing an expensive, long-horizon optimization stage with an online-trained ML surrogate. As demonstrated in Section 5.2, when applied specifically to our case study, this framework establishes a performance benchmark that reveals a potential for over 10% in operational cost savings compared to a standard rule-based controller.
- We demonstrate, through an ablation study, the synergistic effect of the framework's core components. We show that both the multi-resolution structure (which provides strategic guidance) and the elite solution seeding (which preserves high-quality prior solutions) are independently critical for achieving robust convergence and computational efficiency.
- We present a novel supervisory control logic that leverages the ML model's uncertainty quantification to intelligently balance computational cost and solution quality. This uncertainty-aware mechanism ensures both computational efficiency, by using the fast ML model when it is confident, and robustness, by triggering a full optimization to explore and gather new data when uncertainty is high or conditions change.

We demonstrate our approach to the limit of performance analysis in energy system and controls co-design of an industrial heating process, for which we design the system and its implementable control using PBD, consisting of three layers of abstraction as shown in Fig. 1: In the first platform layer, we co-optimize electrical and mechanical system together with optimal operating schedules using nonlinear programming. This gives specification for equipment sizes and insight into how to operate the system. For the second and third platform layer, we implement a verification model that is partitioned into a process model, a supervisory controller and local loop controllers. In the second platform layer, we use the process and the local loop control models together with an ML-based supervisory controller to compute a numerical approximation of limit of performance for the system, thereby providing an estimated upper bound for the performance of the verification model under the assumption that the limit of performance analysis found the global optimal control sequence. In the third platform layer, we replace the ML-based controller with a rule-based supervisory controller, and compare its performance against the estimated limit of performance computed using the ML-based supervisory controller applied to the same model of the process and the local loop controllers. The rule-based controllers ensure that practical constraints such as equipment lockout and operating envelopes are maintained. We note that while we used a rule-based controller, the limit of performance analysis does not depend on that choice; the same process could have been done using for example a robust model predictive controller instead of the rule-based controller in which case one would see how much performance has been left on the table due to provision of robustness guarantees.

The estimation of the limit of performance is achieved through our novel ML-accelerated, multi-resolution optimal control framework. The foundation of the framework is a multi-resolution strategy, where an initial, coarse-grained “exploratory” optimization over a long horizon determines a strategic terminal target value (e.g. battery State of Charge), which then guides subsequent, finer-grained optimization stages. Our core contribution is to accelerate this process by training an online ML model with historical forecast data to predict this terminal target value based on current forecasts. Crucially, the ML model also provides an estimate of its own prediction uncertainty, which is central to the algorithm's performance.

The ML-based controller provides no optimality guarantees, owing to the non-differentiable nature of the verification model and the reliance on a stochastic, derivative-free optimization algorithm. However, since the verification model offers no analytical structure or gradient information, gradient-based or formally-guaranteed optimization methods are inapplicable, making this stochastic, derivative-free approach the most viable option available. Nonetheless, the resulting controller provides a practical limit of performance estimate, offering a meaningful starting point against which other control strategies can be evaluated.

The rest of the paper is structured as follows: Section 2 presents a description of the industrial heating system that will be the focus of this work. Section 3 presents at a high level the nonlinear programming optimization formulation for the architectural exploration and system sizing problem. Next, we briefly describe the verification model, followed by a detailed description of our proposed ML-Accelerated optimal control framework (Section 4). Section 5 presents a detailed analysis of our results for the case study, covering both the architectural optimization and optimal control problems. This is followed by a brief discussion of the limitations of the proposed approach, and potential future research directions to expand this work (Section 7).

2. Pilot system description

In this section, we describe the energy system and the control intent that is used for the subsequent co-optimization of the energy and control system.

Fig. 2 shows the schematic diagram of the system. The objective is to serve an industrial heat load $\dot{Q}_l(\cdot)$ daily from 8am to 8pm. The industrial process is controlled using a heat exchanger that requires on the plant side to maintain a supply temperature of 90°C and a return temperature of 70°C. The energy and control system should be designed to minimize total annualized cost and energy use, subject to meeting the load at the specified temperatures.

The energy system shown in Fig. 2 is as follows: Hot water can be produced either by a heat pump which uses as its source temperature a district energy system whose temperature varies between 8°C and 16°C, or a gas boiler. The heat pump has internal control to track the leaving evaporator water outlet temperature, subject to sufficient capacity, and the heat provided by the gas boiler is modulating. The heat pump evaporator loop has a variable speed pump to track an evaporator temperature difference of $\Delta T = 4$ K. On the heat supply side, both heat production units have on/off pumps and three-way valves

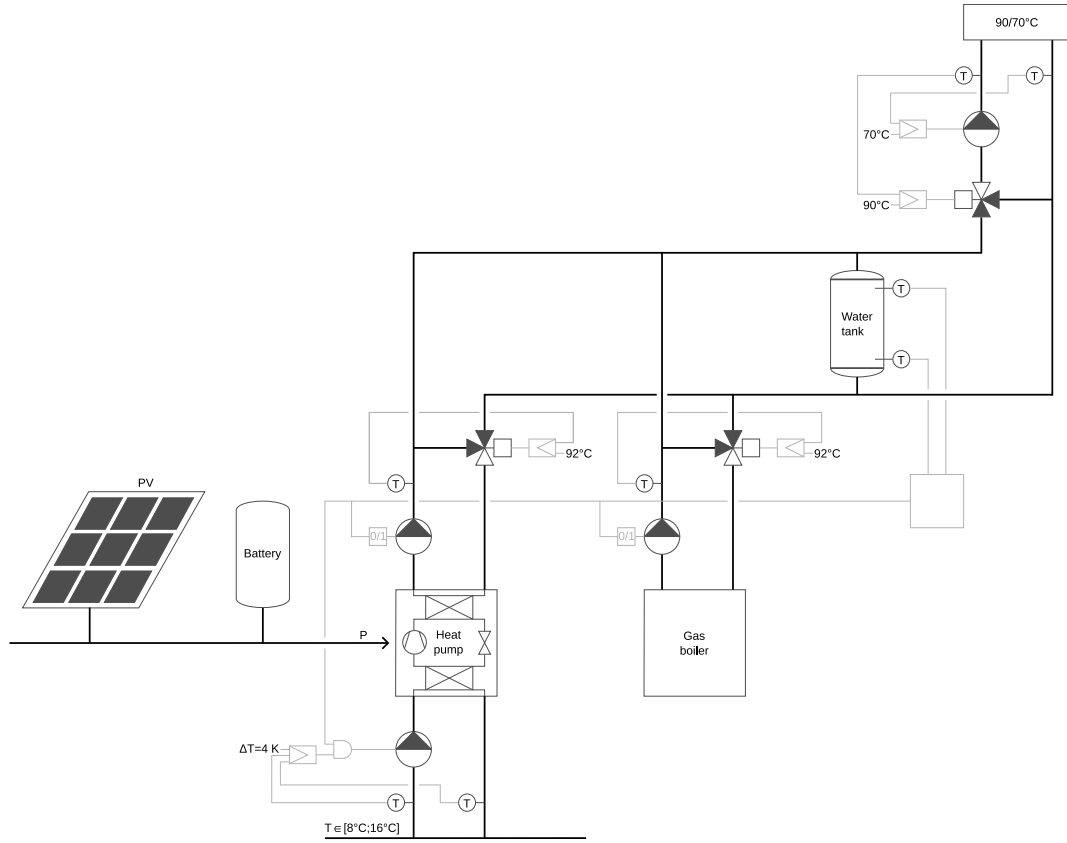


Fig. 2. Schematic diagram of the energy system.

that regulate the leaving temperature to at least 92°C. The heat can either be directly used by the load or stored in a stratified water tank. The circulation pump at the load is a variable speed pump, controlled to track the required return temperature. Electricity can be consumed from or fed back to the electrical grid. The plant has a PV array and a battery that allows storing electricity for the heat pump.³

To protect equipment from short-cycling and associated wear-out, we enforce the following lockout constraints based on engineering judgment: If the heat pump is switched on, it must operate for at least 30 min, and when it is switched off, it must remain off for at least 15 min. If the gas boiler is switched on, it must remain on for at least 15 min, but there is no minimum lockout time.

For the solar irradiation, TMY3 weather data (Wilcox and Marion, 2008) for the Sacramento international airport are used.

3. Architectural exploration via nonlinear programming

For the system described above, we determine the optimal architecture and component sizes for the energy system by solving a nonlinear programming (NLP) optimization problem. Given input data on the thermal load demand, solar irradiation, and electricity and gas costs, we solve a multi-objective optimization problem to minimize the total annualized cost $TAC(\mathbf{x})$ and total energy imported $TE(\mathbf{x})$,

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) = TAC(\mathbf{x}) = C_{\text{cap}}(\mathbf{x}) \cdot \text{CRF} + C_{\text{op}}(\mathbf{x}) \\ f_2(\mathbf{x}) = TE(\mathbf{x}) = \int_{t_0}^{t_f} E_{\text{grid}}(t, \mathbf{x}) dt + \int_{t_0}^{t_f} Q_{\text{boiler}}(t, \mathbf{x}) dt \end{bmatrix}, \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, \quad i \in \{1, \dots, m\}, \\ & h_j(\mathbf{x}) = 0, \quad j \in \{1, \dots, z\}, \end{aligned} \quad (1)$$

³ Compared to the required heat energy, the energy of the circulation pumps is negligible and hence not accounted for in the optimization.

where $\mathbf{x}$ is the vector of decision variables representing the capacities of the individual units and the variables defining the operation of each equipment at each point in time; $C_{\text{cap}}(\mathbf{x})$ and $C_{\text{op}}(\mathbf{x})$ are the capital and operating costs, respectively; CRF is the capital recovery factor; $E_{\text{grid}}(t, \mathbf{x})$ is the energy purchased from the grid at time t ; and $Q_{\text{boiler}}(t, \mathbf{x})$ is the energy generated from the natural gas imported for the boiler at time t . Equality constraints $h_j(\mathbf{x})$ on the optimization problem include constraints enforcing the mass and energy balances within the system, performance equations, and mechanistic models of the main energy system components (PV, gas boiler, heat pump, battery, storage tank, load). Inequality constraints $g_j(\mathbf{x})$ include the thermal load satisfaction and other operational constraints, as well as bounds on the generation and storage capacities of the different system components. Details about the process and costing models for the individual components are presented in Appendix A.

The architectural optimization model is implemented and solved using the IDAES Integrated Platform (Lee et al., 2021), an algebraic modeling framework with advanced mathematical tools for solving large scale optimization problems within the Python ecosystem. We adopt the multi-period modeling approach (Rawlings et al., 2022; Rao et al., 2024), which converts a dynamic optimization problem into a series of discrete models linked together over a time horizon with rate-limiting constraints. This is accomplished by approximating the dynamic behavior of the system components by algebraic equations via a forward-difference discretization scheme, enabling implementation in IDAES with an hourly time horizon. With this multi-period modeling approach, each time step in the horizon is implemented as a separate algebraic model, and the different models are connected together by user-defined set of variables and constraints and solved simultaneously. For all generation and storage units k , we implement linking constraints that force the installed capacities S of the units to be the same across

all time periods,

$$S_{k,t} = S_{k,t-1}, \quad t \in \{1, 2, \dots, N\}. \quad (2)$$

For the storage units (battery, tank), linking constraints also connect the final state of charge of the previous time horizon to the starting state of charge of the current time horizon,

$$E_{k,t,start} = E_{k,t-1,end}, \quad t \in \{1, 2, \dots, N\}; k \in \{\text{battery, storage tank}\}. \quad (3)$$

The architectural optimization problem is formulated as a boundary value problem, with constraints that enforce that the storage levels for the battery and heat tank at the start and end of operation are equal,

$$E_{k,1,start} = E_{k,N,end}, \quad k \in \{\text{battery, storage tank}\}. \quad (4)$$

We solve the multi-objective problem with the ϵ -constraint method, converting the single multi-objective problem into a series of single-objective cost minimization problems in ϵ ,

$$\begin{aligned} \min_{\mathbf{x}} \quad & \text{TAC}(\mathbf{x}) \\ \text{s.t.} \quad & \text{TE}(\mathbf{x}) \leq \epsilon, \\ & g_i(\mathbf{x}) \leq 0, \quad i \in \{1, \dots, m\} \\ & h_j(\mathbf{x}) = 0, \quad j \in \{1, \dots, z\}. \end{aligned} \quad (5)$$

with the range of ϵ ($TE_{LB} \leq \epsilon \leq TE_{UB}$) determined by solving two single objective problems minimizing the total annualized cost TE_{UB} and total imported energy TE_{LB} , respectively.

We solved the NLP architectural optimization problem with IPOPT (Wächter and Biegler, 2006). The discretized nonlinear system of algebraic equations for the full year ($t = 1, 2, \dots, 8760\text{h}$) consists of over 1.4M equations and constraints, with 35,040 degrees of freedom (four per time step).

The solution of the architectural optimization problem is a set of Pareto-optimal points that are a trade-off between TAC and imported energy from the grid. As part of the optimization solution, the model also returns an idealized operating strategy for each design that minimizes the operating costs. However, the architectural optimization problem assumes perfect knowledge of the future (i.e., solar irradiation and utility rates are known for the whole year a priori), does not capture full dynamics such as startup, shutdown, lockouts, and fast transients, and ignores all non-differentiable control implementations such as actuator saturation and switches between mode of operations. Thus, while the optimal control profiles obtained at the architectural optimization level will capture the slow dynamics of energy storage well, they cannot be applied to verify operation and controls, and de-risking is necessary using verification models that capture all these intricacies. However, the idealized control trajectories enable us to optimally size equipment under consideration of the dynamic system operation, and they provide a valuable insights for operating the systems.

4. Energy system control

The architectural optimization process described in the previous section determines the optimal unit capacities for the different components of the energy system. Given those capacities, the next task is to design control logic that gets the best possible performance out of the specified architecture, given the additional controls and dynamics not captured by the optimization model. For this, we use a verification model of the system implemented in Modelica, an object-oriented, equation-based modeling language (Mattsson and Elmqvist, 1997). The verification model effectively represents a digital representation of the actual process. In this section, first, we describe the Modelica model. We follow that with a description of a rule-based controller, which provides a controller as is typical in such thermal systems. Finally, we provide details of our proposed ML-accelerated optimal control strategy which is used to approximate the limit of performance. This estimated limit of

Table 1
Modelica model components.

Component	Modelica model
Heat pump	<i>Buildings.Fluid.HeatPumps.Carnot_TCon^a</i>
Gas boiler	<i>Buildings.Fluid.Boilers.BoilerPolynomial^b</i>
Thermal storage	<i>Buildings.Fluid.Storage.StratifiedEnhanced^c</i>
Battery	<i>Buildings.Electrical.DC.Storage.Battery^d</i>
PV	<i>Buildings.Electrical.DC.Sources.PVSimple^e</i>

^a https://simulationresearch.lbl.gov/modelica/releases/v12.0.0/help/Buildings_Fluid_HeatPumps.html.

^b https://simulationresearch.lbl.gov/modelica/releases/v12.0.0/help/Buildings_Fluid_Boilers.html.

^c https://simulationresearch.lbl.gov/modelica/releases/v12.0.0/help/Buildings_Fluid_Storage.html.

^d https://simulationresearch.lbl.gov/modelica/releases/v12.0.0/help/Buildings_Electrical_DC_Storage.html.

^e https://simulationresearch.lbl.gov/modelica/releases/v12.0.0/help/Buildings_Electrical_DC_Sources.html.

performance allows us to quantify how much performance is lost due to the rule-based controller relative to the highest performing controller that we were able to develop, but which may not be sufficiently robust for an actual implementation.

4.1. Modelica model

Two models were developed in Modelica using the Modelica Buildings Library (Wetter et al., 2014). Modelica is an equation-based, object-oriented language (Mattsson et al., 1999) which can be used to simulate models that couple thermal systems, fluid flow systems, electrical systems and control. Modelica can properly resolve the continuous time dynamics, discrete time dynamics and event-driven dynamics that is needed for verification of such coupled physical and control systems. In Modelica, components are encapsulated models that expose interface variables, such as mass flow rate, pressure and enthalpy of a pipe, through ports, and through these ports, components can be connected to assemble a system. For our application, the Modelica model results in a differential algebraic system of equations that is governed by continuous dynamics as well as time- and state-events. Thus, the resulting system model is not differentiable in equipment sizes or in control inputs. To translate Modelica models to executable code, we used Dymola 2026 on Linux Ubuntu.

The two models differ as follows: One model is a complete model of the plant with rule-based local loop control, but with supervisory control set points exposed as inputs, allowing the limit-of-performance calculation to vary the supervisory control in order to maximize performance. The exposed control inputs were charging and discharging rates of the battery and the mass flow rate of the heat pump condenser. By controlling the mass flow rate, the limit-of-performance controller can dispatch the equipment. This model is used in the 2nd platform layer shown in Fig. 1. The other model adds rule-based supervisory controls, described in Section 4.2, and is used in the 3rd platform layer. Table 1 summarizes the key Modelica modeling components used in our analysis; further details may be found in Appendix B.

4.2. Rule-based control

The rule-based controller attempts to minimize the operating cost by dispatching the heat pump or boiler, and charging and discharging the battery, based on the time-varying electricity price.

We implemented a rule-based strategy as follows:

When the process load is active (from 8am to 8pm), the heat pump and the gas boiler are commanded on. When the process load is inactive, the heat pump charges the hot water tank during the hours when the electricity cost is the least expensive, provided the electricity cost is below $C_{ele} < 0.13$ \$/kWh. This limit is computed based on the

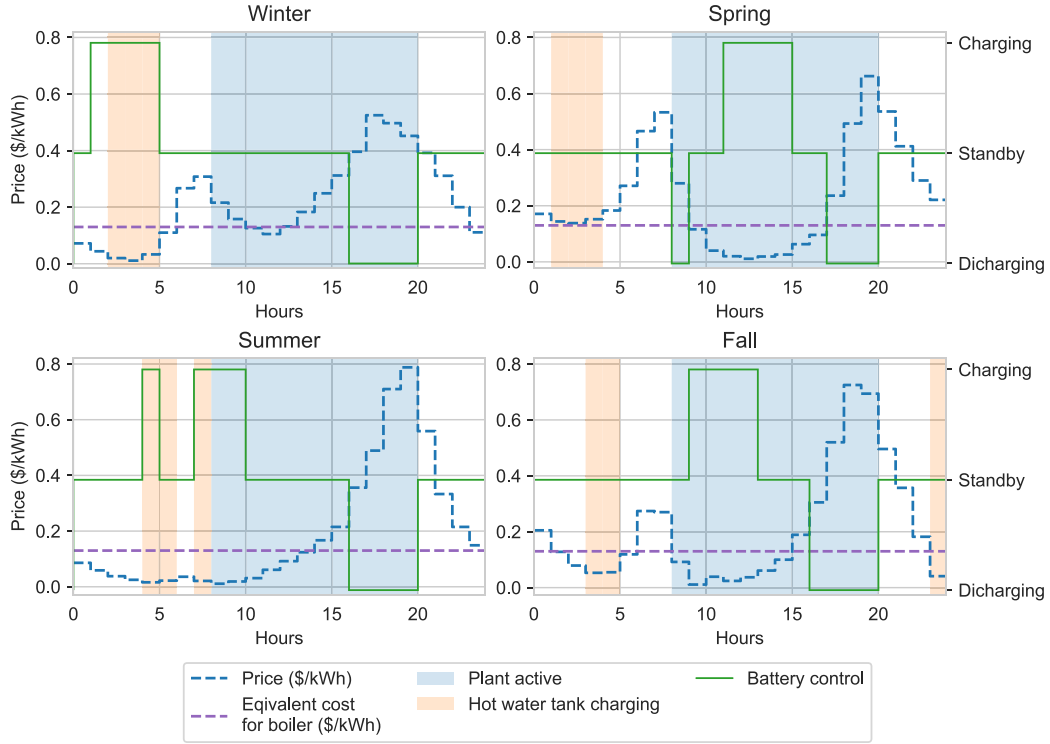


Fig. 3. Charging and discharging of the battery (green), and hot water tank charging (orange). The electricity price and the equivalent for gas are shown with dotted lines.

comparison of gas versus electricity costs, taking into account the gas boiler efficiency and the coefficient of performance of the heat pump as

$$C_{ele} < C_{gas} \frac{COP}{\eta_{boiler}}. \quad (6)$$

The battery operation is based on the hourly electricity prices. When the process load is active (from 8am to 8pm), the battery is discharging during the 4 h when the electricity is the most expensive. The battery is charging during the 4 h when the electricity is the least expensive.

Fig. 3 shows the resulting control strategy.

4.3. ML-accelerated optimal control

This section details a general framework for accelerating a multi-resolution, receding-horizon optimal control problem using a ML model. We first present the generalized mathematical formulation of the optimal control problem. We then describe the ML-accelerated algorithm, which adaptively decides when to leverage a predictive model versus performing a full, computationally expensive optimization. Finally, we provide the specific implementation details and hyperparameters of the framework.

4.3.1. General optimal control formulation

We consider a discrete-time optimal control problem solved in a receding-horizon fashion. The system's evolution is defined over a finite time horizon, which is discretized into H steps of duration Δt .

The state of the system at time step k is represented by a vector $x_k \in \mathbb{R}^n$, and the control inputs are given by $u_k \in \mathcal{U} \subseteq \mathbb{R}^p$. These control inputs are constrained by element-wise lower and upper bounds, represented by vectors $u_{\min}$ and $u_{\max}$, ensuring that the inputs remain within their limits.

