# *CAED-Agent*: AN AGENTIC FRAMEWORK TO AUTOMATE SIMULATION-BASED EXPERIMENTAL DESIGN

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The adjustment of parameters for expensive computer simulations is a challenging and universal task in the scientific research pipeline. We refer to these problems as **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign (*CAED*). Traditional approaches include: a) brute force search, which is prohibitive for high-dimensional parameter combinations; b) Bayesian optimization, which struggles to generalize across setup variations and does not incorporate prior knowledge; c) case-by-case experts designs, which is effective but difficult to scale. Recent work on language models (LLMs) as scientific agents has shown an initial ability to combine pre-trained domain knowledge with tool calling, enabling workflow automation. Naturally, replacing the expert's manual design with this automation seems to be a scalable remedy to general *CAED* problems. As will be shown in our empirical evaluations, LLMs lack cost awareness for parameter tuning tasks in scientific simulation, leading to poor and inefficient choices. Inference-time scaling approaches enable better exploration, but the massive additional simulator queries they incur add up to total cost and contradict the target of being efficient. To address this challenge, we propose the **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign Agent (*CAED-Agent*), an agentic framework that combines inference-time scaling with the cost-efficiency feedback from a lightweight surrogate model for solving *CAED* problems. Our experiments in three different simulation cases show that *CAED-Agent* outperforms both Bayesian optimization and LLM baselines by significant margins, achieving success rates comparable to inference-time scaling with a ground truth simulator, while being far more cost-efficient.

## 1 INTRODUCTION

Simulation-based experimental design (Huan et al., 2024) involves tuning parameters for often expensive simulators to achieve a specific design goal. A particular challenge in simulation-based experimental design is to balance the outcome of interest (e.g., accuracy) versus the experimental costs: time, computational power, and financial budgets, aka **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign (*CAED*). Consider the design task in computational fluid dynamics (Anderson et al., 1995), where engineers need to juggle multiple design choices: low vs. high fidelity turbulence models, dimensionality reduction levels, spatial/temporal resolutions, and truncation limits for numerical solvers. Maximum-fidelity settings guarantee accuracy but can extend the runtime of simple simulations from seconds to days, while overly coarse resolutions or model choices may fail to converge or yield unphysical solutions. Striking the correct balance in physical simulators is crucial for accomplishing tasks within reasonable budgets.

Three common approaches address this challenge: 1) Brute force search, while comprehensive, becomes prohibitive as parameter combinations grow exponentially. 2) Bayesian optimization (Snoek et al., 2012; Yao et al., 2024) and evolutionary algorithms (Perera et al., 2023) offer black-box alternatives to grid search, but struggle to generalize across problem variations (e.g., laminar to turbulent flows, different geometry, or fluid to solid mechanics) and cannot leverage prior knowledge from related domains (Char et al., 2019; Trabucco et al., 2022). 3) Expert-based design, while most effective for individual problems, is based heavily on human intervention and remains case-specific, limiting scalability in related problems (Fromer and Coley, 2024; Bharti et al., 2024).