The full decision vector for the optimization horizon is the concatenation of all control inputs, $U = [u_0^T, u_1^T, \dots, u_{H-1}^T]^T \in \mathcal{U}^m$, where $m = pH$.

The objective is to find the sequence of control inputs U that minimizes a total cost function, $J(U)$, subject to the system's dynamics

and operational constraints. The general optimal control problem is formulated as

$$\begin{aligned} \min_{U \in \mathbb{R}^m} \quad & J(U), \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k, w_k), \quad \text{for } k \in \{0, \dots, H-1\}, \\ & x_k \in \mathcal{X}, \\ & u_k \in \mathcal{U}, \\ & x_0 \text{ is the known initial state.} \end{aligned} \quad (7)$$

Here, the function $f(\cdot, \cdot, \cdot)$ is a discrete-time, non-differentiable function that describes the system dynamics. The vector w_k represents known exogenous forecasts, such as weather or market prices, that influence the system's evolution. The sets $\mathcal{X}$ and $\mathcal{U}$ represent the feasible states and controls, respectively, enforcing the physical operational limits of the system, including applicable box bounds.

The cost function $J(U)$ is defined as the sum of a stage cost $\ell(x_k, u_k, w_k)$, which is summed over the horizon, and a terminal cost $V_f(x_H)$, which is evaluated at the final state x_H as

$$J(U) = \sum_{k=0}^{H-1} \ell(x_k, u_k, w_k) + V_f(x_H). \quad (8)$$

4.3.2. ML-accelerated multi-resolution strategy

Solving the above optimization problem can be computationally intractable, especially when the dynamics function $f(\cdot, \cdot, \cdot)$ is expensive to evaluate, when it is not continuously differentiable, as is the case for our problem because of local controllers, or when the horizon H is long and requires exhaustive search. To address this, we employ a multi-resolution strategy composed of three distinct stages: a long-horizon exploratory stage ($r = \hat{e}$), a low-resolution stage ($r = \hat{l}$), and a high-resolution stage ($r = \hat{h}$).

- The problem is first solved at the exploratory resolution, which uses a coarse time discretization, i.e., a large Δt , to cover a long total time duration. This allows the optimizer to find a good long-term strategy while keeping the number of decision steps $H_{\hat{e}}$,

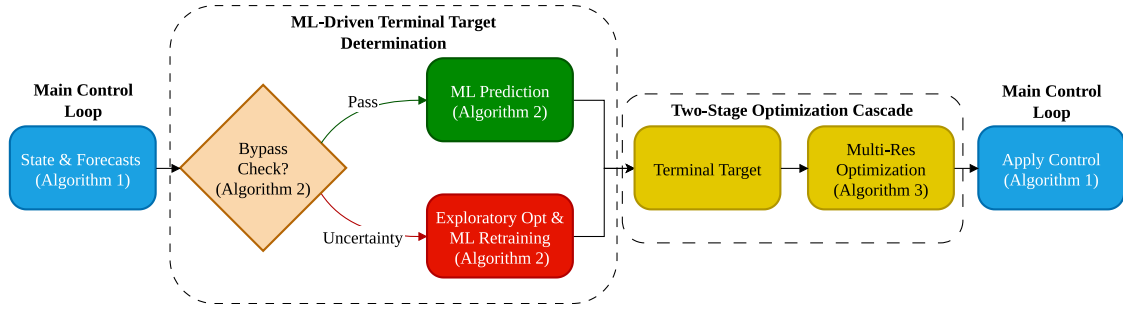


Fig. 4. The flowchart of the ML-Accelerated multi-resolution control strategy.

and thus the problem’s dimensionality, computationally tractable. The goal of this stage is to determine any terminal targets that will be enforced as constraints placed on subsequent optimization stages. Subsequently, this strategy is refined in two stages with progressively finer detail.

- The low-resolution stage ($r = \hat{l}$) uses a smaller time step ($\Delta t_l < \Delta t_{\hat{e}}$) over a shorter total duration, resulting in a new planning horizon $H_{\hat{l}}$.
- Finally, the high-resolution stage ($r = \hat{h}$) uses the finest time step ($\Delta t_h < \Delta t_l$) over the duration for a horizon of $H_{\hat{h}}$ that is equal to $H_{\hat{l}}$.

For all stages, the number of decision steps H_r is chosen to ensure the problem remains computationally tractable.

Even with this three-stage decomposition, the exploratory stage remains one of the primary computational bottlenecks, since its long horizon and coarse search still require many expensive solver evaluations to identify a good terminal target. The core of our proposed method is to use machine learning to accelerate this process by strategically replacing this expensive component: the exploratory optimization used to determine the terminal target. As shown in Fig. 4, at each discrete step the main control loop (Algorithm 1) passes the current state and determines whether to bypass the exploratory optimization using an ML model (Algorithm 3). The decision is made based on several factors, such as how confident the ML model is in its prediction, and the point in the optimization horizon at which the decision is made. If the check passes, the terminal target is supplied by the ML prediction alone; if it does not, the exploratory optimization is run instead, both to produce the terminal target for the current cycle and to generate a new training example that is used to retrain the ML model. Either way, the resulting terminal target then feeds into the two-stage, multi-resolution optimization, whose output is finally applied by the main control loop. In this way, the expensive exploratory search is only performed when needed to maintain accuracy or safety, while the inexpensive ML path is used by default whenever the model is confident.

Beyond this core decision, a few additional mechanisms govern the method’s behavior over successive discrete steps. Each time the exploratory optimization is run, its result is added to a growing set of training examples that keep the ML models current, and its full population of candidate solutions is folded into a cumulative pool (called an “elite pool” henceforth) that helps seed later high-resolution searches, alongside the current cycle’s low-resolution solutions. This accumulated pool and the trained models are valuable for recurring, similar conditions, but can become actively misleading if the underlying price regime shifts – for instance, moving from winter to summer pricing. To guard against this, the algorithm monitors incoming price forecasts for large discontinuities; detecting one triggers a full reset that clears the elite pool and training datasets, temporarily raises the solver’s evaluation budget to encourage broader exploration, and begins a new mandatory “warm-up period” period during which the exploratory optimization is always used, allowing the datasets and elite pool to be rebuilt for the new regime. Independent of resets, two

further checks ensure the ML path is trusted only when it should be: a periodic trigger forces an occasional exploratory run as a routine sanity check even when the model reports confidence, and a per-target uncertainty threshold flags any prediction that falls outside the model’s demonstrated reliability. We next describe the implementation details underlying this logic, formalized in Algorithms 1–3. Readers more interested in the application of the proposed strategy to our pilot case study may proceed directly to Section 4.3.3.

Algorithm notation. Before presenting the algorithms, we define the key data structures. The elite pool $\mathcal{E}$ is a cumulative set containing the entire group of final solutions, denoted as the decision vectors U , from each preceding high-resolution optimization cycle. The training datasets $\{D_j\}_{j=1}^{N_v}$ store pairs of input and output features $(\phi_k, v_{\text{target},j})$ used to train the specific ML model for each terminal target component. The history of applied controls U_{hist} is a set of the optimal high-resolution control sequences that were used to advance the system state. In the context of the optimizer, $U^{(r)*}$ refers to the single best solution vector found at resolution r , while $\mathcal{P}^{(r)*}$ refers to the entire final set of candidate solutions obtained from the solver, hence $U^{(r)*} \in \mathcal{P}^{(r)*}$.

Algorithm parameters. Given the non-differentiable nature of the system dynamics, we employ a population-based evolutionary algorithm to solve the optimal control problem. The optimization process and the ML-driven logic are governed by several key parameters. For each resolution r , the solver uses a size of S_r candidate solutions over each iteration. The computational effort is controlled by a black-box function evaluation budget, which has a default value E_r^{def} and is temporarily increased to a larger initial value E_r^{init} to encourage broader exploration at the start of the optimal control or after a reset. A reset is triggered by comparing the latest price forecast p_k^{reset} against the forecast from the previous cycle $p_{\text{last}}, p_{\text{last}}$, allowing the controller to detect fundamental shifts in the market regime that invalidate previous optimization results. A cycle constitutes one complete iteration of the control process: solving the optimization problem for a planning horizon $H_{\hat{h}}$ and then applying the initial control actions to the system. The core of our method relies on a set of conditions that determine whether the ML model’s prediction should be accepted, or bypassed in favor of a full exploratory optimization. There are three such bypass conditions:

1. a mandatory warm-up period of duration t_{warmup} that begins at the time of the last reset t_{reset} ;
2. a periodic trigger activated every t_{periodic} time steps; and
3. a model uncertainty check based on a generalized uncertainty metric $\Psi(\cdot)$ (e.g., standard deviation, entropy, or ensemble variance). This condition triggers if any target component j exceeds its specific threshold: $\Psi_j(\phi_k) \geq \theta_{\text{unc},j}$. This formulation is designed to function as a confident baseline check, where the baseline itself is a tunable hyperparameter established either empirically from historical data or from domain knowledge of the system’s most frequent operational strategy.

Main control loop. The main loop (Algorithm 1) operates on a receding-horizon basis, repeatedly solving an optimization problem to determine the optimal control plan for the upcoming horizon. The central feature of our method lies in how this plan is determined at each cycle. The loop first invokes the ML-driven sub-routine (detailed below) to efficiently determine the terminal target state $\mathbf{v}_{\text{target}}$. Using this target as a terminal cost, it then solves the detailed, high-resolution optimization problem to find the final control actions. This loop also incorporates an adaptive reset mechanism to handle major changes in market conditions. When a significant shift in the price forecast triggers a reset, indicative of a seasonal or market regime change, the system deliberately discards the elite pool. While caching solutions is effective for recurring daily patterns, solutions optimized for a previous operational regime (e.g., winter prices) are likely suboptimal for the new one and could negatively bias the search. Therefore, the reset boosts exploration budgets to E_r^{init} and initiates a new warm-up period, forcing the ML model to adapt to the new environment. The elite pool $\mathcal{E}$ and the training datasets $\{D_j\}$ are cleared as well to avoid concept drift and bias towards prior seasons.

ML-driven terminal target determination. This sub-routine (Algorithm 2) is where the primary computational savings are realized. Its goal is to provide the terminal target vector $\mathbf{v}_{\text{target}}$ for the two-stage multi-resolution optimization problem (described in next section). A feature vector ϕ_k is constructed from the current forecasts and fed to the online-trained ML model $\{\mathcal{M}_j\}$ which yield predictions and a generalized uncertainty estimate via $\Psi(\cdot)$. A decision is then made between two paths:

Predict Path (Default): The algorithm bypasses the expensive exploratory optimization and simply sets $\mathbf{v}_{\text{target}} \leftarrow \hat{\mathbf{v}}$, where $\hat{\mathbf{v}}$ is the vector that contains the mean predictions of the ML models specific for each terminal state.

Compute Path (Exception): If any of the bypass conditions defined in the Algorithm Parameters are met, a full, exploratory optimization ($r = \hat{e}$) is executed. This is done to gather new data during warm-up and periodic triggers or to ensure safety when the model's uncertainty is high. If this path is taken, the resulting optimal target vector is used, and crucially, the new observation pairs $(\phi_k, \mathbf{v}_{\text{target},j})$ are added to their respective datasets D_j to continuously retrain and improve the ML models.

Two-stage optimization cascade. Once $\mathbf{v}_{\text{target}}$ is determined, this sub-routine (Algorithm 3) executes a two-stage optimization incorporating the $\mathbf{v}_{\text{target}}$ as the terminal cost. First, a low-resolution ($r = l$) optimization is performed. Second, the final, high-resolution ($r = h$) stage is seeded with an initial set blended from two sources: N_{elites} solutions sampled from the global elite pool set $\mathcal{E}$ and the N_{seed} top-performing solutions from the low-resolution stage. The low-resolution solutions are up-sampled to the finer timescale using a linear interpolation operator, denoted by $\mathcal{I}_{l \rightarrow h}$.

4.3.3. Pilot system implementation

This section provides the specific details of applying the general framework to the Pilot System involving a heat pump and a battery. In the context of our system, the generic target vector $\mathbf{v}_{\text{target}}$ from the algorithm contains the battery's State of Charge (SoC), $\text{SoC}_{\text{target}}$ and the median temperature of the thermal storage $T_{\text{median,target}}$. These values guide the trade-off between completely emptying the battery and the thermal storage at the end of the time period for immediate cost savings within the H_l and H_h horizons versus preserving some of its charge for potentially more valuable uses in the future, as determined by the long-horizon exploratory optimization. The `ExtractTargetComponent(x)` function is therefore implemented as a simple selection of the SoC and median thermal storage temperature components from the state vector $\mathbf{x}$.

System model and control spaces. The state vector $\mathbf{x}_k \in \mathbb{R}^{n_r}$ includes the battery's SoC $[\mathbf{x}_k]_{\text{SoC}}$, the thermal storage median temperature $[\mathbf{x}_k]_{T_{\text{median}}}$, and the instantaneous energy production output $[\mathbf{x}_k]_{q_{\text{prod}}}$. The control input vector for this system is defined as $\mathbf{u}_k = [u_k^{\text{pump}}, u_k^{\text{boi}}, u_k^{\text{bat}}]^T$. The pump commands for the heat pump and boiler, $[u_k^{\text{pump}}, u_k^{\text{boi}}]^T \in [0, u^{\text{nom}}]^2$, represent their respective mass flow rates measured upstream of the valve, and u^{nom} is the nominal mass flow rate. The battery command, $u_k^{\text{bat}} \in [-1, 1]$, is a normalized, dimensionless variable representing charge/discharge instructions.

Control projection rules. The unconstrained decision variables are mapped to physical commands through the following projection rules. For both pumps, a minimum flow rate is enforced to prevent inefficient operation at low part load ratio as

$$u_k^{\text{pump,proj}} = \begin{cases} 0, & \text{if } u_k^{\text{pump}} < \alpha_{\text{nom}} u^{\text{nom}} \\ u_k^{\text{pump}}, & \text{otherwise,} \end{cases} \quad (9)$$

where α_{nom} is the minimum part load operation.

The normalized battery command is projected to a physical power command $P_k^{\text{bat,proj}}$. This projection accounts for the available energy based on the current SoC, battery capacity E_{max} , discharging and charging efficiencies $\eta_{\text{discharge}}$ and η_{charge} , and the battery's nominal power rating $P_{\text{bat}}^{\text{nom}}$ as

$$P_k^{\text{bat,avail}} = \begin{cases} \frac{(1 - \text{SoC}_k) E_{\text{max}}}{\Delta t_r \eta_{\text{charge}}}, & \text{if } u_k^{\text{bat}} \geq 0 \quad (\text{charging}) \\ \frac{\text{SoC}_k E_{\text{max}} \eta_{\text{discharge}}}{\Delta t_r}, & \text{if } u_k^{\text{bat}} < 0 \quad (\text{discharging}). \end{cases} \quad (10)$$

This available power is then scaled by the normalized command and clipped by the battery's nominal power rating $P_{\text{bat}}^{\text{nom}}$ which defines the absolute upper and lower power limits as

$$P_k^{\text{bat,proj}} = \text{clip} \left(u_k^{\text{bat}} P_k^{\text{bat,avail}}, -P_{\text{bat}}^{\text{nom}}, P_{\text{bat}}^{\text{nom}} \right). \quad (11)$$

This ensures the commanded power respects both state-dependent energy limits and the absolute power limits.

Cost function. This section defines the specific components that constitute the stage cost ℓ and the terminal cost V_f in our problem formulation.

Stage Cost ℓ

The stage cost $\ell(x_k, u_k, w_k)$ represents the total cost incurred during each time step k . It is the sum of the direct energy cost and a penalty for daytime production shortfall:

1. The **energy cost** ℓ_{energy} is the incremental energy cost computed by the plant model based on known price $p(t_k)$ and weather $w(t_k)$ forecasts:

$$\ell_{\text{energy}}(x_k, u_k, t_k) = C_{\text{energy}}(x_k, u_k, p(t_k), w(t_k)) \quad (12)$$

2. The **production shortfall penalty** $\ell_{\text{shortfall}}$ penalizes any failure to meet the thermal production target $q_{\text{prod,target}}$ required to meet the plant load demand during the specific hours the plant is in operation. Rather than representing a physical operating cost, this term serves as a feasibility-enforcing penalty that penalizes any candidate trajectory during optimization in which the thermal production falls short of the required target, effectively ensuring that only feasible solutions are retained in the final result. It is

$$\begin{aligned} \ell_{\text{shortfall}}(t_k, [\mathbf{x}_k]_{q_{\text{prod}}}) &= \begin{cases} \gamma_{\text{day}} \max(0, q_{\text{prod,target}} - [\mathbf{x}_k]_{q_{\text{prod}}}), & \text{if } t_k \in [08:00, 20:00], \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \quad (13)$$

Algorithm 1 ML-Accelerated Optimal Control (Main Loop)

Require: ML models $\{\mathcal{M}_j\}_{j=1}^{N_v}$, total time T_{end} , resolution parameters $\{\Delta t_r, H_r, S_r\}$, budgets $\{E_r^{\text{init}}, E_r^{\text{def}}\}$, uncertainty thresholds $\{\theta_{\text{unc},j}\}_{j=1}^{N_v}$, $t_{\text{warmup}}, t_{\text{periodic}}$.

- 1: Let $K_{\text{end}} = T_{\text{end}} / \Delta t_{\hat{h}}$
- 2: **for** $k \leftarrow 0, H_{\hat{h}}, 2H_{\hat{h}}, \dots, K_{\text{end}}$ **do**
- 3: Obtain current state x_k at time $t_k = k \cdot \Delta t_{\hat{h}}$
- 4: Acquire price forecast for reset check: $\mathbf{p}_k^{\text{reset}}$
- 5: **if** $k = 0$ **or** $\|\mathbf{p}_k^{\text{reset}} - \mathbf{p}_{\text{last}}\|_{\infty} > 10^{-8}$ **then** ▷ Detect change in price profile
- 6: *//– Reset logic: Clear elites, clear datasets, boost budgets, and start new warm-up –//*
- 7: $\mathcal{E} \leftarrow \emptyset$;
- 8: $\{D_j\} \leftarrow \emptyset$; ▷ Clear all datasets $E_r \leftarrow E_r^{\text{init}}$ for all r
- 9: $\mathbf{p}_{\text{last}} \leftarrow \mathbf{p}_k^{\text{reset}}$
- 10: $t_{\text{reset}} \leftarrow t_k$
- 11: **end if**
- 12: Acquire forecasts and form ML feature vector ϕ_k
- 13: $(\mathbf{v}_{\text{target}}, \{D_j\}, \{\mathcal{M}_j\}) \leftarrow \text{DetermineTerminalTarget}(x_k, \phi_k, t_k, t_{\text{reset}}, \{\mathcal{M}_j\}, \{D_j\}, \{\theta_{\text{unc},j}\}_{j=1}^{N_v})$ ▷ Pass/Return sets
- 14: $(U^{(\hat{h})}, \mathcal{P}^{(\hat{h})}) \leftarrow \text{FindOptimalControl}(x_k, \mathbf{v}_{\text{target}}, \mathcal{E})$
- 15: *//– Apply control and update algorithm state for next cycle –//*
- 16: $\mathbf{U}_{\text{hist}} \leftarrow \text{Concatenate}(\mathbf{U}_{\text{hist}}, U^{(\hat{h})})$
- 17: $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{P}^{(\hat{h})}$
- 18: **if** $E_{\hat{h}} = E_{\hat{h}}^{\text{init}}$ **then** ▷ If this was a boosted cycle...
- 19: $E_r \leftarrow E_r^{\text{def}}$ for all r ▷ ...reset budgets to default values.
- 20: **end if**
- 21: **end for**

Algorithm 2 Sub-routine: Determine Terminal Targets

- 1: **function** DETERMINE_TERMINAL_TARGETS($x_k, \phi_k, t_k, t_{\text{reset}}, \{\mathcal{M}_j\}, \{D_j\}, \{\theta_{\text{unc},j}\}_{j=1}^{N_v}$) ▷ Modified: Args
- 2: *//– Decide whether to run expensive optimization or trust the ML models –//*
- 3: $\text{is_warmup} \leftarrow (t_k - t_{\text{reset}}) < t_{\text{warmup}}$
- 4: $\text{is_periodic_trigger} \leftarrow t_k \pmod{t_{\text{periodic}}} = 0$
- 5: $\text{is_uncertain} \leftarrow \text{false}$
- 6: Initialize prediction vector $\hat{\mathbf{v}} \in \mathbb{R}^{N_v}$
- 7: **for** $j \leftarrow 1$ **to** N_v **do**
- 8: $\hat{\mathbf{v}}[j] \leftarrow \text{Predict}(\mathcal{M}_j, \phi_k)$
- 9: $u_{\text{val}} \leftarrow \Psi_j(\mathcal{M}_j, \phi_k)$ ▷ Calculate generic uncertainty
- 10: **if** $u_{\text{val}} \geq \theta_{\text{unc},j}$ **then**
- 11: $\text{is_uncertain} \leftarrow \text{true}$
- 12: **end if**
- 13: **end for**
- 14: **if** is_warmup **or** $\text{is_periodic_trigger}$ **or** is_uncertain **then**
- 15: *//– Compute Path: Run full exploratory optimization –//*
- 16: $[U^{(\hat{e})}, X^{(\hat{e})}] \leftarrow \text{Solve}(\arg \min_{U^{(\hat{e})}} \sum_{i=0}^{H_{\hat{e}}-1} \ell(x_i, u_i, t_i) \text{ s.t. } x_0 = x_k)$
- 17: $T_{\text{target_horizon}} \leftarrow H_{\hat{e}} \cdot \Delta t_{\hat{h}}$
- 18: $x_{\text{target}} \leftarrow \text{State from trajectory } X^{(\hat{e})} \text{ at time } t_k + T_{\text{target_horizon}}$
- 19: $\mathbf{v}_{\text{target}} \leftarrow \text{ExtractTargetComponents}(x_{\text{target}})$
- 20: *//– Online learning: Update datasets and retrain the models –//*
- 21: **for** $j \leftarrow 1$ **to** N_v **do** ▷ Loop to update specific datasets
- 22: $D_j \leftarrow D_j \cup \{(\phi_k, \mathbf{v}_{\text{target}}[j])\}$
- 23: **end for**
- 24: $\{\mathcal{M}_j\} \leftarrow \text{RetrainModels}(\{D_j\})$
- 25: **else**
- 26: *//– Predict Path: Use the confident ML prediction –//*
- 27: $\mathbf{v}_{\text{target}} \leftarrow \hat{\mathbf{v}}$
- 28: **end if**
- 29: **return** $(\mathbf{v}_{\text{target}}, \{D_j\}, \{\mathcal{M}_j\})$ ▷ Return sets
- 30: **end function**

where $\gamma_{\text{day}} > 0$ is a penalty term. The total stage cost included in the main objective function's summation is thus $\ell(x_k, u_k, w_k) = \ell_{\text{energy}}(x_k, u_k, w_k) + \ell_{\text{shortfall}}(x_k, u_k, w_k)$.

Terminal Cost V_f

The terminal cost $V_f(x_H)$ is a soft constraint applied only once to the final state x_H . Its purpose is to guide the system towards desirable terminal states. It measures the deviation of the final SoC

Algorithm 3 Sub-routine: Two-Stage Optimal Control Cascade

```

1: function FINDOPTIMALCONTROL( $x_k, \mathbf{v}_{\text{target}}, \mathcal{E}$ )
2:   //– Stage 1: Low-Resolution Optimization ( $r = \hat{l}$ ) –//
3:   ( $U^{(\hat{l})}, \mathbf{p}^{(\hat{l})}$ )  $\leftarrow$  Solve for the solution to:
        $\arg \min_{U^{(\hat{l})}} \left[ \sum_{i=0}^{H_{\hat{l}}-1} \ell(x_i, u_i, t_i) + V_f^{(\hat{l})}(x_{H_{\hat{l}}}, \mathbf{v}_{\text{target}}) \right]$ 
4:   //– Stage 2, Step A: Blend initial set for high-resolution –//
5:    $N_{\text{elites}} \leftarrow \min(|\mathcal{E}|, \lfloor S_{\hat{l}}/2 \rfloor)$ 
6:    $\mathcal{E}_{\text{sub}} \leftarrow \text{UniformRandomSample}(\mathcal{E}, N_{\text{elites}})$ 
7:    $\mathcal{P}_{\text{interp}} \leftarrow \{I_{\hat{l} \rightarrow \hat{h}}(U) \mid U \in \mathcal{P}^{(\hat{l})}\}$   $\triangleright I$  is linear interpolation
8:   Sort  $\mathcal{P}_{\text{interp}}$  by cost
9:    $N_{\text{seed}} \leftarrow S_{\hat{h}} - N_{\text{elites}}$ 
10:   $\mathcal{P}_{\text{seed, sub}} \leftarrow$  first  $N_{\text{seed}}$  solutions from  $\mathcal{P}_{\text{interp}}$ 
11:   $\mathcal{P}_0^{(\hat{h})} \leftarrow \mathcal{E}_{\text{sub}} \cup \mathcal{P}_{\text{seed, sub}}$   $\triangleright$  Final blended initial set
12:  //– Stage 2, Step B: High-Resolution Optimization ( $r = \hat{h}$ ) –//
13:  ( $U^{(\hat{h})}, \mathbf{p}^{(\hat{h})}$ )  $\leftarrow$  Solve for the solution to:
        $\arg \min_{U^{(\hat{h})}} \left[ \sum_{i=0}^{H_{\hat{h}}-1} \ell(x_i, u_i, t_i) + V_f^{(\hat{h})}(x_{H_{\hat{h}}}, \mathbf{v}_{\text{target}}) \right]$ 
       using initial set  $\mathcal{P}_0^{(\hat{h})}$ 
14:  return ( $U^{(\hat{h})}, \mathbf{p}^{(\hat{h})}$ )
15: end function

```

and thermal storage temperature from their respective target values, $\text{SoC}_{\text{target}}$ and $T_{\text{median, target}}$. These targets are the key outputs from the initial exploratory optimization and provide consistent long-term goals for subsequent refinement stages. The terminal cost is formulated as the weighted sum of these deviations:

$$V_f(x_H) = \begin{cases} \gamma_{\text{soc}} |x_{H_r} |_{\text{SoC}} - \text{SoC}_{\text{target}}| + \gamma_T |x_{H_r} |_{T_{\text{median}}} - T_{\text{median, target}}|, & \text{if } r \in \{\hat{l}, \hat{h}\}, \\ 0, & \text{if } r = \hat{e}. \end{cases} \quad (14)$$

With the production shortfall measured in megawatts (MW), the γ_{day} factor applies a cost of 1000 for every 1 MW of unmet daytime demand. Note that the exploratory stage itself ($r = \hat{e}$) does not use the terminal cost term ($V_f = 0$), as its objective is to determine the optimal $\text{SoC}_{\text{target}}$ and $T_{\text{median, target}}$. To address the disparity in magnitude between the tank temperature T_{median} and the SoC , the corresponding penalty weights were calibrated via numerical experimentation to ensure a balanced contribution to the cost function.

Multi-resolution scheme. The optimization proceeds sequentially through three stages:

- **Exploratory Stage** ($r = \hat{e}$): A 48-hour horizon with a 2-hour time step ($H_{\hat{e}} = 24$). This stage runs without the terminal SoC and T_{median} penalties ($\gamma_{\text{soc}} = 0$ and $\gamma_T = 0$). This extended horizon is critical for avoiding myopic decisions, as it allows the optimizer to properly value the “carryover” of stored energy from day 1 into day 2. Its purpose is to determine the optimal $\text{SoC}_{\text{target}}$ and $T_{\text{median, target}}$ values at the 24-hour mark, which is then passed to the subsequent stages. While longer horizons may be considered, forecast quality degrades beyond 48 h, limiting the practical benefit of extending the exploratory window further; a longer horizon would require either a higher-dimensional optimization problem at the same discretization or a coarser one that sacrifices solution quality. This is the stage that the ML model aims to replace.
- **Low-Resolution Stage** ($r = \hat{l}$): A 24-hour horizon with a 1-hour time step ($H_{\hat{l}} = 24$). It uses the $\text{SoC}_{\text{target}}$ and $T_{\text{median, target}}$ values from the exploratory stage to compute its terminal penalty. This 24-hour duration represents a crucial trade-off: it is long enough to encompass a full diurnal cycle of solar PV generation

and plant load, it provides complete daily solutions suitable for populating the elite pool for recurring daily patterns, and it keeps the problem computationally tractable. A 24-hour horizon also aligns with the period over which price signals are typically available in wholesale energy markets under day-ahead pricing (Alam, 2019).

- **High-Resolution Stage** ($r = \hat{h}$): A 24-hour horizon with a 30-minute time step ($H_{\hat{h}} = 48$). It also uses the $\text{SoC}_{\text{target}}$ and $T_{\text{median, target}}$ values, and its solution provides the actual controls to be applied to the system. The 24-hour horizon is maintained for the reasons stated above, while the fine-grained time step is necessary to better align the rate of change of control action with the major system dynamics.

Algorithm hyperparameters.

- **Black-Box Optimizer:** The proposed framework is solver-agnostic. We use the Adaptive Differential Evolution algorithm (JADE) (Zhang and Sanderson, 2009). Key hyperparameters include a mutation greediness $p = 0.05$, parameter adaptation rate $c = 0.1$, and the “DE/current-to-pbest” strategy. Population sizes were set to $S_{\hat{e}} = 48$, $S_{\hat{l}} = 96$, and $S_{\hat{h}} = 96$. We utilize our own implementation of JADE in PyTorch 2.3.0 and Python 3.11 (Paszke et al., 2019). The JADE solver is openly available as a part of the whole framework.
- **Budgets:** Default optimization budgets were 4000 evaluations for the exploratory stage, and 5000 for low-resolution and high-resolution stages. During a reset, the high-resolution budget was boosted to 20,000.
- **Triggers:** For the SoC terminal target, the low uncertainty threshold of $\theta_{\text{unc}} = 0.01$ was set, as it establishes a highly conservative trigger. This combination ensures that the computationally inexpensive ML path is only used when the ML model’s upper confidence bound of the target SoC is itself below 1%, effectively green-lighting only the most confident predictions of the common “discharge” strategy. For the thermal storage median temperature, the uncertainty threshold θ_{unc} was set to be 71°C. This threshold was selected to force the system to run the full exploratory stage whenever the ML model gives any indication that there may be some benefit to the tank being partially charged at the end of the operating horizon; exploration is only avoided when the ML model is completely confident that the tank should be emptied (i.e., the temperature is at 70°C, the allowable lower bound of operation). In essence, the thresholds act as a decision point: the algorithm assesses if it is confident enough to commit to the greedy discharge strategy based on current forecasts, or if the level of uncertainty warrants invoking the full exploratory optimizer to find a potentially more better solution. Finally, we set a warmup period of $t_{\text{warmup}} = 240$ hours (10 days), and a periodic trigger of $t_{\text{periodic}} = 120$ hours (5 days).
- **Feature Vectors:** The price reset vector $\mathbf{p}_k^{\text{reset}}$ uses a 24-hour forecast at 30-minute intervals (48 dimensions). The ML input feature vector ϕ_k uses a 48-hour weather forecasts at 2-hour intervals (24 dimensions total), while the output features are the $\text{SoC}_{\text{target}}$ and the T_{median} .

4.3.4. ML models

We evaluated two distinct supervised learning ML architectures to generate the terminal target predictions $\hat{\mathbf{v}}$ and the associated uncertainty estimates $\Psi(\cdot)$ required by our framework. Both implementations utilize scikit-learn 1.6.1 (Pedregosa et al., 2011) and were executed with fixed random seeds to ensure a controlled experimental environment.

Random Forest (RF): The first approach employs RF regressors (Breiman, 2001). We train one independent model with 200 decision trees for each target component j . This architecture is efficient as it

allows for simultaneous estimation of the value and the uncertainty from a single model. For a given input ϕ_k , the mean prediction $\hat{\nu}_j$ is computed as the average of the predictions from all individual trees in the specific model for target j . The uncertainty $\sigma_{k,j}$ is derived from the standard deviation of these individual tree predictions. The upper confidence bound used to trigger the uncertainty bypass condition is calculated assuming a Gaussian distribution of errors as $\text{Upper Bound} = \hat{\nu}_j + \kappa \cdot \sigma_{k,j}$. We evaluated this approach using two distinct exploration settings: $\kappa = 1$ and $\kappa = 2$, corresponding to theoretical confidence intervals of approximately 68% and 95%, respectively, under the normality assumption.

Gradient Boosting with Quantile Regression (GB): The second approach utilizes GB Regressors. Unlike the RF approach, this method requires training two distinct models for each target component j . The first model is trained to predict the mean value $\hat{\nu}_j$ using the standard squared error loss. The second model is specifically trained to estimate the upper confidence bound directly using quantile regression. This second model minimizes the pinball loss function L_α . Defining z as the true target value and $\hat{z}$ as the predicted value, the loss for a target quantile $\alpha = 0.95$ is:

$$L_\alpha(z, \hat{z}) = \max(\alpha(z - \hat{z}), (\alpha - 1)(z - \hat{z})) \quad (15)$$

Because α is close to 1, this loss function imposes a heavy penalty on underestimation ($z > \hat{z}$) and a small penalty on overestimation. Consequently, the model learns to output a prediction $\hat{z}$ that exceeds the true value z in approximately 95% of cases. Physically, this intentional overestimation of terminal targets creates a conservative buffer. While this forces the optimizer to perform a more exhaustive and computationally expensive search to meet stricter targets, it safeguards against the sub-optimality that would arise from underestimating the true terminal requirements.

While the RF approach is computationally lighter (one model per target), the calculation of the upper bound using $\kappa \cdot \sigma$ implicitly assumes that the posterior distribution of the target uncertainty is Gaussian. The GB approach with quantile loss is theoretically superior for this application because it is distribution-free; it does not rely on assumptions of normality. This allows the framework to more accurately capture the asymmetric or heavy-tailed risk profiles often seen in energy systems, albeit at the cost of managing and training twice as many models.

5. Results

In this section, we present the results for the pilot case study. We start with a brief discussion of the results from the architectural optimization process (Section 5.1). We then present in detail the estimated limit of performance analysis, presenting results that highlight the benefits of both the multi-fidelity and ML contributions (Sections 5.2–5.4). In Section 5.5, we present a brief comparison of the actual control decisions produced by the different strategies explored in this work.

For our analysis, we consider electricity prices in California obtained through the California Energy Commission's Market Informed Demand Automation Server (Prakash and Paul, 2022). Details on the electricity prices, along with the weather data and equipment design parameters used in our case study, are presented in Appendix C. Equipment cost data for turn-key installed components are presented in Table A.1.

5.1. Architectural optimization results

Fig. 5 shows the Pareto front obtained for the architectural design optimization problem. Limiting the amount of energy that can be imported from outside the facility incurs a significant cost penalty; the TAC of the designs range from to \$0.235–0.704 M/yr a three-fold cost increase.

The Pareto front exhibits three distinct regimes (shaded areas), with the gradient of the Pareto front decreasing in each regime (left to right).

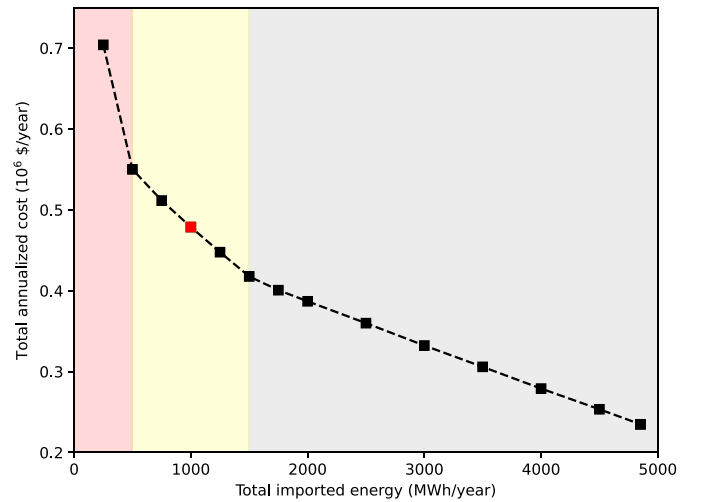


Fig. 5. Pareto front for multi-objective problem. The squares represent individual designs on the solution front. The shaded areas highlight the three distinct regimes observed. The red square represents the design to be considered in the limit of performance analysis.

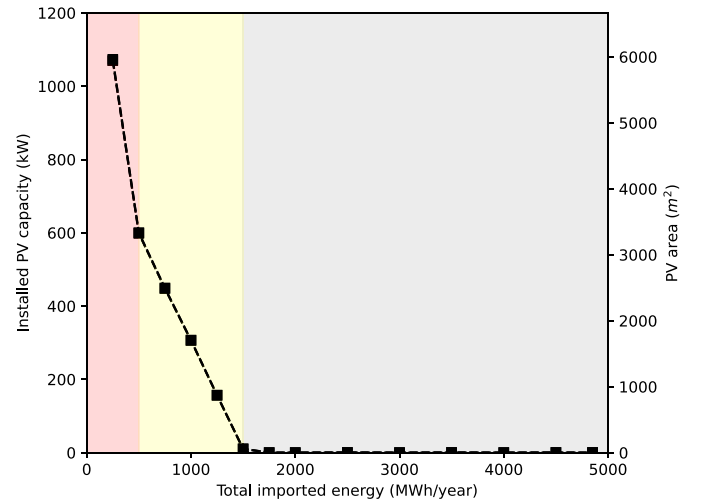


Fig. 6. Installed PV capacities for designs on Pareto front.

To better understand the shape of the Pareto front, we present a plot of the installed PV capacities in Fig. 6, and plots of the generation and storage capacities of the designs are presented in Appendix D. The shape of the trade-off curve is clearly driven by photovoltaic installation. The rightmost regime (gray) corresponds to the regime with no PV installation; all energy demands from the plant are met by either purchasing electricity from the grid or purchasing gas. In this regime ($TE > 1500 \text{ MWh/yr}$), the Pareto front is linear and shows the slowest rate of cost increase ($\approx \$53$ per MWh of imported energy reduction). In the far-right design ($TE = 4852 \text{ MWh/yr}$; $TAC = \$0.235 \text{ M/yr}$) which represents the design when there is essentially no constraint on energy imports, the TE is just above the annual thermal demand of the plant (4380 MWh/yr). The design uses only natural gas for heat generation; a 1MW gas boiler is installed and operated continuously to meet the load demands (no storage required). Designs in the gray regime trade off between the installed heat pump and gas boiler capacities while continuously increasing the installed tank and battery storage capacities; the leftmost design in the gray regime drops the gas boiler to near zero. Beyond this point, PVs are installed to reduce the imported

Table 2
Equipment sizes for design selected for the limit of performance analysis^a.

Component	Capacity
PV	306.2 kW
Heat pump	862.2 kW
Battery	1124.8 kWh
Hot water tank	1911 kWh

^a The optimal design includes a 6 kW gas boiler, but since this is less than 1% of the heat pump capacity, therefore we will in the limit of performance analysis and in the control verification below assume that the boiler contribution is negligible, i.e., $[u_k^{\text{boi}}]^T = 0$ for all k .

energy. The maximums of the installed storage capacities for both the battery (1.2 MWh) and hot water tank (2.0 MWh) occur in this regime.

For designs in the yellow regime ($500 \text{ MWh/yr} < \text{TE} < 1500 \text{ MWh/yr}$), PV installation is required. Designs in the yellow shaded region are solely dependent on heat pumps for heat generation alongside battery and hot water storage; gas boiler capacities are less than 1% of the plant load ($< 10 \text{ kW}$). Designs in this regime come at a cost penalty, costing roughly twice as much as in the gray regime ($\approx \$120 - \$130$ per MWh of imported energy reduction).

For designs in the red regime ($\text{TE} < 500 \text{ MWh/yr}$), every additional MWh of imported energy reduction comes at a steep cost. For the left-most design ($\text{TE} = 250 \text{ MWh/yr}$), the optimal solution is to install PV and heat pump capacities of 1.07 MW and 1.00 MW, respectively, roughly the level of the hourly load.

The results of the architectural optimization provides us with information on the optimal system configurations and equipment sizes required to meet the load of the pilot plant. Given the system architecture, the next step is design verification, where the goal is to design the best possible control policy/strategy for operating a fixed energy system architecture at a minimum cost while still satisfying all requirements. In selecting a design for the limit-of-performance analysis, we selected a design that includes some PV. The intrinsic variability of PV generation provides a valuable source of disturbance, enabling a more rigorous evaluation and stress-testing of our controller design methodologies. Based on this requirement, we select the design at the midpoint of the yellow regime ($\text{TE} = 1000 \text{ MWh/yr}$; $\text{TAC} = \$0.479 \text{ M/yr}$). Table 2 summarizes the equipment sizes for the design.

For our application, the modelica models represent the heating and electrical system shown in Fig. 2. We used the optimization results from IDAES for the sizes of the heat pump, the PV, the battery and the hot water tank. As IDAES sized the gas boiler to less than 1% of the size of the heat pump, we added a control option to permanently disable the gas boiler, as this equipment will not be part of the recommended design.

5.2. Limit of performance analysis

5.2.1. Yearly operating costs

In this section, we present the limit of performance analysis that uses our ML-Accelerated optimal control approach, which will be compared against a Rule-based controller. This comparison will show the performance gap between these two controllers. The primary metric for evaluation is the total cost of operating the system at the end of one year. To provide a comprehensive assessment, we deploy three distinct variants of the ML-Accelerated framework, each distinguished by its underlying predictive model and uncertainty quantification mechanism. We denote the RF-based implementations as $\text{ML}_{\text{RF1}}\text{-MR}_{\text{ws}}$ and $\text{ML}_{\text{RF2}}\text{-MR}_{\text{ws}}$, utilizing uncertainty weights of $\kappa = 1$ and $\kappa = 2$, respectively. Additionally, we evaluate $\text{ML}_{\text{GB}}\text{-MR}_{\text{ws}}$, which employs the GB model calibrated to the 0.95 quantile loss.

Additionally, we assess the ML-Accelerated optimal control approaches against a multi-resolution strategy (denoted MR_{ws}) that incorporates elite pool warm starting but does not include ML acceleration. This will allow us to evaluate the computational savings and quantify any potential loss of solution quality due the ML model integration. We also compare it to a baseline single high-resolution approach (denoted as HR) that lacks elite pool warm starting. In the HR approach, we use the same number of evaluations as all the other approaches, however, all are utilized to evaluate only the high-resolution objective function. Both the multi-resolution and high-resolution methods encompass the full exploratory stage, where we solve the exploratory optimization problem to obtain the terminal SoC. This analysis highlights how the ML-Accelerated approaches can achieve the same or even better performance while requiring a reduced exploratory evaluation budget over the entire year. For the three methods, all numerical experiments were repeated 4 times and conducted on a single AMD EPYC 9534 processor with 128 cores, utilizing parallel objective function evaluations. The comprehensive yearly optimal control run, considering the full evaluation budget of both high-resolution and multi-resolution approaches, required approximately 2.5 days to complete.