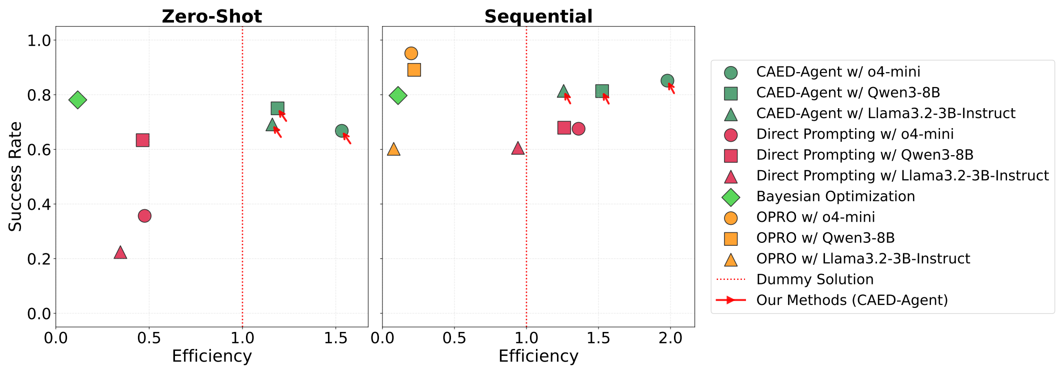


Figure 1: Performance averaged over simulators *Heat 1D* , *Euler 1D* and *NS Transient 2D* , in both Single-Turn setting (left) and Multi-Turn setting (right). *Efficiency* (↑) is the mean normalized cost of the successful simulations; *Success Rate* (↑) is the ratio of successful simulations. Compared to baselines OPRO (Optimization by PROmpting (Yang et al., 2023)) and BO (Bayesian Optimization with Gaussian Process (Nogueira, 2014)), our method achieves Pareto optima in all base models.

To address the scaling limitations of expert-based design, recent work explores large language models (LLMs) (OpenAI et al., 2024; Grattafiori et al., 2024; DeepSeek-AI et al., 2025) as tool-using agents (Ren et al., 2025) to conduct simulation-based experimental designs across domains (Zhong et al., 2024; Lv et al., 2025), potentially providing scalable expert-based intuitive design. These methods often require **inference-time scaling** to produce solutions on par with non-LLM methods. For example, BioDiscoveryAgent (Roohani et al., 2025) and LLAMBO (Liu et al., 2024) iteratively call tools multiple times, adding feedback to prompts for subsequent refinement. However, in real-world deployment, inference-time scaling's computational cost becomes a critical limitation. Other works use token regression to predict experimental metrics (Chen et al., 2022; Song et al., 2024; Tang et al., 2025), such as cost awareness (Wu et al., 2024), for optimization. These works require extensive retraining of the model and tokenizer for specific tools, and assumes cost is a fixed tool property, which is invalid for physics-based simulations where cost depends heavily on parameter choices and simulation scenario setup.

In light of the computational overhead of inference-time scaling and training, some works use **neural-network** (NN) as surrogates for LLM directly calling the simulator (Lyu et al., 2024). However, neural surrogates are usually trained on a specific working condition and does not generalize to alternative settings. Moreover, as surrogates are direct maps from parameters to physical fields with no consideration of computation, they do not provide LLM agents with any insight on using the simulator efficiently. Combined, when encountering out-of-distribution (OOD) scenarios beyond the surrogate's training range, agents must choose between inaccurate results predicted by the NN or calling the costly simulator, harming either the performance or the cost.

We propose training lightweight neural networks that learn only the low-dimensional cost-efficiency signal for cost-efficient inference-time scaling. The benifit is twofold: (1) data collection and model training are made easier as the simulation cost depends only on a few key numerical parameters such as spatial/temporal resolution and interpolation order; and (2) once the signal network is trained, it provides lightweight feedback within LLM inference-time scaling pipelines, enabling effective exploration scaling without expensive simulator evaluations. We call this methodology **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign Agent. Our contributions are as follows.

1. We introduce **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign Agent, a novel methodology that integrates a lightweight neural network that learns only the cost-efficiency feedback with LLM inference scaling frameworks for cost-aware simulation-based experimental optimization. To our knowledge, we are the first to combine low-dimensional cost-efficiency signal neural network with inference scaling for *CAED*.
2. We demonstrate in three physics simulator environments, each with varying environmental setting and precision requirements, that *CAED-Agent* significantly outperforms both traditional Bayesian optimization and state-of-the-art LLM-based approaches.
3. We provide a comprehensive benchmark for the design of simulation experiments based on cost-aware physics. Code for in-house simulator along with benchmark setups are open-sourced at [redacted].

## 2 Related Work

**Non-LLM Methods for Experimental Design and Benchmarks.** Black-box optimization, particularly Bayesian approaches, represents a well-established experimental design framework (Smucker et al., 2018; Huan et al., 2024; Snoek et al., 2012; Knudsen et al., 2021; Pandita, 2019), motivated by the complex procedures underlying experiments or simulators. Recent works explore state-of-the-art methods for experimental design and inverse problems, such as diffusion models (Daras et al., 2024). Benchmarks like Design-Bench (Trabucco et al., 2022) and Inverse-Bench (Zheng et al., 2025a) present 4-5 inverse design problems in scientific domains, respectively. However, few benchmarks or methods explicitly consider evaluation cost as an optimization target, making them unsuitable for our work. Consequently, we use in-house simulators for experiments.