Drawing from the insights of the architectural optimization, we define the specific parameters for the limit of performance analysis. The boiler was determined to be insignificant and is therefore excluded from the problem, simplifying the control input vector to two dimensions, i.e., the heat pump mass flow rate and battery command. Consequently, the decision vectors for the exploratory, low-resolution, and high-resolution stages are defined as $U^{(\varepsilon)} \in \mathbb{R}^{48}$, $U^{(l)} \in \mathbb{R}^{48}$, and $U^{(h)} \in \mathbb{R}^{96}$, respectively.

The key system parameters are set as follows: the battery has a maximum capacity of $E_{\text{max}} = 1124.8 \text{ kWh}$ and a nominal power rating of $P_{\text{bat}}^{\text{nom}} = 400 \text{ kW}$, while the heat pump has a maximum mass flow rate of $u^{\text{nom}} = 12 \text{ kg/s}$ (computed from water requirement to meet 1MW thermal load). For the control projection rules, a threshold of $\alpha_{\text{nom}} = 0.2$ is applied to the pump command. The weighting factors for the terminal cost penalty were determined through numerical tuning to be $\gamma_{\text{day}} = 1000$, $\gamma_{\text{soc}} = 5000$ and $\gamma_T = 5$. All optimizations utilize the same price $p(t_k)$ and weather $w(t_k)$ forecast data as the architectural optimization study, with further details provided in Appendix C.

Fig. 7 illustrates the mean total operating cost of the system over the course of a full year for all evaluated methods (note that the IDAES and Rule-based approaches are only a single run). Firstly, the results indicate that the HR method, specifically the out-of-the-box JADE solver, exhibits the poorest performance, falling significantly short of the Rule-based approach, with a mean total cost of $\$96,225$ at the end of the year. This underperformance of the HR method underscores the need for developing more sophisticated and complex methodologies. Furthermore, the standard multi-resolution method MR_{ws} and the three ML-Accelerated variants demonstrate comparable performance, achieving mean total costs ranging between $\$73,054$ and $\$74,157$. All multi-resolution optimization-based approaches consistently outperform the Rule-based baseline, which incurred a total cost of $\$80,194$. Notably, while $\text{ML}_{\text{RF2}}\text{-MR}_{\text{ws}}$ achieved the lowest average cost, the absolute best individual result across all experimental runs was achieved by the $\text{ML}_{\text{GB}}\text{-MR}_{\text{ws}}$ method, with a minimum total cost of $\$71,778$. This yields a savings of $\$8416$, representing a 10.5% improvement over the Rule-based approach. The best performing ML run produces significant controller performance improvement, cutting the performance gap between the Rule-based controller and the theoretical lower bound established by the IDAES approach ($\$60,284$) by 42%.

The convergence of all stochastic methods remains relatively stable over 4 runs. As shown in Table 3, the standard deviations of the total annual operating cost from all methods is under 2%, with MR_{ws} and HR showing the highest and lowest percentage deviations, respectively. The GB approach $\text{ML}_{\text{GB}}\text{-MR}_{\text{ws}}$, which produces the best single run of the ML-based approaches, also shows slightly higher variation than the

Table 3

Total yearly cost statistics for different approaches over 4 runs. The standard deviation percentages are computed relative to the mean values.

Approach	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
HR	97,482	97,498	97,019	97,913	317	0.33
MR _{ws}	74,157	73,845	72,723	76,216	1275	1.72
ML _{RF1} -MR _{ws}	73,807	74,024	72,660	74,520	695	0.95
ML _{RF2} -MR _{ws}	73,054	73,250	72,039	73,677	659	0.91
ML _{GB} -MR _{ws}	73,188	73,109	71,778	74,758	1075	1.47
Rule-based ^a	80,194	–	–	–	–	–
IDAES ^a	60,284	–	–	–	–	–

^a Single run (no statistics available).

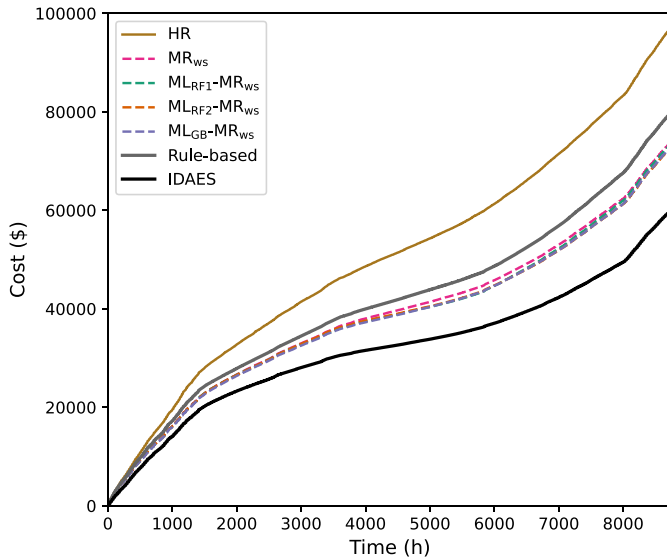


Fig. 7. Mean total operating cost of the energy system over one year for all used methods. A lower operating cost at any point in the year indicates better performance.

other two AI-enabled approaches, though this remains comparable to the baseline multi-resolution strategy.

The results above indicate that all multi-resolution approaches outperform direct high-resolution optimization, regardless of whether the ML component is incorporated; importantly, the inclusion of the ML models do not degrade solution quality and appear to enhance performance stability across repeated runs. However, where the ML-Accelerated approaches really excel over the baseline multi-resolution method MR_{ws} is their ability to reduce the total number of evaluations during the exploratory search. Fig. 8(a) illustrates the number of exploratory evaluations for each run across all variants. All three ML-driven variants demonstrate a significant advantage over the MR_{ws} approach, which requires a fixed total of 1,460,000 exploratory evaluations. However, a comparison of the variants reveals that the GB approach ML_{GB}-MR_{ws} is the most computationally efficient. As shown in Table 4, the GB method achieves a mean total of 962,000 evaluations, representing a 34% reduction in function evaluation calls without any noticeable loss in solution quality. This is also notably lower than the RF variants, which both average over 1.1 million evaluations. This efficiency stems from the GB model's specific uncertainty quantification; it triggers the expensive high upper confidence bound fallback condition significantly less often (averaging 498,000 triggers) compared to the RF models (averaging over 640,000 triggers). This suggests that the distribution-free quantile regression provides a more precise uncertainty bound than the Gaussian assumption used in the RF approaches, allowing the system to trust the ML predictions more frequently. Moreover, Comparing the two Random Forest variants

reveals that the uncertainty scaling parameter κ has a negligible impact on computational effort. The mean difference between ML_{RF1}-MR_{ws} ($\kappa = 1$) and ML_{RF2}-MR_{ws} ($\kappa = 2$) is approximately 1%, indicating that the uncertainty estimates rarely fall in the marginal zone between one and two standard deviations. Thus, tuning κ yields minimal efficiency gains compared to switching to the GB architecture, which significantly alters the evaluation frequency. Despite this, both RF-based approaches still manage to reduce the required number of evaluations by over 24%, representing significant computational savings.

The standard deviation observed within each method is driven entirely by the stochastic nature of the exploration triggers, as the remaining evaluations are fixed overhead. This variability arises from the JADE solver used in the exploratory optimization, which generates slightly different optimal trajectories for the training targets in each run. These variations lead to evolving datasets and differing uncertainty estimates σ_k across runs. Crucially, Fig. 8(b) compares the yearly operating costs of these ML runs against the MR_{ws} baseline. This comparison confirms that the massive computational reductions achieved by the ML_{GB}-MR_{ws} variant do not compromise system performance. Despite performing significantly fewer evaluations than the baseline, the ML-accelerated strategy maintains comparable operating costs, validating it as the preferred choice for solving the limit of performance problem.

Our proposed solution ML_{GB}-MR_{ws} uses several parameters such as objective function penalty factors (γ_{day} , γ_{soc} , γ_T), number of optimizer evaluations (E_f , E_h , E_e), the ML model's uncertainty thresholds ($\theta_{unc,Soc}$, $\theta_{unc,T_{median}}$), and the duration of the warm-up periods (t_{warmup}). To understand the impact of each of these on the solution quality and computational performance, we conduct extensive parametric sensitivity analyses on these factors. The results indicate that our algorithm shows sufficient robustness in cost performance to these parameters across a range of values; the details and findings of our parameter sensitivity studies are presented in Appendix F. The sensitivity analysis also establishes that the principal impact of the ML model parameters is on the frequency of exploration and High-UCB triggers; the ML model configuration and triggering logic has little impact on the cost, as expected. Furthermore, we provide additional analysis of the computational savings of the approach compared to the MR_{ws} baseline in a controlled environment in Appendix G.

5.2.2. In-season performance analysis

To quantify how controller performance varies with changes in price and weather patterns, we analyze in-season performance by decomposing the full-year simulation into four seasonal blocks. Since each season has distinct price and cost profiles, and the elites are discarded at the start of each season, each seasonal optimization problem represents a distinct use case for our algorithm. Thus, a comparison of in-season performance represents an assessment of the generalizability and robustness of our approach to variabilities in price and weather signals.

For each season, Figs. 9(a), 9(b), 9(c), and 9(d) show the mean within-season operating cost over time across the 4 repeated runs (with the exploratory optimization evaluation count included in the legend).

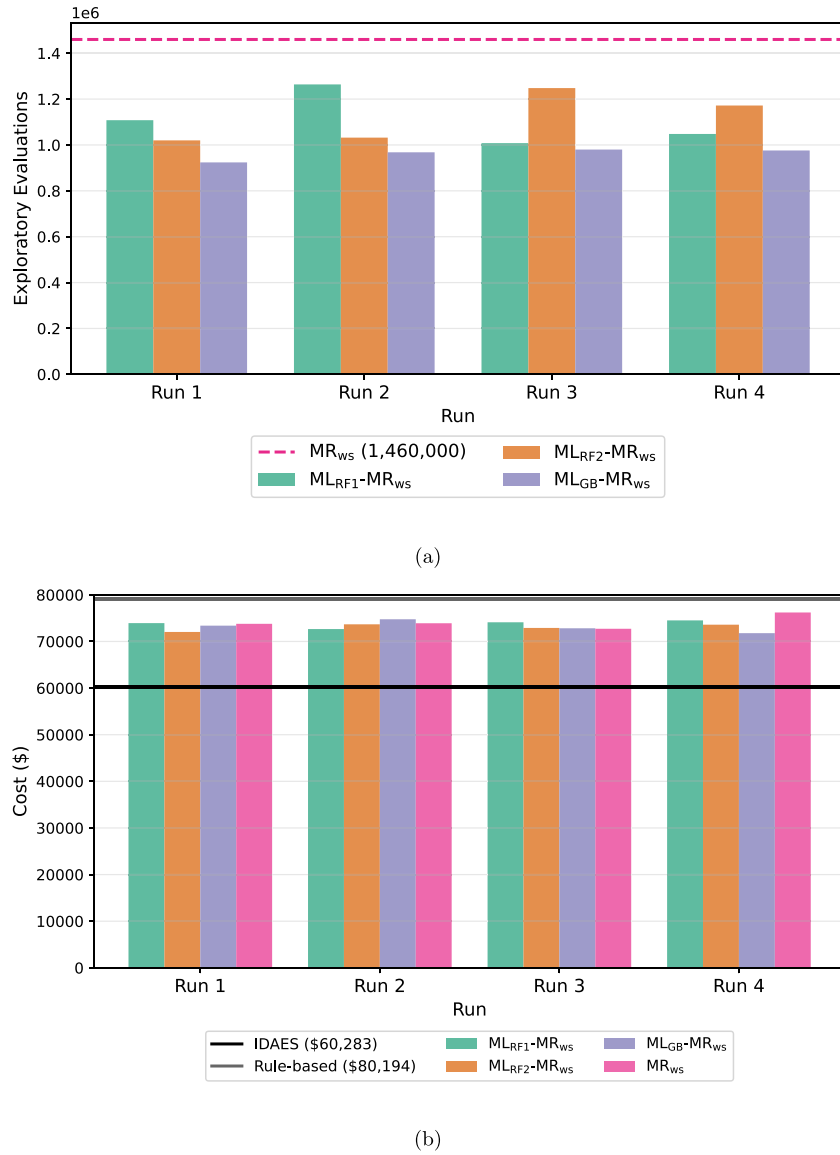


Fig. 8. A detailed analysis of each of the four runs of the ML-Accelerated variants is presented, where (a) displays the total number of exploratory evaluations (lower is better), and (b) shows a comparison of the total yearly operating costs (\$) of each run against the MR_{ws} method and the best-performing IDAES approach (lower is better).

Table 4

Statistics of exploratory evaluations over four runs for each ML-Accelerated variant. The Total denotes the total exploratory evaluations, while the High UCB Only column denotes the exploratory evaluations triggered by the ML model uncertainty. The standard deviation is derived from the high UCB exploratory runs, while the remaining evaluations are fixed across all runs.

Statistic	ML _{RF1} -MR _{ws}		ML _{RF2} -MR _{ws}		ML _{GB} -MR _{ws}	
	Total	High UCB	Total	High UCB	Total	High UCB
Mean	1107,000	643,000	1118,000	654,000	962,000	498,000
Std	97,381	97,381	95,937	95,937	22,361	22,361
Min	1008,000	544,000	1020,000	556,000	924,000	460,000
Max	1264,000	800,000	1248,000	784,000	980,000	516,000

Correspondingly, Table 5 summarizes the season-level operating costs and mean exploratory evaluation counts.

The GB-based ML variant remains competitive under these shifting operating conditions. The GB-based ML variant maintains a consistently good performance in every season, actually resulting in better operating costs than the MR_{ws} in three of the four seasons, and remaining within 1% of the MR_{ws} result at its worst. This performance is maintained while reducing the computational burden by as much

as 71%. The largest computation reductions occur in Seasons 1 and 3, where the GB variant avoids a substantial fraction of costly exploratory high-UCB fallbacks, yielding large exploration savings without sacrificing season-level cost performance. These results suggest that the computation-performance benefit of the GB-based ML acceleration is not tied to a single operating pattern, but instead demonstrates robustness across multiple seasonal forcing scenarios, supporting its practical applicability to similar types of problems.

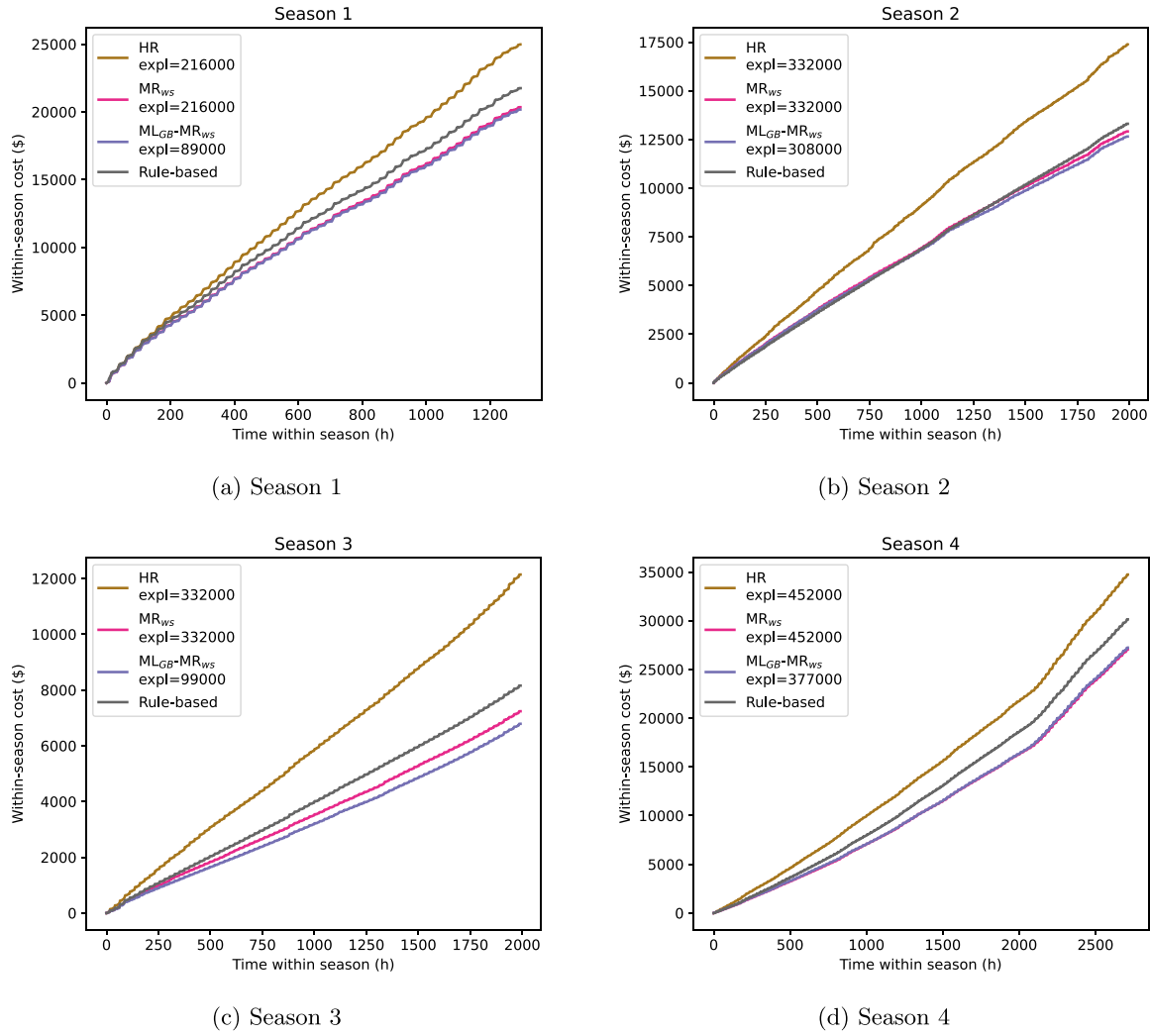


Fig. 9. Mean within-season operating costs (exploratory optimization evaluation counts shown in legends) across 4 runs.

Table 5

Total cost and exploratory evaluation statistics for different approaches in each season. Costs are computed over 4 runs; exploration counts are mean totals within the season. Values in bold represent best in each column.

Approach	Season 1		Season 2		Season 3		Season 4	
	Mean (\$)	Expl. count	Mean (\$)	Expl. count	Mean (\$)	Expl. count	Mean (\$)	Expl. count
HR	24,987	216,000	17,383	332,000	12,143	332,000	34,736	452,000
MR _{ws}	20,352	216,000	12,914	332,000	7240	332,000	27,074	452,000
ML _{GB} -MR _{ws}	20,179	89,000	12,653	308,000	6790	99,000	27,237	377,000
Rule-based ^a	21,756	–	13,308	–	8159	–	30,139	–

^a Single run (no statistics available).

5.3. ML models analysis

5.3.1. Metrics

To analyze the effectiveness of the ML-Accelerated multi-resolution framework's uncertainty-driven exploration strategy, we evaluate the high uncertainty exploration decisions observed across the 4 runs. We focus on two constraints: the terminal SoC and the thermal storage median temperature T_{median} . Let N_{unc} denote the total number of high uncertainty-triggered explorations. For each exploration i , let y_i^{soc} and y_i^{temp} represent the actual terminal SoC and terminal T_{median} values, respectively. We establish uncertainty thresholds of $\theta_{\text{unc},\text{soc}} = 0.01$ and $\theta_{\text{unc},T_{\text{median}}} = 344.15$ K. We define True Positives (TP) as cases where

exploration revealed a significant violation as

$$\text{TP} = \sum_{i=1}^{N_{\text{unc}}} \mathbb{1}(y_i > \theta_{\text{unc}}) \quad (16)$$

and False Positives (FP) as explorations that found negligible violations as

$$\text{FP} = \sum_{i=1}^{N_{\text{unc}}} \mathbb{1}(y_i \leq \theta_{\text{unc}}), \quad (17)$$

where $\mathbb{1}(\cdot)$ is the indicator function. The Precision (Prec) measures the fraction of explorations that discovered significant violations as

$$\text{Precision (\%)} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{100\text{TP}}{N_{\text{unc}}}. \quad (18)$$

The Prediction Interval Coverage Probability (PICP) quantifies the calibration of uncertainty estimates by measuring how often the true value falls within the predicted upper confidence bound (Bazionis and Georgilakis, 2021) as

$$\text{PICP (\%)} = \frac{100}{N_{\text{unc}}} \sum_{i=1}^{N_{\text{unc}}} \mathbb{1}(y_i \leq u_i). \quad (19)$$

Finally, we evaluate the conservativeness of the uncertainty quantification using the Normalized Mean Interval Width (NMIW). To make the width interpretable across different physical units, we express it as a percentage of the feasible operating range:

$$\text{NMIW (\%)} = \frac{100}{N_{\text{unc}}} \sum_{i=1}^{N_{\text{unc}}} \frac{(u_i - \hat{y}_i)}{R_{\text{target}}}, \quad (20)$$

where $\hat{y}_i$ represents the model's predicted mean value, and R_{target} denotes the physical operating range of the target variable ($R_{\text{soc}} = 1$ for battery SoC and $R_{\text{temp}} = T_{\text{max}} - T_{\text{min}} = 20$ K for the thermal storage).