**LLM for Experimental Design.** Recent work explores LLMs in agentic frameworks for scientific experimental design, using inference scaling to sample experimental trajectories. Examples include AI scientists autonomously designing experiments (Wang et al., 2023; Lu et al., 2024; Boiko et al., 2023), automated hypothesis generation frameworks (Zheng et al., 2025b; Wang et al., 2024), and LLM-driven laboratory automation (Bran et al., 2023; Jablonka et al., 2024). While demonstrating the potential of LLMs as experimental designers, these serve as "evidence papers" showing feasibility rather than efficiency. Their pass@k metrics (k up to 256 or even 1024) are specially problematic for expensive experiments. MLEBench (Chan et al., 2025) represents a notable exception by incorporating cost considerations (training time) for LLM on machine learning tasks, but does not capture the nuance of scientific experiment design. In contrast, we specifically target *CAED* as our research target, and contribute in-house simulator, evaluation, and benchmark setups.

**Large Language Models as Optimizers.** Recent studies leverage LLMs as black-box optimizers. OPRO (Yang et al., 2023) uses iterative LLM prompting while LLM-assisted EA (Hao et al., 2024) positions LLMs as evolutionary algorithm surrogatesboth requiring multiple evaluations per step. Other approaches (Song et al., 2024; Chen et al., 2022; Liu et al., 2024; Tang et al., 2025) similarly depend on extensive evaluations for inference or post-training data generation. Their fundamental limitation is expensive trajectory generation requiring abundant simulator queries, computationally infeasible for costly experiments. Our method combines inference scaling with lightweight signal neural network for efficient cost-aware trajectory generation, avoiding expensive simulator queries.

## 3 Methodology

### 3.1 Problem Definition

Given design variable space $\mathcal{X}$ (e.g., spatial/temporal resolution, spatial interpolation methods), environmental parameter space $\Theta$ (e.g., initial or boundary conditions), and output observation space $\mathcal{Y}$, we define the forward simulation-based experimental process as $\mathcal{F} : \mathcal{X} \times \Theta \to \mathcal{Y}$:

$$y = \mathcal{F}(x, \theta), \quad \text{where } \; x \in \mathcal{X}, \; \theta \in \Theta \tag{1}$$

With utility function $\Phi : \mathcal{Y} \times \Theta \to \mathbb{R}$ (e.g. representing accuracy or physical validity of simulated results) and cost function $\mathbf{C} : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$ (e.g. wall time, complexity analysis, RAM consumption), the *CAED* problem becomes:

$$x^* = \arg\max_{x \in \mathcal{X}} \; \Big(\Phi(y, \theta), \; -\mathbf{C}(x, y, \theta)\Big) \tag{2}$$

In this work, we define computational cost as the number of floating point operations (consistent with complexity analysis) and normalize cost relative to a brute-force reference (dummy) solution $z_\theta$ that satisfies accuracy requirements with optimal cost (within a coarse search granularity):

$$\hat{\mathbf{C}}(x, y, \theta) = \frac{\mathbf{C}(x, y, \theta)}{\mathbf{C}(z_\theta, \theta)}. \tag{3}$$

Following previous works (Snoek et al., 2012; Fromer and Coley, 2024), we combine the normalized cost and utility objectives into a single reward metric for an experiment $(x, y, \theta)$:

$$\mathcal{R}^0(x, y, \theta) = \frac{\Phi(y, \theta)}{\hat{\mathbf{C}}(x, y, \theta)} \tag{4}$$

We consider two variants of the *CAED* problem: Single-Turn *CAED*, where the algorithm proposes only one configuration, and Multi-Turn *CAED*, where the algorithm proposes a trajectory of configurations for iterative refinement (Huan et al., 2024; Bharti et al., 2024)):

**Definition 3.1** (Single-Turn **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign (*CAED*) )**.**

$$\mathcal{Q}_0 : x^* = \arg\max_{x \in \mathcal{X}} \mathcal{R}^0(x, y, \theta), \tag{5}$$