5.3.2. Performance analysis

The analysis of the uncertainty-driven bypass conditions, summarized in Tables 6, 7, and 8, evaluates the efficiency of the ‘‘Compute Path’’ logic defined in Algorithm 2. The metrics reveal a consistently exploitation-averse behavior in the triggering mechanism. The System Precision, which measures the proportion of uncertainty-triggered optimizations that properly triggered exploration, ranges between 24.7% and 29.4% across the three models. This indicates that in approximately 75% of cases where the algorithm detected high uncertainty and reverted to the computationally expensive exploratory optimization. While this implies a computational overhead due to ‘‘false alarms’’, it confirms that the bypass condition effectively prioritizes exploration, ensuring that the fast ‘‘Predict Path’’ is never taken when the model suspects a potential violation.

Decomposing the triggers by objective reveals how the different physical dynamics influence the optimization logic. The strategy is notably conservative regarding the battery's SoC, with precision values consistently remaining below 7%. This indicates that the uncertainty bounds frequently exceed the exploration threshold $\theta_{\text{unc, SoC}}$, triggering the full solver even when the battery operation should be greedy. Conversely, the thermal dynamics drive the majority of the justified computational expenditures. The thermal precision is significantly higher (averaging $\approx 27\%$), suggesting that when the model flags uncertainty, there is a higher probability that the system really needs exploratory optimization for the T_{median} .

To assess the reliability of these triggers, we examine the interplay between the PICP, which measures the frequency with which the true terminal state falls within the predicted uncertainty envelope, and the NMIW, which quantifies how much of the physical operating range is consumed by that uncertainty buffer. The PICP remains high across all experiments, particularly for the RF with $\kappa = 2$ model (Table 8), which achieves aggregate coverage of over 91% for both objectives. Physically, this means that in over 90% of cases where the model flagged uncertainty, the actual terminal state fell within the predicted bounds. However, high coverage is only valuable if the bounds are tight enough to be operationally useful; a trivial model could achieve 100% PICP simply by predicting the entire physical range. The NMIW metric validates that the high PICP scores are achieved with meaningful bounds. For instance, for the thermal storage, the RF model with $\kappa = 2$ generates significantly wider confidence bounds (NMIW $\approx 67.5\%$) compared to the $\kappa = 1$ configuration ($\approx 36.3\%$). In contrast, the SoC uncertainty remains tight, consuming only $\approx 7.1\%$ of the battery's capacity while maintaining $> 92\%$ coverage. Ultimately, the combination of high PICP and bounded NMIW confirms that the uncertainty-driven strategy functions as a robust net, creating necessary, physically constrained buffers rather than vague, infinite estimates. Furthermore, it incurs the cost of precautionary computations to guarantee that the system's

performance rarely falls outside the predicted worst-case scenario, building confidence in the algorithm's ability to bypass the expensive optimization loop when the model is certain.

5.4. Ablation study

In this section, we present the results of an ablation study designed to evaluate the contributions of both the multi-resolution approach and the elite warm starting strategy (Table 9). We compare four different methodologies: MR_{ws} , which represents the multi-resolution approach with elite warm starting; HR, the high-resolution baseline approach without any form of warm starting; MR, which denotes the multi-resolution approach lacking the elite warm starting mechanism; and HR_{ws} , the high-resolution baseline approach that incorporates the elite warm-starting technique. The elite warm starting mechanism implemented in HR_{ws} operates by utilizing only half of the elite solutions as seeds (through random selection), aligning with the mechanics of the other approaches, while the remaining half, typically sourced from lower-resolution models, are randomly initialized.

For the ablation study, we skipped the exploratory stage to save computational time and focus on comparing the core performance of each approach. We ran each method to control the system for 7 days and repeated each experiment 30 times to account for variability and ensure reliable results. All other hyperparameter have remained the same as in the yearly numerical experiments.

Fig. 10 illustrates the mean operating cost over a 7-day period for each of the evaluated methods. The HR approach, which lacks the elite warm starting mechanism, performs the least effectively with a mean total cost of \$4202. In comparison, the HR_{ws} method, which incorporates elite warm starting, demonstrates a significant improvement, reducing the cost by 6% (\$3954). This result is notably comparable to that of the Rule-based controller, which recorded an operating cost of \$3991. These results indicate that introducing warm starting allows the HR method to achieve comparable performance to the baseline rule-based controller, clearly highlighting the effectiveness of the elite warm starting strategy.

Additionally, the multi-resolution strategy without warm starting, denoted as MR, achieves a total operating cost of \$3864 (an additional 2% decrease), outperforming both the HR_{ws} method and the Rule-based controller. This result underscores the advantages of the multi-resolution approach. The top-performing method is the multi-resolution strategy with elite warm starting, MR_{ws} , which records a total cost of \$3760 and the smallest standard deviation of \$29. This suggests a boost in performance and enhanced stability in outcomes, further reinforcing the value of integrating both elite warm starting and multi-resolution approaches within the framework. Our algorithm outperforms standard optimization approaches such as Bayesian optimization; in Appendix H, we present a comparison between all methods discussed in this subsection and TuRBO (Eriksson et al., 2019), a state-of-the-art Bayesian Optimization approach for high-dimensional problems.

5.5. Operating profile comparisons

To further assess the performance of the ML-based multi-resolution strategy, we compare the system's operational characteristics under this control approach to the idealized results from the IDAES optimization.

Fig. 11 shows results for the state of charge of the battery across different seasons in the year. For comparison, we also show the result of the Rule-based control strategy. The ML-based multi-resolution strategy captures most of the trends present in the IDAES solution. This is particularly evident in the Winter case, where the ML-based controller mirrors the IDAES strategy by discharging the battery at midday, coinciding with peak PV generation. In Winter, electricity is cheapest in the morning hours (Fig. C.1, 0–5 h), unlike in other seasons where electricity is cheapest in the middle of the day (10–15 h). As a result, in

Table 6

Uncertainty Exploration Analysis for GB. System column represents the union of violations. All metrics in %.

Run	Total Evals	SoC (%)			T _{median} (%)			System Prec
		Prec	PICP	NMIW	Prec	PICP	NMIW	
1	115	2.6	81.7	0.07	20.0	85.2	47.19	21.7
2	126	5.6	87.3	0.50	31.7	83.3	43.23	31.7
3	129	4.7	89.1	0.36	31.8	80.6	48.54	31.8
4	128	6.2	83.6	1.08	25.8	89.1	48.20	28.1
Avg	124	4.8	85.4	0.51	27.5	84.6	46.79	28.5

Table 7

Uncertainty Exploration Analysis for RF1. System column represents the union of violations. All metrics in %.

Run	Total Evals	SoC (%)			T _{median} (%)			System Prec
		Prec	PICP	NMIW	Prec	PICP	NMIW	
1	139	4.3	84.2	1.06	30.2	79.9	40.27	32.4
2	142	7.7	87.3	3.39	27.5	81.7	37.13	30.3
3	196	5.6	89.3	4.75	21.9	85.7	33.04	24.0
4	177	5.1	91.0	5.20	29.4	82.5	34.76	32.2
Avg	163	5.7	87.9	3.60	26.9	82.4	36.30	29.4

Table 8

Uncertainty Exploration Analysis for RF1. System column represents the union of violations. All metrics in %.

Run	Total Evals	SoC (%)			T _{median} (%)			System Prec
		Prec	PICP	NMIW	Prec	PICP	NMIW	
1	161	6.2	90.7	4.68	24.8	91.9	69.37	25.5
2	200	2.0	95.5	1.93	21.5	94.5	69.63	22.0
3	136	9.6	91.9	9.71	22.8	90.4	60.84	24.3
4	146	9.6	92.5	11.86	25.3	89.7	69.95	28.1
Avg	160	6.4	92.6	7.05	23.5	91.6	67.45	24.7

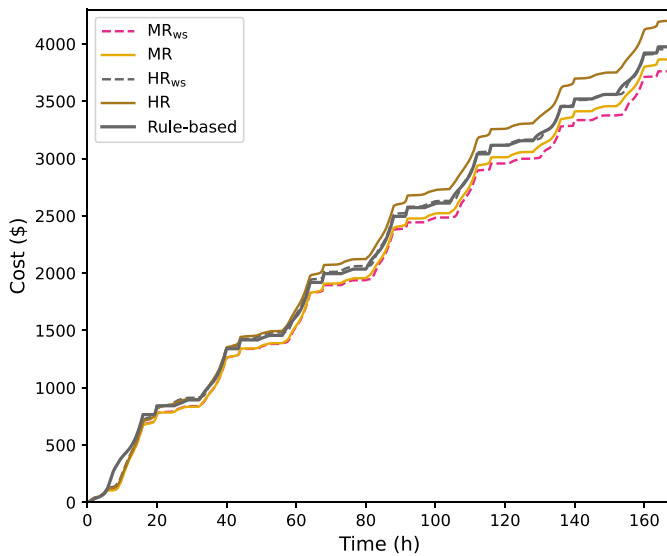


Fig. 10. Mean total operating cost of the energy system over one week of all compared methods for ablation purposes. A lower operating cost at any point in the week indicates better performance.

Winter, all three control strategies charge the battery to full capacity early in the morning. However, the IDAES and ML-based controllers further exploit small Winter price differentials to make additional cost savings by discharging the fully charged battery at the start of plant operation (8–10 h, when electricity is slightly more expensive), and recharging a few hours later (10–12 h), when electricity is cheaper.

Table 9

Cost statistics for the methods in the ablation study.

Approach	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)
MR	3864	3856	3808	4034	46
MR _{ws}	3760	3752	3715	3823	29
HR	4202	4170	4026	4533	117
HR _{ws}	3954	3897	3775	4308	150
Rule-based ^a	3991	–	–	–	–
IDAES ^a	2727	–	–	–	–

^a Single run.

This is overlooked by the rule-based controller, highlighting the ML approach's ability to capture complex operational trade-offs. The ML-based controller also tracks the IDAES behavior more closely than the Rule-based approach in summer, deciding to empty the battery at the end of each day. The most significant deviation we observe in the state of charge is in Spring, where the ML-based strategy discharges the battery completely at the end of the day while the other two approaches carry over energy between days. Despite this difference however, the ML-based strategy still consistently ends up with lower daily operating costs than the Rule-based approach during this season. A similar trend is observed when comparing the grid electricity purchase for the two control strategies where the ML-based controller makes similar purchase decisions to the IDAES model; further details may be found in [Appendix E](#). For the tank temperature, the ML-based strategy was found to produce identical profiles to the rule-based controller, suggesting that the cost savings primarily stem from better battery (SoC) control.

5.6. Limit of performance discussion

Table 10 summarizes the results of the limit of performance estimation problem. For our pilot case, the results indicate that at least 42% of

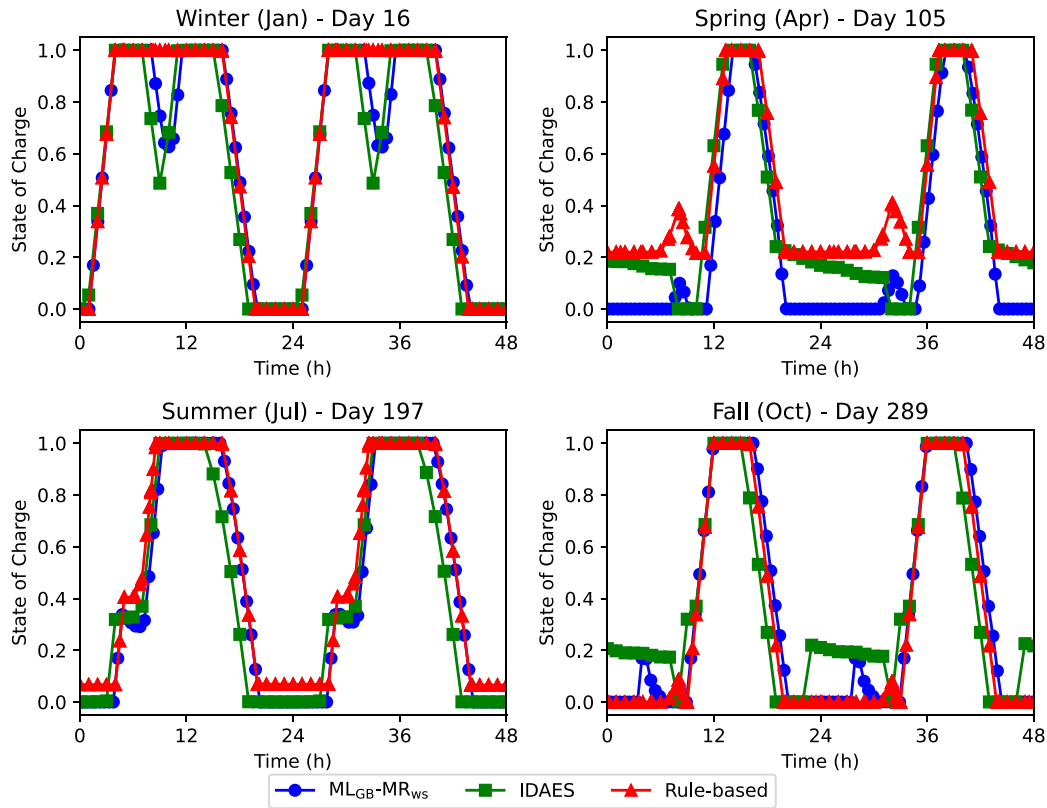


Fig. 11. SOC comparison from different control strategies across seasons.

Table 10
Optimization vs. verification model summary.

Model	Cost	Performance Gap (\$)	(%)
Optimization model	60,284		
Verification model (default rule-based controller)	80,194	19,910	33.03
Verification model (best found via optimization)	71,778	11,494 (↓ 42.2%)	19.07

the performance gap between the optimization model and the original verification model with rule-based control is directly attributable to the control implementation – improving the controller can cut the gap between the models to under 20%. This information is useful to the designer, as such knowledge clearly indicates that further attempts to improve the controller would be beneficial. While the remaining gap is likely due to a process model mismatch, our approaches do not offer optimality guarantees, so such a claim cannot be made definitively. Nonetheless, the results provide a practical limit of performance estimate, offering a meaningful starting point for engineering design.

6. Practical considerations and deployment

The multi-resolution optimization with elite warm starting approach improves performance while remaining straightforward to implement in practice. Building on this, we incorporate ML as a classifier to accelerate the performance of control algorithms, using it specifically to determine when more exploratory optimization is warranted; in this sense, ML serves as a practical accelerator for core engineering tools like optimization, often yielding comparable results in less time. Taken together, we propose a way for engineers to understand the source of performance gaps in energy systems designs and to obtain a first estimate of what is actually achievable with a given system, distinguishing whether these gaps stem from differences in modeling fidelity or from the practical implementation of control strategies. This distinction is critical for evaluating new designs, particularly in the era

of datacenters, where such performance considerations carry increasing significance.

Our proposed algorithm offers several practical advantages for deployment in industrial settings. The ML-based bypass mechanism accelerates optimization, producing quality solutions while reducing online computational cost. A full day-ahead simulation can be solved in under 10 min on a high-end workstation, making the framework practical for decision-making. Furthermore, the framework is built around Modelica-based verification models via the Functional Mock-up Interface standard, which is already widely adopted by industrial energy system designers, allowing it to integrate directly with existing tooling rather than requiring a new, bespoke simulation environment. Furthermore, our open-source implementation of the algorithm and framework allows practitioners to inspect, modify, and deploy it with minimal effort, further supporting its adoption in industrial practice.

Finally, we propose a general framework and showcase it on an industrial heating case study; for practical deployment, several factors need to be considered/adapted to obtain the best performance. The algorithm has several parameters that should be tuned to the specific system it is being deployed upon – we provide some insights on these in [Appendix F](#). In our experience, the most important consideration is how to handle hard constraints, making constraint penalties the most important parameters to be tuned. The constraint penalties should be selected to be high enough to influence the cost function and penalize violations appropriately, but low enough to not burden the

algorithm with unneeded computation. In addition, while we have demonstrated the algorithm with JADE solver, in practical implementation, other derivative-free optimization algorithms should be evaluated, since different algorithms perform well on different types of problems.

7. Limitations and future research

This study demonstrated the efficacy of an ML-accelerated multi-resolution control framework applied to an industrial heating system. The primary objective was to reduce the computational burden of optimal control without sacrificing performance, achieved by learning to predict the optimal terminal states of long-horizon trajectories. By establishing this performance baseline in a controlled environment, we confirmed that ML can effectively bypass expensive optimization steps while obtaining the best operating cost of the evaluated controllers.

We have demonstrated that given a fixed design, the proposed ML-enhanced multi-resolution optimization methodology will yield a fairly good control sequence that outperforms conventional approaches such as Bayesian optimization. We have also shown that our approach is robust to variations in weather and price signals, demonstrating its generalizability. However, our robustness analysis has focused on a single use case with a fixed design; in future work, we will look to demonstrate the methodology on other use cases.

To further enhance the algorithm's efficiency, future work should focus on refining the uncertainty quantification that drives the exploration trigger. While our analysis of the GB and RF models demonstrated high probabilistic coverage (PICP), the low system precision indicates that the trigger is often overly conservative. Although the GB model improves upon the heuristic variance of the RF by using quantile loss, it remains dependent on a fixed, manually selected quantile parameter (α) to define the exploration margin. Replacing these with more rigorous UQ frameworks, such as Conformal Prediction, could yield tighter, statistically guaranteed bounds. This would improve the precision of the trigger and reducing the frequency of unnecessary computational expenses.

A parallel enhancement to the triggering logic is to evolve the static uncertainty threshold, θ_{unc} , into a dynamic, state-dependent parameter. The current use of a fixed threshold presents a significant limitation: it requires manual calibration and assumes a uniform tolerance for uncertainty across all operating regimes. A dynamic threshold would address this by allowing the controller to autonomously adjust its sensitivity based on the system's proximity to constraints or historical volatility. This would eliminate the need for manual tuning and ensure the algorithm adapts its exploration behavior as it encounters new operating conditions.

Finally, this study utilized deterministic price forecasts to provide a clear assessment of the algorithm's core mechanics; this is sufficient for the limit of performance estimation problem. However, real-world deployment of the algorithm for control purposes must account for forecast errors. The current model is reactive to a single predicted future rather than being robust to a range of plausible scenarios. A significant avenue for future research is to integrate stochastic or ensemble-based forecasts.

8. Conclusion

In this work, we sought to understand the gap between the optimization and verification models used in industrial energy systems design via a limit of performance analysis. To achieve this, we propose an online, ML-accelerated multi-resolution optimization framework that estimates architecture-specific upper bounds on achievable performance while minimizing the number of expensive high-fidelity model evaluations. Our ML-guided controller adaptively schedules the optimization resolution based on predictive uncertainty and warm-starts the high-resolution optimization solver using elite low-resolution solutions. We demonstrate our approach on the pilot case of heating an

industrial process. Our results show that the multi-resolution strategy improves both controller performance and optimization tractability, yielding operating cost reductions of up to 24% relative to direct high-resolution optimization and up to 10.5% relative to rule-based control. We show that integrating online, uncertainty-based ML strategies for decision-making on design space exploration can provide significant computational savings without compromising solution quality. Gradient Boosting in particular shows significant potential for our application due to its distribution-free approach to uncertainty quantification. An analysis of the proposed control strategies shows that the ML-based controller mimics the behavior of the idealized IDAES-based control strategy, suggesting that the ML-based approach converges to a high-quality solution. Our proposed approach of combining multi-resolution optimization and ML makes high-fidelity verification tractable and computationally efficient for approximating the limit-of-performance, providing a concrete way to quantify achievable energy-system operating performance prior to industrial deployment.

CRedit authorship contribution statement

Oluwamayowa O. Amusat: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Luka Grbcic:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Remi Patureau:** Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis. **M. Jibran S. Zuberi:** Writing – review & editing, Data curation, Conceptualization. **Dan Gunter:** Writing – review & editing, Software, Project administration, Conceptualization. **Michael Wetter:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

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.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231. This research used the Lawrence Berkeley National Laboratory computational cluster resource provided by the IT Division at the Lawrence Berkeley National Laboratory (Supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231).

Appendix A. IDAES model equations

A.1. Costing

A.1.1. Capital costs

The capital cost of the system is computed as a sum of the costs of the fixed and variable capital costs of the individual components in the overall system architecture,

$$C_{\text{cap}} = \sum_k \tilde{C}_{\text{fixed},k} + \tilde{C}_{\text{var},k} B_k,$$

where k represents the generation and storage technologies available (PV, gas boiler, heat pump, storage tank, battery), $\tilde{C}_{\text{fixed},k}$ are the fixed

Table A.1
Sizing and unit cost parameters for energy system.

Component	Sizing parameter B_k	Fixed cost $\tilde{C}_{fixed,k}$ (USD)	Variable capital cost $\tilde{C}_{var,k}$	Ref.
PV	Capacity - power (kW)	0	3500 USD/kW	Southland (2024)
Gas boiler	Capacity - power (kW)	79,670	367 USD/kW	Southland (2024)
Heat pump	Capacity - power (kW)	0	2055 USD/kW	Southland (2024)
Battery	Capacity - energy (kWh)	0	757 USD/kWh	Bolinger et al. (2023)
Hot water tank	Capacity - volume (m ³)	1520	4845 USD/m ³	Southland (2024)

costs of the components k , $\tilde{C}_{var,k}$ are the variable capital costs, and B_k are the sizing parameters for the different units.

Table A.1 shows the fixed and variable capital costs for each unit, along with the sizing parameters.

A.1.2. Annual operating costs

The annual operating cost is computed as the sum of the hourly costs of electricity purchase from the grid and gas purchase for the boiler over the entire year,

$$C_{op} = \sum_{t=1}^{8760} \tilde{C}_{op,elec} E_{grid}(t, \mathbf{x}) + \tilde{C}_{op,gas} E_{gas}(t, \mathbf{x}),$$

where $E_{grid}(t, \mathbf{x})$ and $E_{gas}(t, \mathbf{x})$ are the hourly energies from the grid and gas purchases, respectively. For our analysis, we use a fixed gas price of $[/0.039]USD/kWh$.

A.2. Process models

A.2.1. Photovoltaics

The PV model calculates the electricity output of the PV system P_{PV} as a function of the solar irradiation G and the installed PV area A as

$$P_{PV} = \eta_{PV} A G,$$

where $\eta_{PV} = 17.1\%$ is the PV efficiency.

A.2.2. Gas boiler

The gas boiler computes the power output from the boiler $\dot{Q}_{boiler}$ (kW) as a function of the volumetric gas flow rate to the boiler v_{gas} as

$$\dot{Q}_{boiler} = \eta_{boiler} v_{gas} HHV,$$

where HHV = 35396 kJ/m³ is the higher heating value of natural gas. The energy generated from the boiler raises the temperature water based on the heat transfer equation as

$$\dot{Q}_{boiler} = m_{water} C_p \Delta T_{water},$$

where m_{water} is the mass flow rate into the boiler (kg/s).

A.2.3. Heat pump

In addition to the heat transfer equations on the evaporator and condenser sides of the heat pump, the heat pump model includes the overall energy balance

$$\dot{Q}_{condenser} + P_{HP} = \dot{Q}_{evaporator},$$

where $\dot{Q}_{condenser}$ and $\dot{Q}_{evaporator}$ represent the energy rates on the condenser and evaporator sides, respectively, and P_{HP} is the electricity into the heat pump. The performance equation of the heat pump is based on a fixed coefficient of performance η_{COP} ,

$$\eta_{COP} P_{HP} = \dot{Q}_{evaporator}$$

A.2.4. Battery

The battery module is based on the standard dynamic battery model that represents the rate of change of energy within the battery as a function of the instantaneous energy rates into and out of the battery while accounting for charging and discharging losses. For the battery model, we constrain the maximum rate of charge (i.e., battery energy to power capacity) to be greater than three hours.

Further details about battery modeling may be found in Amusat et al. (2018).

A.2.5. Hot water tank

The hot water tank is modeled as a cylindrical storage tank with perfect mixing assumptions, hence the temperature within the tank is considered to be uniform with no stratification. The model consists of dynamic energy balances for the storage tank, and accounts for thermal losses from the storage tank to the surroundings. The thermal loss from the tank $\dot{Q}_{loss}$ is computed as a function of the tank state of charge as

$$\dot{Q}_{loss} = U_{tank} A_{tank} (T_{tank} - T_{ambient}) SoC_{tank},$$

where U_{tank} , A_{tank} , T_{tank} , and $T_{ambient}$ denote the overall heat transfer coefficient, tank surface area, tank temperature, and ambient temperature, respectively, and SoC_{tank} represents the tank state of charge, defined as the fraction of stored energy relative to the installed capacity.

Appendix B. Modelica modeling

B.1. Heat production

Heat is provided using a heat pump or a gas boiler.

The heat pump model is *Buildings.Fluid.HeatPumps.Carnot_TCon*. It takes as a control input the condenser leaving temperature and it computes the coefficient of performance at off-design conditions based on a Carnot efficiency. The coefficient of performance at design conditions is $COP_0 = 3$. The electric power consumption of the heat pump compressor is

$$P_{HP} = \frac{\dot{Q}_{con}}{COP},$$

where COP is the coefficient of performance at the current operating temperatures and $\dot{Q}_{con}$ is the condenser heat flow rate.

The gas boiler model is *Buildings.Fluid.Boilers.BoilerPolynomial*. It computes the gas consumption as

$$\dot{Q}_{gas} = \frac{\dot{Q}_{use}}{\eta_{boiler}},$$

where $\dot{Q}_{use}$ is the useful heat and $\eta_{boiler} = 0.9$ is the efficiency.

B.2. Energy storage

There is a battery for electric storage, and a hot water tank for thermal storage.

The battery model is *Buildings.Electrical.DC.Storage.Battery*, which uses charging and discharging efficiency of $\eta = 0.95$

The thermal storage model is *Buildings.Fluid.Storage.StratifiedEnhanced*. It models a stratified storage tank with heat loss to the

environment. The tank also exposes the fluid temperature at various heights, which are used by the rule-based controller described in Section 4.2.

B.3. Electricity production

The electricity is either imported from the grid or locally produced via PV.

The PV production model is *Buildings.Electrical.DC.Sources.PVSimple*. It computes the efficiency as

$$\eta_{PV} = f_{act} \eta_{conv} (1 - \eta_{loss}) = 0.171,$$

where $f_{act} = 0.9$ is the fraction of the active solar cell surface area, $\eta_{conv} = 0.2$ is the initial system conversion efficiency, considering PV panels, wiring, and inverters (NREL, 2025), and $\eta_{loss} = 0.05$ is the conversion loss after 10 years (Jordan and Kurtz, 2013). Similar to the IDAES model, the produced electricity is

$$P_{PV} = \eta_{PV} A G,$$

where A is the collector area and G is the total solar irradiation onto the PV.

B.3.1. Mass flow rates

In the rule-based controls model, the mass flow rates of the pumps for the heat pump condenser, the gas boiler and the heat exchanger are set to

$$\dot{m} = \frac{\dot{Q}_0}{c_p \Delta T_0},$$

where $\dot{Q}_0$ is the nominal heat flow rate, c_p is the specific heat capacity of water, and $\Delta T_0 = 20$ K is the nominal temperature difference between the inlet and the outlet.

For the pump on the evaporative side of the heat pump, the mass flow rate is

$$\dot{m}_{eva} = \frac{\dot{Q}_{eva}}{c_p \Delta T_{eva}},$$

where $\dot{Q}_{eva}$ is the evaporator heat flow rate, $\Delta T_{eva} = 4$ K is the set point of the temperature difference between the evaporator inlet and the outlet temperature.

In model that is used for the limit-of-performance analysis, the mass flow rates of the pump for the condenser and the gas boiler are provided as input from the optimization.

Appendix C. Simulation data for pilot case study

C.1. Electricity prices

Fig. C.1 shows the used hourly electricity price profiles. The prices are based on the fictitious, prototype research price profiles obtained from the California Load Flexibility Research and Development Hub through the CEC's Market Informed Demand Automation Server (Prakash and Paul, 2022).

C.2. Model parameters for IDAES architectural optimization

Table C.1 summarizes the parameters used for the different components in the architectural optimization step.

Appendix D. Generation and storage capacities for Pareto-optimal designs

Fig. D.1 shows installed heat generation and energy storage capacities for the Pareto-optimal designs.

Table C.1
IDAES Model Parameters.

Process Model	Parameter	Value
Photovoltaics	PV efficiency, η_{PV}	17.1%
Heat Pump	Coefficient of performance, η_{COP}	3.0
	Hot side outlet temperature	92 °C
Gas boiler	Boiler efficiency, η_{boiler}	0.90
	Natural gas heating value, HHV	[35, 396] kJ/m ³
	Outlet temperature	92 °C
Battery	Discharging efficiency, $\eta_{discharge}$	0.95
	Charging efficiency, η_{charge}	0.95
	Ambient temperature, $T_{ambient}$	20 °C
	Overall heat transfer coefficient, U_{tank}	0.4 W/m ² K
	Minimum allowable water temperature, T_{tank}^{min}	70 °C
	Maximum allowable water temperature, T_{tank}^{max}	90 °C
	Tank height-to-diameter ratio	1
Load	Water outlet temperature	70 °C

Appendix E. Comparison of control strategies

Fig. E.1 compares amount of electricity purchased from the grid over a 48-hour period in each season. Generally, the two profiles match quite well, with the ML-based controller buying electricity at the same times and at roughly the same levels as the idealized control. The only notable difference occurs in Fall, where the IDAES strategy buys electricity at end of the day to charge the tank, while the ML controller decides to buy at early morning.

Appendix F. Parameter sensitivity analyses

This appendix evaluates the robustness of the proposed ML-accelerated, multi-resolution controller to key hyperparameters. We perform targeted, one-at-a-time sweeps: for each study, we vary only the parameters listed below while holding all other architecture and controller settings fixed. All sensitivity studies use the GB controller variant since it is best performing approach. Moreover, for these sweeps we set the boosted and default evaluation budgets to be equal, i.e., $E_r^{init} = E_r^{def}$ for all resolutions $r \in \{\hat{l}, \hat{h}, \hat{e}\}$. Therefore, the budget configuration reported below corresponds directly to the evaluation limits used by the solver across the entire episode.

F.1. Evaluation budget sensitivity and convergence analysis

In this sweep, we vary the evaluation budgets at each resolution level while keeping everything else fixed. Since $E_r^{init} = E_r^{def}$, the budgets listed below represent the actual evaluation limits. All results are averaged over 10 repetitions of 4-day episodes.

F.1.1. Budget variants

We use the following budget triples:

- (B1) $(E_{\hat{l}}, E_{\hat{h}}, E_{\hat{e}}) = (1250, 1250, 1000)$,
- (B2) $(E_{\hat{l}}, E_{\hat{h}}, E_{\hat{e}}) = (2500, 2500, 2000)$,
- (B3, baseline) $(E_{\hat{l}}, E_{\hat{h}}, E_{\hat{e}}) = (5000, 5000, 4000)$,
- (B4) $(E_{\hat{l}}, E_{\hat{h}}, E_{\hat{e}}) = (8000, 8000, 7000)$.

F.1.2. Convergence and cost results

We report convergence behavior (per-day) and episode-level total operating cost statistics over a 4-day episode.

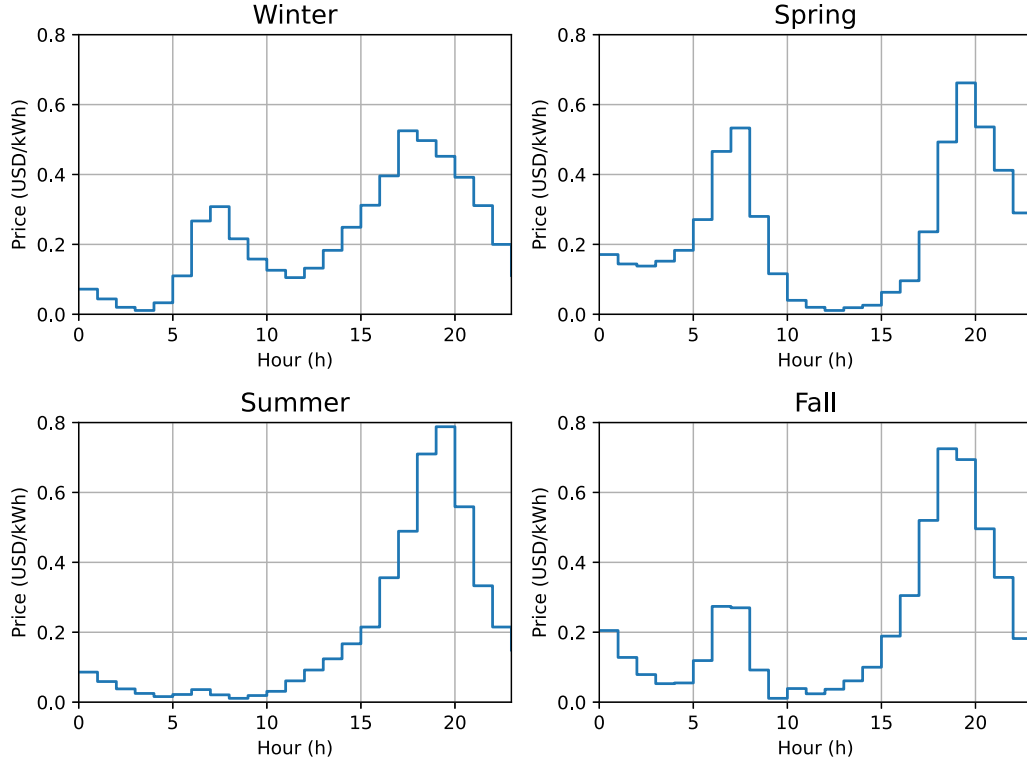


Fig. C.1. Hourly electricity prices per season.

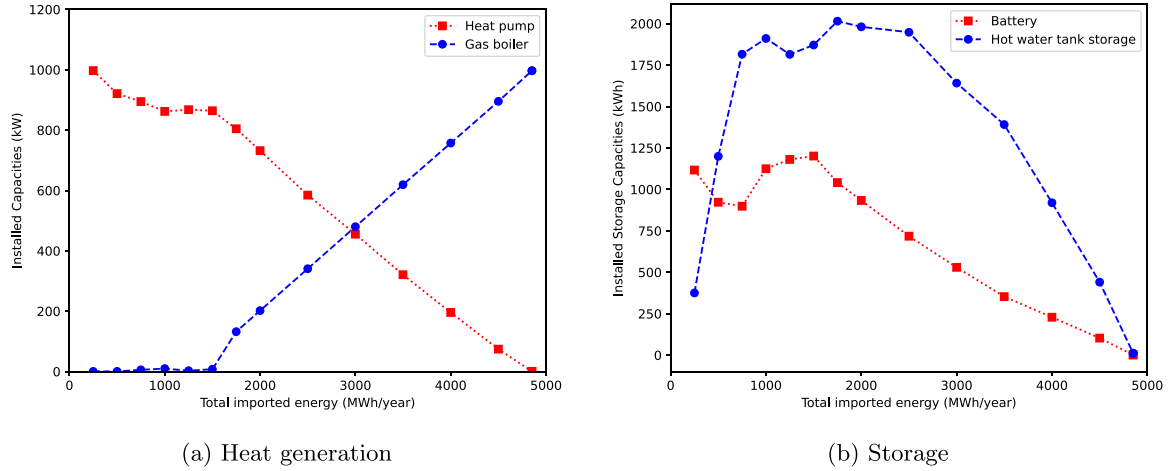


Fig. D.1. Installed capacities of generation and storage technologies for Pareto-optimal designs.

Overall convergence vs. evaluation budget. Fig. F.1 summarizes the episode-level mean total operating cost across Days 1–4 for each budget setting (B1–B4). Increasing the budget from B1 to B2 yields the largest early improvement, but B2 remains noticeably above the best settings. The mean cost reaches its minimum at B3, with B4 providing only a marginal further reduction (from 2463 to 2441, about 0.9%). Table F.1 also shows that run-to-run variability decreases markedly when moving

from B1 to B2, stays moderate at B3, and is lowest at B4 (0.98%). Overall, the results indicate saturation: once the budget reaches B3, additional evaluations yield only diminishing mean improvements.

Per-day convergence under the baseline budget (elite warm-start effect). Fig. F.2 shows mean and standard-deviation bands of the best-so-far fitness by day under the baseline budget (B3). The coarser low-resolution

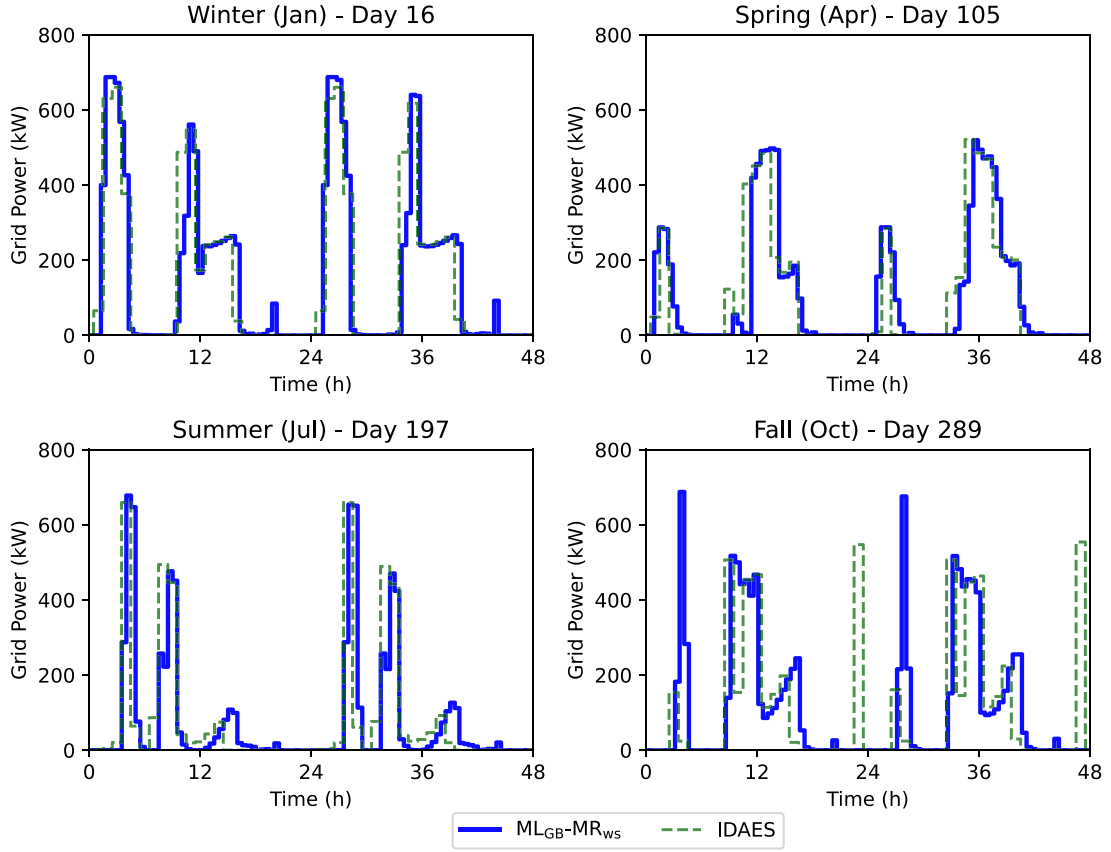


Fig. E.1. Electricity purchase from grid: IDAES vs. ML-based approaches.

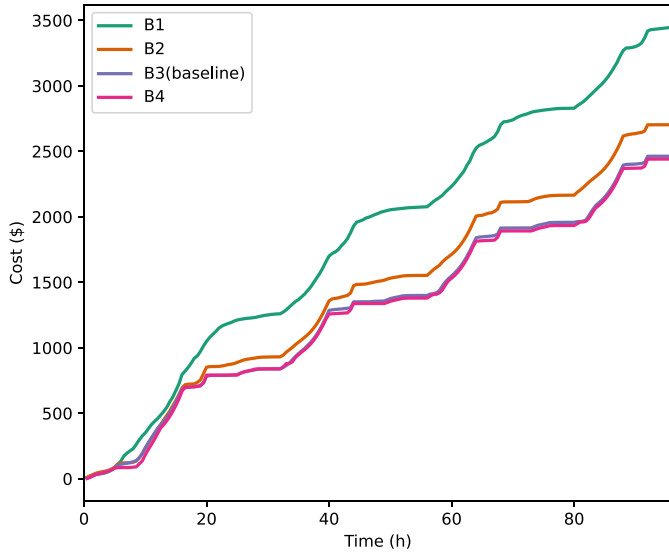


Fig. F.1. Mean total operating cost of the energy system over four days of all compared budget configurations—(E_f , E_h , E_e) (GB variant).

stage drives the dominant early improvements, while the refined stage converges quickly and remains comparatively flat. This is consistent with elite warm-starting: the refined optimization begins from solutions accumulated in the preceding stages, leaving only small residual gains for additional evaluations. The uncertainty bands tighten on later days, indicating reduced run-to-run sensitivity as the episode progresses.

Table F.1

Cost statistics for the maximum evaluation sensitivity of the optimizer over 10 repeated runs, 4 day episodes. Std (%) is computed relative to the mean.

Setting	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
B1	3449	3289	3139	4066	324	9.40
B2	2703	2690	2629	2793	51	1.87
B3 _{base}	2463	2432	2412	2649	67	2.71
B4	2441	2441	2417	2465	24	0.98

F.2. Uncertainty threshold sensitivity (SoC and T_{median})

This study evaluates sensitivity to the uncertainty thresholds used to determine the terminal targets in Algorithm 2: the controller triggers the Compute Path when

$$u_{\text{val}} \geq \theta_{\text{unc},j}.$$

We vary thresholds separately, and report both converged cost and exploratory-evaluation allocation over a 30-day episode and 10 repeated runs.

F.2.1. Threshold variants (separate sweeps)

SoC threshold variants (with T_{median} threshold fixed at baseline).

- Baseline (conservative): $\theta_{\text{unc},\text{SoC}} = 0.01$
- Mid: $\theta_{\text{unc},\text{SoC}} = 0.03$
- Loose: $\theta_{\text{unc},\text{SoC}} = 0.1$

T_{median} threshold variants (with SoC threshold fixed at baseline).

- Baseline (conservative): $\theta_{\text{unc},T_{\text{median}}} = 344.15$
- Mid: $\theta_{\text{unc},T_{\text{median}}} = 346.15$
- Loose: $\theta_{\text{unc},T_{\text{median}}} = 349.15$

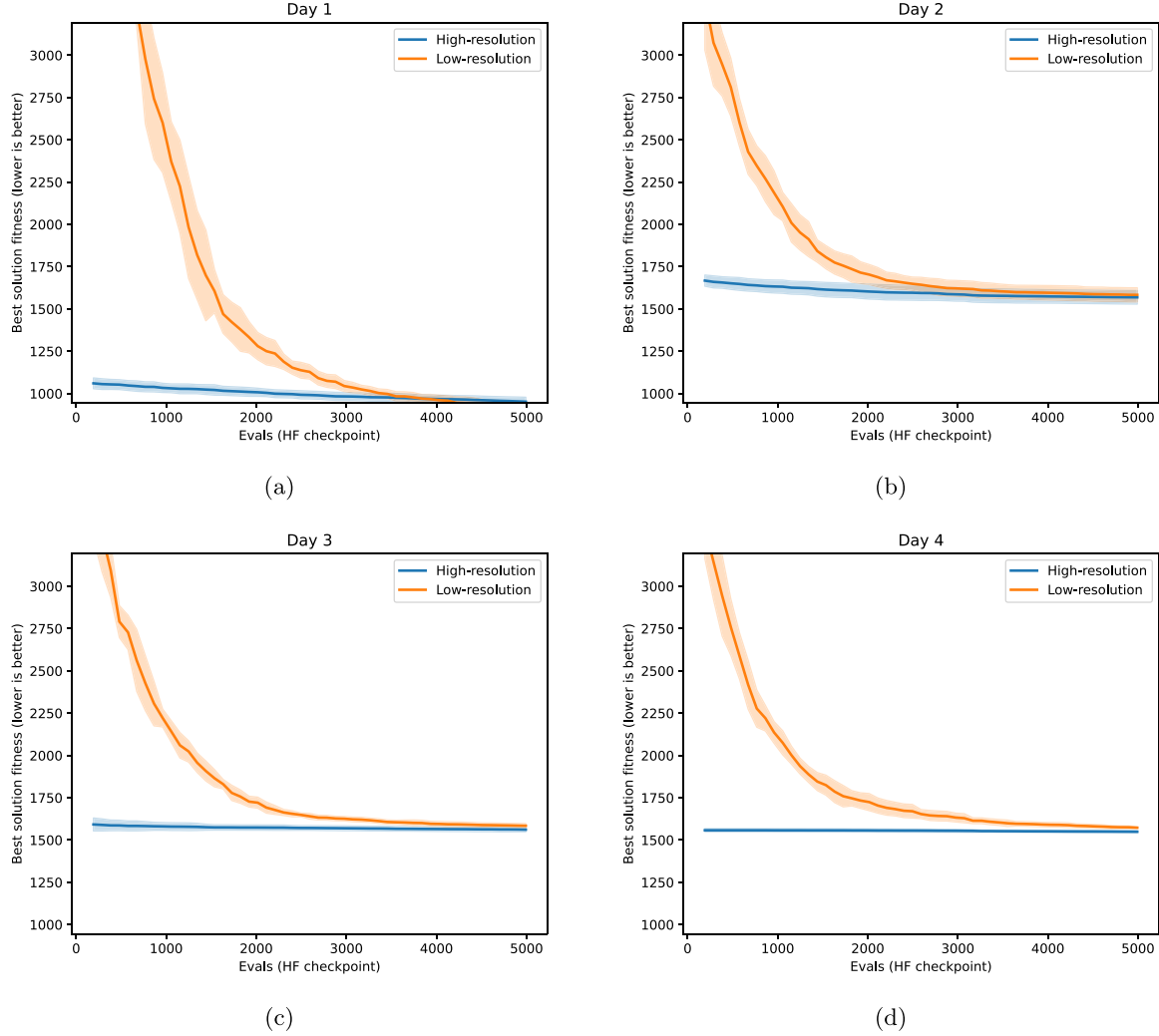


Fig. F.2. Per-day mean convergence of the best found value for High-resolution vs Low-resolution objective functions (GB variant). Shaded region shows the 10 run standard deviation.

Table F.2

Total operating cost statistics for SoC threshold sensitivity (with T_{median} fixed at baseline). Over 10 repeated runs. Std (%) is computed relative to the mean.

Setting	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
$\theta_{\text{unc,SoC}} = 0.01$	12,446	12,432	12,313	12,645	92	0.74
$\theta_{\text{unc,SoC}} = 0.03$	12,419	12,420	12,302	12,553	92	0.74
$\theta_{\text{unc,SoC}} = 0.1$	12,424	12,412	12,305	12,601	82	0.66

SoC uncertainty threshold (with T_{median} fixed at baseline). Table F.2 shows that the episode-level operating cost is largely insensitive to the choice of the SoC uncertainty threshold. Across the three settings, the mean cost varies only within $\sim 0.2\%$ – 0.3% of the baseline and all relative standard deviations remain below 0.74%. This indicates that the controller remains stable even when the exploration trigger becomes more or less conservative with respect to SoC uncertainty.

Moreover, Table F.3 confirms that the threshold does affect how the exploratory budget is spent. As $\theta_{\text{unc,SoC}}$ increases from 0.01 to 0.1, the mean total exploratory budget decreases slightly (from 64,000 to 61,200, $\sim -4.4\%$), while the mean budget allocated to high-UCB decisions drops more substantially (from 8000 to 5200, $\sim -35\%$). The results suggest that looser SoC uncertainty thresholds trigger fewer high-uncertainty exploratory events; a higher threshold makes the algorithm depend more on the ML model.

T_{median} uncertainty threshold (with $\theta_{\text{unc,SoC}}$ fixed at baseline). Table F.4 confirms that the episode-level operating cost is also largely insensitive to the choice of the T_{median} uncertainty threshold. Relative to the baseline mean (12,446), the mean cost changes by only $\sim 0.11\%$ – 0.48% across the tested variants. Moreover, the relative standard deviation ranges from 0.58% to 1.43%, indicating stable closed-loop performance across the threshold sweep.

Table F.5 shows that raising the uncertainty threshold reduces both the total exploratory budget and, more sharply, the portion of it triggered by high-UCB decisions. Going from the baseline to $\theta_{\text{unc},T_{\text{median}}} = 346.15$, the mean total exploratory evaluations decreases from 64,000 to 63,200 ($\sim -1.3\%$), while the mean high-UCB evaluations drop from 8000 to 7200 ($\sim -10\%$). The corresponding high-UCB share of the total falls from 12.5% to 11.4%. Increasing the threshold further to $\theta_{\text{unc},T_{\text{median}}} = 349.15$ continues this trend: the mean total drops further

Table F.3

Statistics of exploratory evaluations over 10 runs for each SoC uncertainty threshold variant. The Total denotes the total exploratory evaluations, while the High UCB Only column denotes the exploratory evaluations triggered by the ML model uncertainty. The standard deviation is derived from the high UCB exploratory runs, while the remaining evaluations are fixed across all runs.

Statistic	Baseline		$\theta_{unc,SoC} = 0.03$		$\theta_{unc,SoC} = 0.1$	
	Total	High UCB	Total	High UCB	Total	High UCB
Mean	64,000	8000	61,600	5600	61,200	5200
Std	10,276	10,276	11,200	11,200	11,321	11,321
Min	56,000	0	56,000	0	56,000	0
Max	88,000	32,000	92,000	36,000	92,000	36,000

Table F.4

Total episode cost statistics for T_{median} threshold sensitivity. Over 10 repeated runs. Std (%) is computed relative to the mean.

Setting	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
$\theta_{unc,T_{median}} = 344.15$	12,446	12,432	12,313	12,645	92	0.74
$\theta_{unc,T_{median}} = 346.15$	12,506	12,414	12,320	12,831	179	1.43
$\theta_{unc,T_{median}} = 349.15$	12,460	12,455	12,355	12,592	72	0.58

Table F.5

Statistics of exploratory evaluations over 10 runs for each T_{median} uncertainty threshold variant. The Total denotes the total exploratory evaluations, while the High UCB Only column denotes the exploratory evaluations triggered by the ML model uncertainty. The standard deviation is derived from the high UCB exploratory runs, while the remaining evaluations are fixed across all runs.

Statistic	Baseline		$\theta_{unc,T_{median}} = 346.15$		$\theta_{unc,T_{median}} = 349.15$	
	Total	High UCB	Total	High UCB	Total	High UCB
Mean	64,000	8000	63,200	7200	62,800	6800
Std	10,276	10,276	10,998	10,998	6705	6705
Min	56,000	0	56,000	0	56,000	0
Max	88,000	32,000	92,000	36,000	72,000	16,000

to 62,800 ($\approx -2.2\%$ relative to baseline) and the mean high-UCB evaluations fall to 6800 ($\approx -15\%$ relative to baseline), giving a high-UCB share of $\approx 10.8\%$.

Similar to the SOC case, the results suggest that looser T_{median} uncertainty thresholds trigger fewer high-uncertainty exploratory events; a higher threshold makes the algorithm depend more on the ML model. However, compared to the SOC case, the impact of the threshold appears to be less pronounced here.

F.3. Penalty weight sensitivity (γ_T , γ_{soc} , γ_{day})

This study varies the penalty weights appearing in the stage/terminal costs: γ_{day} penalizes production shortfall during plant active hours in $\mathcal{L}$, while γ_{soc} and γ_T weight terminal deviation penalties in V_f .

F.3.1. Penalty sets (4-day episodes)

- Baseline: $\gamma_T = 5$, $\gamma_{soc} = 5000$, $\gamma_{day} = 1000$
- Lower end: $\gamma_T = 2.5$, $\gamma_{soc} = 2500$, $\gamma_{day} = 500$
- Higher end: $\gamma_T = 10$, $\gamma_{soc} = 10000$, $\gamma_{day} = 2000$

Penalty weight selection in a robust (conservative) regime. Table F.6 reports the total episode cost statistics over 10 repeated runs (4-day episodes) for three penalty-weight settings. Relative to the baseline case (mean cost \$2462), the lower-end variant yields only a modest change in mean performance (mean \$2409, about -2.15%), while the higher-end variant remains close as well (mean \$2479, about $+0.69\%$). Moreover, the relative standard deviations remain low (below 1.39%), indicating that the closed-loop performance is largely insensitive to moderate rescaling of the penalty weights.

We evaluate load constraint violations across all three variants. Fig. F.3 confirms this robustness at the signal level: during the evaluation window (days 2–4), q_{prod} closely tracks the 1, MW target for all three

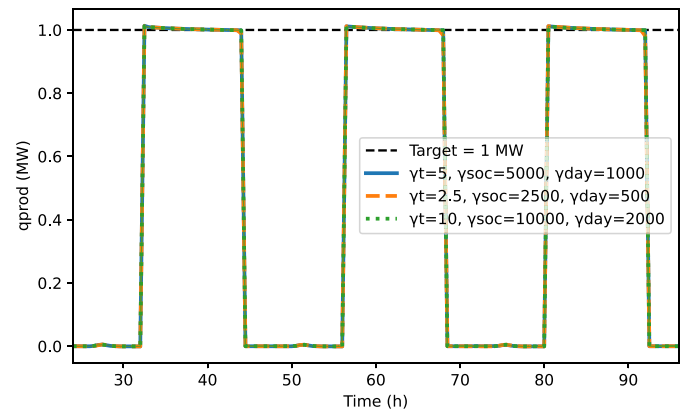


Fig. F.3. q_{prod} over days 2–4 (absolute hours $24 \leq t < 96$) for three penalty-weight settings. Lines show the mean trajectory, and the dashed line indicates the target $q_{prod} = 1$ MW. We skip the first day since it is the warm-up day.

penalty settings, with trajectories nearly overlapping. Fig. F.4, however, shows an example where the lowest-penalty variant briefly violates the constraint.

Consistent with Table F.7, only the lowest-penalty variant exhibits a nonzero violation rate, corresponding to rare underproduction events. The resulting average shortfall is numerically negligible (reported as 0.000, MW at the displayed precision), indicating that any deficits are brief and small in magnitude. Nonetheless, these results demonstrate that the higher penalty value ($\gamma_{day} = 1000$) is necessary to eliminate such violations entirely, compared to the lower value ($\gamma_{day} = 500$).

Table F.6

Total episode cost statistics for penalty weight sensitivity over 10 repeated runs (4-day episodes). Std (%) is computed relative to the mean.

Penalty set	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
$\gamma_T = 5, \gamma_{soc} = 5000, \gamma_{day} = 1000$	2462	2462	2397	2516	34	1.39
$\gamma_T = 2.5, \gamma_{soc} = 2500, \gamma_{day} = 500$	2409	2410	2352	2453	29	1.19
$\gamma_T = 10, \gamma_{soc} = 10000, \gamma_{day} = 2000$	2479	2476	2448	2548	26	1.03

Table F.7

Warm-up excluded load violation statistics across 10 runs for time-of-day window $8:00 \leq t < 20:00$ ($q_{prod} < 1$ MW).

Penalty set	Violation rate (%)	Mean shortfall (MW)
$\gamma_T = 5, \gamma_{soc} = 5000, \gamma_{day} = 1000$	0.000	0.000
$\gamma_T = 2.5, \gamma_{soc} = 2500, \gamma_{day} = 500$	0.145	0.000
$\gamma_T = 10, \gamma_{soc} = 10000, \gamma_{day} = 2000$	0.000	0.000

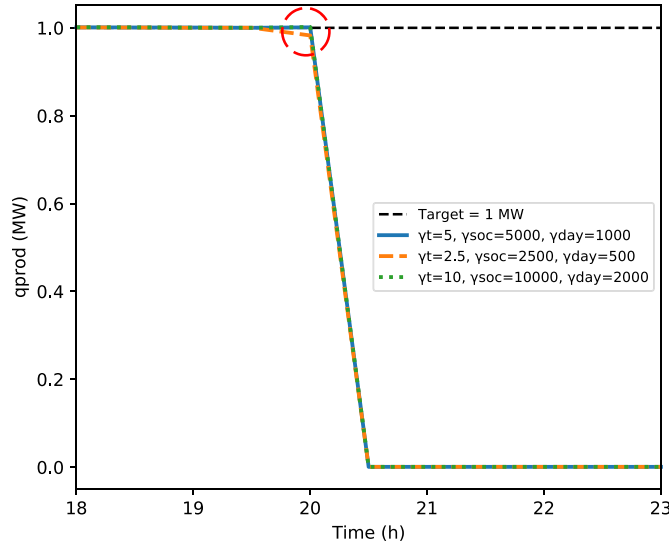


Fig. F.4. q_{prod} over a five hour period for three penalty-weight settings. The dashed line indicates the target $q_{prod} = 1$ MW, and the red circle points to a detail when the lowest penalty weight setting (orange dashed line) breaks the constraints.

F.4. Warm-up period sensitivity (t_{warmup})

In DetermineTerminalTargets (Algorithm 2) we enforce

$$is_warmup \leftarrow (t_k - t_{reset}) < t_{warmup}.$$

During warm-up, the controller executes the Compute Path to gather new observation pairs $(\phi_k, v_{target,j})$ and retrains the models. We test three warm-up durations in 30-day episodes: $t_{warmup} \in \{120, 240, 360\}$ hours and repeat each run 10 times.

Effect of the warm-up duration t_{warmup} . Table F.8 indicates that the controller performance is robust to the choice of the warm-up duration t_{warmup} across the tested range (120, 240, 360 h). Relative to the baseline ($t_{warmup} = 240$ h), the mean episode cost changes only marginally (12,446 to 12,435 at 120 h, and to 12,486 at 360 h), while all configurations exhibit low relative variability ($\leq 0.90\%$).

At the same time, Table F.9 shows that t_{warmup} has a clear impact on how the exploratory evaluation budget is used. Decreasing t_{warmup} from 240 h to 120 h reduces the mean total exploratory evaluations from 64,000 to 45,200 ($\approx -29\%$) and the mean high-UCB portion from 8000 to 5200 ($\approx -35\%$), while increasing t_{warmup} to 360 h raises these to 86,400 ($\approx +35\%$) and 14,400 ($\approx +80\%$), respectively. This also increases the relative share of high-UCB-triggered exploration, from about 12.5% at 240 h to about 16.7% at 360 h.

However, a raw comparison of High-UCB counts is confounded by the fact that longer warm-up periods leave less remaining episode time in which High-UCB triggers can occur (25, 20, and 15 post-warm-up days for the 120 h, 240 h, and 360 h configurations, respectively, within the 30-day episode). To control for this, we normalize by the post-warm-up duration $T_{post} = 30 \text{ days} - t_{warmup}$ to obtain a High-UCB rate,

$$r_{UCB} = \frac{\bar{N}_{UCB}}{T_{post}} \quad [\text{evaluations/day}],$$

where $\bar{N}_{UCB}$ is the mean number of High-UCB evaluations. Table F.10 shows that this rate still increases with t_{warmup} (208, 400, and 960 evaluations/day for 120 h, 240 h, and 360 h), confirming the trend is not simply an artifact of the shrinking post-warm-up window. This suggests that data gathered during a longer, contiguous warm-up block does not proportionally improve the model's coverage of the operating envelope, plausibly because observations within a single extended block are more temporally correlated and less diverse than an equivalent number spread more broadly, leaving the model just as prone to high-uncertainty states once it transitions to the Predict Path. Overall, this indicates that longer warm-up primarily prolongs exploratory activity without improving model confidence, while episode-level operating cost remains comparatively insensitive, consistent with diminishing returns from additional warm-up-driven exploration.

F.5. Periodic trigger sensitivity ($t_{periodic}$)

In addition to uncertainty-triggered exploration, the controller executes the Compute Path at fixed intervals $t_{periodic}$ regardless of model uncertainty, ensuring that the ML models are periodically refreshed with new observations. We test three periodic trigger intervals in 30-day episodes: $t_{periodic} \in \{48, 120, 240\}$ hours, and repeat each run 10 times.

Effect of the periodic trigger interval $t_{periodic}$. Table F.11 indicates that the controller performance is robust to the choice of the periodic trigger interval $t_{periodic}$ across the tested range ($\{48, 120, 240\}$ h). Relative to the baseline ($t_{periodic} = 120$ h), the mean episode cost changes only marginally (12,446 to 12,427 at 48 h, $\approx -0.15\%$, and to 12,461 at 240 h, $\approx +0.12\%$), while all configurations exhibit low relative variability ($\leq 0.74\%$).

At the same time, Table F.12 shows that $t_{periodic}$ has a clear effect on the composition of the exploratory evaluation budget. Decreasing $t_{periodic}$ from 120 h to 48 h (more frequent periodic triggering) increases the mean total exploratory evaluations from 64,000 to 84,000 ($\approx +31\%$), consistent with more frequent scheduled Compute Path executions, but reduces the mean high-UCB evaluations from 8000 to 4000 ($\approx -50\%$). Conversely, increasing $t_{periodic}$ to 240 h (less frequent periodic triggering) reduces the mean total evaluations to 62,400 ($\approx -2.5\%$) while increasing the mean high-UCB evaluations sharply to 14,400 ($\approx +80\%$). This shifts the relative share of high-UCB-triggered exploration from $\approx 12.5\%$ at baseline down to $\approx 4.8\%$ at 48 h, and up to $\approx 23.1\%$ at 240 h.

The raw comparison directly reflects how the frequency of periodic model refreshes affects residual model uncertainty: frequent periodic retraining (48 h) keeps the models' internal state current, curbing the incidence of high-UCB events at the cost of more frequent data gathering and thus high total exploration. On the other hand, infrequent

Table F.8

Total episode cost statistics for seasonal warm-up sensitivity (120, 240, 360 h). Over 10 repeated runs. Std (%) is computed relative to the mean.

Warm-up (h)	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
120 h	12,435	12,405	12,284	12,624	112	0.90
240 h	12,446	12,432	12,313	12,645	92	0.74
360 h	12,486	12,461	12,362	12,619	96	0.77

Table F.9

Statistics of exploratory evaluations over 10 runs for each warmup period variant. The Total denotes the total exploratory evaluations, while the High UCB Only column denotes the exploratory evaluations triggered by the ML model uncertainty. The standard deviation is derived from the high UCB exploratory runs, while the remaining evaluations are fixed across all runs.

Statistic	240 h		120 h		360 h	
	Total	High UCB	Total	High UCB	Total	High UCB
Mean	64,000	8000	45,200	5200	86,400	14,400
Std	10,276	10,276	8953	8953	9831	9831
Min	56,000	0	40,000	0	72,000	0
Max	88,000	32,000	68,000	28,000	96,000	24,000

Table F.10

High-UCB exploration rate normalized by post-warm-up duration, over 10 runs.

Warm-up (h)	Mean High UCB	High-UCB rate (evals/day)
120	5200	208.0
240	8000	400.0
360	14,400	960.0

retraining (240 h) allows uncertainty to accumulate between refreshes, resulting in more high-UCB triggers and substantially higher run-to-run variability.

Overall, this indicates that $t_{\text{periodic}} = 120\text{h}$ represents a favorable middle ground: it avoids the higher total exploratory overhead incurred by more frequent triggering at 48 h, while also avoiding the substantially larger run-to-run variability (like the 240 h configuration).

Appendix G. Computational profiling and runtime savings

This appendix quantifies the computational savings attributable to the ML-driven bypass of the exploratory optimization. Specifically, we compare two approaches:

- **MR_{us}**: the multi-resolution controller that always executes the exploratory optimization on the Compute Path (resolution $r = \hat{e}$), i.e., no ML bypass.
- **ML_{GB}-MR_{us}**: the same controller, but with the ML-accelerated terminal-target logic (Predict Path vs. Compute Path) described in Algorithms 1–2.

The exploratory optimization horizon used in the Compute Path is 48 h. In contrast, the closed-loop evaluation runs for a 30-day episode, and we repeat this evaluation for 10 independent runs. All numerical experiments were conducted on a single compute node equipped with a dual-socket Intel Xeon Gold 6330 (56 physical cores) and 256–512 GB of memory.

G.1. Runtime measurement methodology

For each episode and run, we log the following wall-clock quantities:

- T_{expl} : wall-clock time of the exploratory optimization (Compute Path), i.e., the $r = \hat{e}$ Solve call executed when DetermineTerminalTargets chooses the Compute Path.

- $T_{\text{infer/retrain}}$: wall-clock time of ML activity on the Predict Path, including inference to compute $\hat{v}$ and any GB-model (re)training overhead. This quantity is only defined for ML_{GB}-MR_{us}, since MR_{us} performs no ML computation.
- T_{total} : total wall-clock time over the full 30-day episode, i.e., the sum of all Compute Path and Predict Path cycle times (including T_{expl} and, where applicable, $T_{\text{infer/retrain}}$).

G.2. Profiling results

Table G.1 first confirms that the runtime reduction of the ML_{GB}-MR_{us} variant does not come at the expense of control performance. Over 10 repeated runs, the mean episode-level operating cost decreases only slightly by $\approx -0.23\%$. In addition, the variability in cost reduces, indicating that the ML-guided bypass improves runtime stability without degrading closed-loop behavior.

More importantly, Table G.2 quantifies the source of the computational savings. Overall, introduction of ML for acceleration leads to a mean speedup of 1.16 $\times$ and a wall-clock reduction of $\approx 13.5\%$. This improvement is driven primarily by the reduction in the exploratory optimization time T_{expl} : the mean total exploratory optimization time drops from 4694.10 s (MR_{us}) to 2580.73 s (ML_{GB}-MR_{us}), yielding an exploration time speedup of 1.82 $\times$ (a reduction of $\approx 45\%$).

In contrast, the ML overhead associated with bypass/model activity, captured by $T_{\text{infer/retrain}}$, is comparatively negligible. The mean inference/retrain time is 3.51 s, which is orders of magnitude smaller than both T_{expl} and the total runtime. Consequently, the speedup is not limited by ML inference/retraining costs; rather, the ML model helps avoid frequent calls to the exploratory optimization on the Compute Path.

This interpretation is consistent with the exploration accounting in Table G.3. For a 30-day episode, MR_{us} executes exploratory optimization on every exploratory trigger (120,000 exploratory evaluations). With ML_{GB}-MR_{us}, the mean total number of exploratory evaluations decreases to 62,400 (about $\approx -48\%$), while exploratory events attributed to the ML model uncertainty (“High UCB”) average only 6400. Thus, the ML-driven bypass effectively reallocates computational effort: it largely suppresses exploratory optimization when the learned terminal-target logic is confident, but still retains targeted exploration when the uncertainty indicates potential model mismatch.

Overall, the profiling results show that ML_{GB}-MR_{us} delivers substantial wall-clock savings (via a large reduction in T_{expl}) while maintaining essentially unchanged operating cost, with ML inference/retraining overhead contributing only a negligible fraction of the total runtime.

Table F.11

Total episode cost statistics for seasonal periodic trigger sensitivity (48, 120, 240 h). Over 10 repeated runs. Std (%) is computed relative to the mean.

Periodic trigger (h)	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
48 h	12,427	12,435	12,334	12,524	55	0.44
120 h	12,446	12,432	12,313	12,645	92	0.74
240 h	12,461	12,444	12,385	12,579	68	0.55

Table F.12

Statistics of exploratory evaluations over 10 runs for each periodic trigger variant. The Total denotes the total exploratory evaluations, while the High UCB Only column denotes the exploratory evaluations triggered by the ML model uncertainty. The standard deviation is derived from the high UCB exploratory runs, while the remaining evaluations are fixed across all runs.

Statistic	120 h		48 h		240 h	
	Total	High UCB	Total	High UCB	Total	High UCB
Mean	64,000	8000	84,000	4000	62,400	14,400
Std	10,276	10,276	5060	5060	20,412	20,412
Min	56,000	0	80,000	0	48,000	0
Max	88,000	32,000	96,000	16,000	104,000	56,000

Table G.1

Total episode cost statistics comparing MR_{ws} and ML-Guided MR_{ws} . Std (%) is computed relative to the mean.

Setting	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)	Std (%)
MR_{ws}	12,487	12,453	12,337	12,671	111	0.89
ML- GB - MR_{ws}	12,458	12,462	12,333	12,540	60	0.48

Table G.2

Total runtime and runtime components, averaged over 10 runs of a 30-day episode. T_{expl} sums all logged exploratory optimization times; $T_{infer/retrain}$ sums all logged ML inference and (re)training times on the Predict Path.

Description	Result
Mean T_{total} , without ML (MR_{ws})	13380.4 $\pm$ 133.5 s
Mean T_{expl} , without ML	4694.1 $\pm$ 90.3 s
Mean T_{total} , with ML (ML- GB - MR_{ws})	11577.7 $\pm$ 502.3 s
Mean T_{expl} , with ML	2580.7 $\pm$ 470.7 s
Mean $T_{infer/retrain}$, with ML	3.5 $\pm$ 0.1 s
Overall speedup (T_{total})	1.16 $\times$
Exploratory-stage speedup (T_{expl})	1.82 $\times$

Table G.3

Statistics of exploratory evaluations over 10 runs for the ML- GB - MR_{ws} variant. The Total denotes the total exploratory evaluations, while the High UCB Only column denotes the exploratory evaluations triggered by ML model uncertainty. The MR_{ws} variant always has 120000 exploratory evaluations for a 30-day episode.

Statistic	ML- GB - MR_{ws}	
	Total	High UCB
Mean	62,400	6400
Std	11,342	11,342
Min	56,000	0
Max	92,000	36,000

Appendix H. Comparison with Bayesian optimization (TurBO)

To evaluate how the multi-resolution, warm-started approach performs relative to state-of-the-art black-box Bayesian optimization (BO) methods for high-dimensional optimization, we extend the 7-day ablation study (each run repeated 30 times) with TurBO (Trust-Region Bayesian Optimization) (Eriksson et al., 2019). TurBO is executed

Table H.1

Final cost statistics for different approaches (including TurBO).

Approach	Mean (\$)	Median (\$)	Min (\$)	Max (\$)	Std (\$)
MR	3864.49	3856.91	3808.66	4034.80	46.66
MR_{ws}	3760.85	3752.39	3715.59	3823.40	29.25
HR	4202.21	4170.02	4026.73	4533.41	117.90
HR_{ws}	3954.03	3897.01	3775.55	4308.74	150.95
TurBO	7092.43	7053.14	5816.06	8314.54	599.98
Rule-based ^a	3974.60	–	–	–	–
IDAES ^a	2727.55	–	–	–	–

^a Single run (no statistics available).

in the same control decision space and evaluated using the total closed-loop operating cost over a 7-day horizon. Following the same computational-efficiency considerations as in the main ablation setting, we do not include a separate exploratory stage; all methods are compared directly over the same horizon. Moreover, to encourage constraint satisfaction for TurBO, we increase the penalty factor γ_{day} to 5000. Despite this, TurBO exhibits higher constraint violation rates than the other approaches, for which we did not observe constraint violations under the same evaluation protocol.

For TurBO, we use the default TurBO parameters from the standard implementation. In addition, to make the BO loop computationally tractable in our setting, we set the batch size to 48 and a maximum surrogate model training data size of 512. The TurBO implementation is based on BoTorch (Balandat et al., 2020), and we utilize the settings described in the TurBO tutorial (TurBO, 2024). We report the resulting 7-day cost statistics.

Overall, TurBO produces substantially higher operating costs than all proposed multi-resolution and warm-started variants, with notably larger variance across runs (Fig. H.1, Table H.1). In contrast, the best-performing method among the ablation baselines remains the multi-resolution strategy with elite warm starting (MR_{ws}), achieving both low mean cost and stable performance.

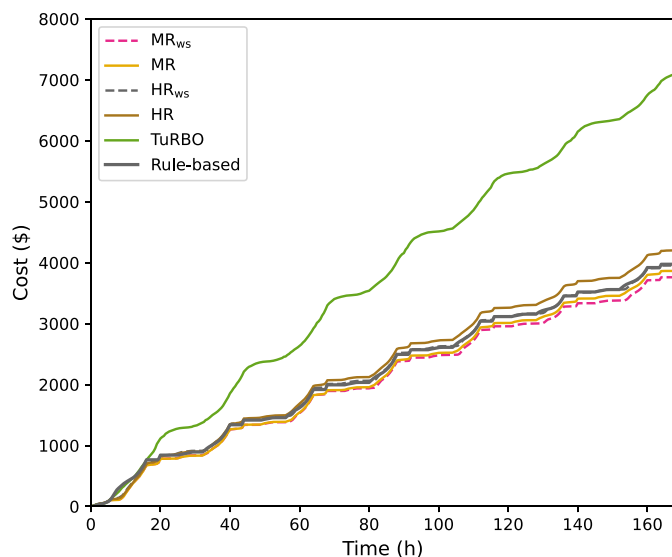


Fig. H.1. Mean total operating cost over one week for all compared methods, including TuRBO. Lower is better.

Data availability

The code needed to reproduce the study can be found on the following repository: <https://github.com/lbl-srg/pbd-industrial-limit-of-performance>.

References

- Abdulla, Khalid, De Hoog, Julian, Steer, Kent, Wirth, Andrew, Halgamuge, Saman, 2017. Multi-resolution dynamic programming for the receding horizon control of energy storage. *IEEE Trans. Sustain. Energy* 10 (1), 333–343.
- Alam, Musharraf, 2019. Day-ahead electricity price forecasting and scheduling of energy storage in LMP market. *IEEE Access* 7, 165627–165634. <http://dx.doi.org/10.1109/ACCESS.2019.2952451>.
- Amusat, Oluwamayowa O., Shearing, Paul R., Fraga, Eric S., 2018. Optimal design of hybrid energy systems incorporating stochastic renewable resources fluctuations. *J. Energy Storage* (ISSN: 2352-152X) 15, 379–399. <http://dx.doi.org/10.1016/j.est.2017.12.003>, URL <https://www.sciencedirect.com/science/article/pii/S2352152X17303365>.
- Arroyo, Javier, Manna, Carlo, Spiessens, Fred, Helsen, Lieve, 2022. Reinforced model predictive control (RL-MPC) for building energy management. *Appl. Energy* 309, 118346.
- Balandat, Maximilian, Karrer, Brian, Jiang, Daniel, Daulton, Samuel, Letham, Ben, Wilson, Andrew G, Bakshy, Eytan, 2020. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Adv. Neural Inf. Process. Syst.* 33, 21524–21538.
- Bazonis, Ioannis K., Georgilakis, Pavlos S., 2021. Review of deterministic and probabilistic wind power forecasting: Models, methods, and future research. *Electricity* 2 (1), 13–47.
- Bolinger, Mark, Seel, Joachim, Mulvaney Kemp, Julie, Warner, Cody, Katta, Anjali, Robson, Dana, 2023. Utility-scale solar, 2023 edition.
- Booker, Andrew J, Dennis Jr., JE, Frank, Paul D, Serafini, David B, Torczon, Virginia, Trosset, Michael W, 1999. A rigorous framework for optimization of expensive functions by surrogates. *Struct. Optim.* 17 (1), 1–13.
- Bre, Facundo, Roman, Nadia, Fachinotti, Víctor D., 2020. An efficient metamodel-based method to carry out multi-objective building performance optimizations. *Energy Build.* 206, 109576.
- Breiman, Leo, 2001. Random forests. *Mach. Learn.* 45 (1), 5–32.
- Cai, Qingsen, Luo, Xingqi, Gao, Chunyang, Guo, Pengcheng, Sun, Shuaihui, Yan, Sina, Zhao, Peiyu, 2021. A machine learning-based model predictive control method for pumped storage systems. *Front. Energy Res.* 9, 757507.
- Eriksson, David, Pearce, Michael, Gardner, Jacob, Turner, Ryan D., Poloczek, Matthias, 2019. Scalable global optimization via local Bayesian optimization. *Adv. Neural Inf. Process. Syst.* 32.
- Favaro, Pietro, Gobert, Maxime, Toubeau, Jean-François, 2024. Multi-fidelity optimization for the day-ahead scheduling of pumped hydro energy storage. *J. Energy Storage* 103, 114096.
- Gao, Yuefen, Zhang, Yiyang, Yun, Chengbo, Huang, Lizhuang, 2025. Online optimization of integrated energy systems based on deep learning predictive control. *Electr. Power Syst. Res.* 243, 111510.
- Grbic, Luka, Müller, Julian, de Jong, Wibe Albert, 2024. Efficient inverse design optimization through multi-fidelity simulations, machine learning, and boundary refinement strategies. *Eng. Comput.* 40 (6), 4081–4108.
- Hewing, Lukas, Wabersich, Kim P., Menner, Marcel, Zeilinger, Melanie N., 2020. Learning-based model predictive control: Toward safe learning in control. *Annu. Rev. Control. Robot. Auton. Syst.* 3 (1), 269–296.
- Jain, Sachin, Tsiotras, Panagiotis, 2008. Trajectory optimization using multiresolution techniques. *J. Guid. Control Dyn.* 31 (5), 1424–1436.
- Jordan, D.C., Kurtz, S.R., 2013. Photovoltaic degradation rates – an analytical review. *Prog. Photovolt., Res. Appl.* 21 (1), 12–29. <http://dx.doi.org/10.1002/pip.1182>.
- Lédée, François, Crawford, Curran, Evins, Ralph, 2025. Improved surrogate modeling for multi-energy system design: Model architecture, sampling and scaling choices. *Appl. Energy* 390, 125812.
- Lee, Andrew, Ghouse, Jaffer H., Eslick, John C., Laird, Carl D., Sirola, John D., Zamarripa, Miguel A., Gunter, Dan, Shinn, John H., Dowling, Alexander W., Bhattacharyya, Debansu, Biegler, Lorenz T., Burgard, Anthony P., Miller, David C., 2021. The IDAES process modeling framework and model library—Flexibility for process simulation and optimization. *J. Adv. Manuf. Process.* 3 (3), e10095. <http://dx.doi.org/10.1002/amp.2.10095>, URL <https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/amp.2.10095>.
- Li, Yuanzheng, Yu, Chaofan, Shahidepour, Mohammad, Yang, Tao, Zeng, Zhigang, Chai, Tianyou, 2023. Deep reinforcement learning for smart grid operations: Algorithms, applications, and prospects. *Proc. IEEE* 111 (9), 1055–1096.
- Liang, Jing, Lou, Yahang, Yu, Mingyuan, Bi, Ying, Yu, Kunjie, 2024. A survey of surrogate-assisted evolutionary algorithms for expensive optimization. *J. Membr. Comput.* 1–20.
- Liu, Qiqi, Lanfermann, Felix, Rodemann, Tobias, Olhofer, Markus, Jin, Yaochu, 2023. Surrogate-assisted many-objective optimization of building energy management. *IEEE Comput. Intell. Mag.* 18 (4), 14–28.
- Mattsson, Sven Erik, Elmqvist, Hilding, 1997. Modelica – An international effort to design the next generation modeling language. In: Boullart, L., Loccufier, M., Mattsson, Sven Erik (Eds.), 7th IFAC Symposium on Computer Aided Control Systems Design. Gent, Belgium, pp. 1–5, URL <http://www.modelica.org/publications/papers/CACSD97Modelica.pdf>.
- Mattsson, S.E., Otter, M., Elmqvist, H., 1999. Modelica hybrid modeling and efficient simulation. In: 38th IEEE Conference on Decision and Control. IEEE, Phoenix, AZ, pp. 3502–3507, URL <http://www.modelica.org/publications/papers/ModelicaCDC99.pdf>.
- Michailidis, Panagiotis, Michailidis, Iakovos, Kosmatopoulos, Elias, 2025. Reinforcement learning for optimizing renewable energy utilization in buildings: A review on applications and innovations. *Energies* 18 (7), 1724.
- NREL, 2025. Champion module efficiencies. URL <https://www.nrel.gov/docs/libraries/pv/champion-module-efficiencies.pdf>.
- Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, et al., 2019. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* 32.
- Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, et al., 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* 12, 2825–2830.
- Perera, A.T.D., Wickramasinghe, P.U., Nik, Wahid M., Scartezini, Jean-Louis, 2019. Machine learning methods to assist energy system optimization. *Appl. Energy* 243, 191–205.
- Polak, Elijah, 1997. Optimization, algorithms and consistent approximations. In: *Applied Mathematical Sciences*, vol. 124, Springer Verlag.
- Prakash, Anand, Paul, Lazlo, 2022. CalFlexHub - California load flexibility research and development hub. URL <https://github.com/LBNL-ETA/CalFlexHub>.
- Prina, Matteo Giacomo, Dallapiccola, Mattia, Moser, David, Sparber, Wolfram, 2024. Machine learning as a surrogate model for EnergyPLAN: Speeding up energy system optimization at the country level. *Energy* 307, 132735.
- Rao, Akshay K., Atia, Adam A., Kneueven, Bernard, Mauter, Meagan S., 2024. Optimizing desalination operations for energy flexibility. *ACS Sustain. Chem. Eng.* 12 (42), 15696–15704. <http://dx.doi.org/10.1021/acssuschemeng.4c06353>.
- Rätz, Martin, Henkel, Patrick, Stoffel, Phillip, Streblow, Rita, Müller, Dirk, 2024. Identifying the validity domain of machine learning models in building energy systems. *Energy AI* 15, 100324.
- Rawlings, Edna, Ghouse, Jaffer, Susarla, Naresh, Sirola, John, Miller, David, 2022. Multiperiod generalized disjunctive programming optimization in idaes: Simultaneous design and operation of an integrated energy system. In: 2022 AIChE Annual Meeting. AIChE.
- Rulff, David, Evins, Ralph, 2025. Systematic refinement of surrogate modelling procedure for useful application to building energy problems. *J. Build. Perform. Simul.* 18 (4), 389–423.
- Sangiovanni-Vincentelli, Alberto, 2007. Quo vadis, SLD? Reasoning about the trends and challenges of system level design. *Proc. IEEE* 95 (3), 467–506. <http://dx.doi.org/10.1109/JPROC.2006.890107>, URL <http://www.gigascale.org/pubs/1023.html>.

- Sha, Xinyi, Ma, Zhenjun, Sethuvenkatraman, Subbu, Li, Wanqing, 2025. Online learning-enhanced data-driven model predictive control for optimizing HVAC energy consumption, indoor air quality and thermal comfort. *Appl. Energy* 383, 125341.
- Shahriari, Bobak, Swersky, Kevin, Wang, Ziyu, Adams, Ryan P, De Freitas, Nando, 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104 (1), 148–175.
- Southland, 2024. Southland industries (personal communication).
- Stoffel, Phillip, Henkel, Patrick, Rätz, Martin, Kümpel, Alexander, Müller, Dirk, 2023. Safe operation of online learning data driven model predictive control of building energy systems. *Energy AI* 14, 100296.
- Su, Tong, Wu, Tong, Zhao, Junbo, Scaglione, Anna, Xie, Le, 2025. A review of safe reinforcement learning methods for modern power systems. *Proc. IEEE*.
- Sulzer, Matthias, Wetter, Michael, Mutschler, Robin, Sangiovanni-Vincentelli, Alberto, 2023. Platform-based design for energy systems. *Appl. Energy* (ISSN: 0306-2619) 352, 121955. <http://dx.doi.org/10.1016/j.apenergy.2023.121955>.
- TuRBO, 2024. TuRBO (trust-region Bayesian optimization) tutorial. https://botorch.org/docs/tutorials/turbo_1. (Accessed 14 July 2026).
- Wächter, Andreas, Biegler, Lorenz T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* 106 (1), 25–57.
- Wetter, Michael, Sulzer, Matthias, 2024. A call to action for building energy system modelling in the age of decarbonization. *J. Build. Perform. Simul.* 17 (3), 383–393. <http://dx.doi.org/10.1080/19401493.2023.2285824>.
- Wetter, Michael, Zuo, Wangda, Nouidui, Thierry S., Pang, Xiufeng, 2014. Modelica Buildings library. *J. Build. Perform. Simul.* 7 (4), 253–270. <http://dx.doi.org/10.1080/19401493.2013.765506>.
- Wilcox, S., Marion, W., 2008. Users Manual for TMY3 Data Sets. Technical Report NREL/TP-581-43156, NREL, Golden, CO.
- Wissmann, Philipp, Hein, Daniel, Udluft, Steffen, Runkler, Thomas A, 2026. Point-wise Q-value maximization for converging Q-learning in continuous state-spaces. In: ESANN.
- Xiao, Zhiwen, Xing, Huanlai, Qu, Rong, Li, Hui, Feng, Li, Zhao, Bowen, Wan, Qian, 2026. Ensemble transitive bidirectional decoupled self-distillation for time-series classification. *IEEE Trans. Syst. Man Cybern.: Syst.*
- Yin, Shihong, Xiang, Zhengrong, 2025a. Adaptive collision avoidance strategy for USVs in perception-limited environments using dynamic priority guidance. *Adv. Eng. Inform.* 65, 103355.
- Yin, Shihong, Xiang, Zhengrong, 2025b. Real-time distributed decision-making for simultaneous target assignment and path planning in multiple unmanned surface vehicles. *Expert Syst. Appl.* 279, 127457.
- Zhang, Peng, Daraz, Amil, Malik, Suheel Abdullah, Sun, Chao, Basit, Abdul, Zhang, Guo-qiang, 2023. Multi-resolution based PID controller for frequency regulation of a hybrid power system with multiple interconnected systems. *Front. Energy Res.* 10, 1109063.
- Zhang, Jingqiao, Sanderson, Arthur C., 2009. JADE: adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* 13 (5), 945–958.
- Zou, Wen, Zhang, Qingbin, Gao, Qingyu, Feng, Zhiwei, 2018. Wavelet multi-resolution approximation for multiobjective optimal control. *PLoS One* 13 (8), e0201514.