**Definition 3.2** (Multi-Turn **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign (*CAED*) )**.**

$$\mathcal{Q}_m : \{x\}^* = \arg\max_{\{x_1,\ldots,x_n\} \in \mathcal{X}^*} \mathcal{R}^s(\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_n\}, \theta), \tag{6}$$

where $\mathcal{X}^*$ is a sequence consisting of an arbitrary number of elements from $\mathcal{X}$. In this work, we allow multi-turn solutions with any length. $\{y\} = \{y_1, y_2, ..., y_n\}$ are observations from sequence $\{x\} = \{x_1, x_2, ..., x_n\}$, and the modified multi-turn reward $\mathcal{R}^m$ is:

$$\mathcal{R}^m(\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_n\}, \theta) = \frac{\max_i \Phi(y_i, \theta)}{\sum_i \hat{\mathbf{C}}(x_i, y_i, \theta)}, \tag{7}$$

i.e., the ratio between maximum utility and total cost incurred by this sequence of proposals.

The two variants of the *CAED* problem, $\mathcal{Q}_0$ and $\mathcal{Q}_m$, are distinct and have different metrics with different reference solution $z_\theta$. They evaluate different abilities of the solution: $\mathcal{Q}_0$ requires an intuitive choice of simulation parameter, while $\mathcal{Q}_m$ requires adaptation based on simulation feedback. They are not to be recognized as the same task with a varying hyperparameter (number of turns).

## 3.2 Cost-Aware Simulation-Based Experimental Design Agent

**Overview.** We adopt the inference-time scaling framework of Optimization by PROmpting (OPRO) (Yang et al., 2023; Song et al., 2024; Chen et al., 2022), with the addition of a module that efficiently provides utility $\Phi(x, y, \theta)$ and cost $\hat{\mathbf{C}}(x, y, \theta)$ information without calling the expensive ground-truth simulations. Specifically, we train a neural-network surrogate to predict these scalar signals from the design variables and environmental parameters. Because the scalar outputs are strongly correlated with a few key design variables, signal model training converges with fewer samples and smaller model size, compared with full-physics surrogates (Ghafariasl et al., 2024; Hou and Evins, 2024); see C for details. The signal model then supplies feedback, the predicted utility and cost, to the LLMs proposed parameter designs. These feedback signals, recorded as designvalue pairs, are appended to the prompts history as in-context examples to aid the LLMs optimization output. See Figure 2 for an illustration of *CAED-Agent*s workflow.

**Signal Neural Network.** We train lightweight networks $\mathcal{S} : \mathcal{X} \times \Theta \to \mathbf{D}_\Phi \times \mathbb{R}$ to predict utility and cost signals only, where $\mathbf{D}_\Phi$ is the short-hand for the range of utility function $\Phi$. Our experiments show that small fully connected neural networks can learn the function well for the experiments in this paper, though we note that architecture and model size can be adapted according to the need of specific solvers. See Appendix C for details on neural network implementation for this paper.

To provide rich, informative utility signals, as opposed to the binary boolean signals in prior works (Smucker et al., 2018; Huan et al., 2024), we design a reward shaping function $f$ that maps the binary experiment outcome $b(y) \in \{0, 1\}$ to a scalar soft success measure $f(y) \in [0, 1]$ defined as follows.

**Definition 3.3** (Soft Utility Function)**.** Following the notations in section 3.1, let $\mathcal{Y}$ be the experiment's observation space, $\Theta$ be the environmental variable space, and $\Phi$ be the original utility function. Define the feasible set

$$\mathcal{G}_\theta := \{\, y \in \mathcal{Y} : \Phi(y, \theta) = 1 \,\}. \tag{8}$$

We call a mapping $f : \mathcal{Y} \times \Theta \to [0, 1]$ a *soft utility function* if it satisfies:

(i) Feasibility calibration: $\forall y \in \mathcal{G}_\theta : \; f(y) = 1, \quad \sup_{y \notin \mathcal{G}_\theta} f(y) \; < \; 1.$

(ii) Normalization: $0 \; \le \; f(y) \; \le \; 1, \quad \forall y \in \mathcal{Y}.$

(iii) Monotone alignment: $\Phi(y_1) \; \preceq \; \Phi(y_2) \implies f(y_1) \; \le \; f(y_2).$

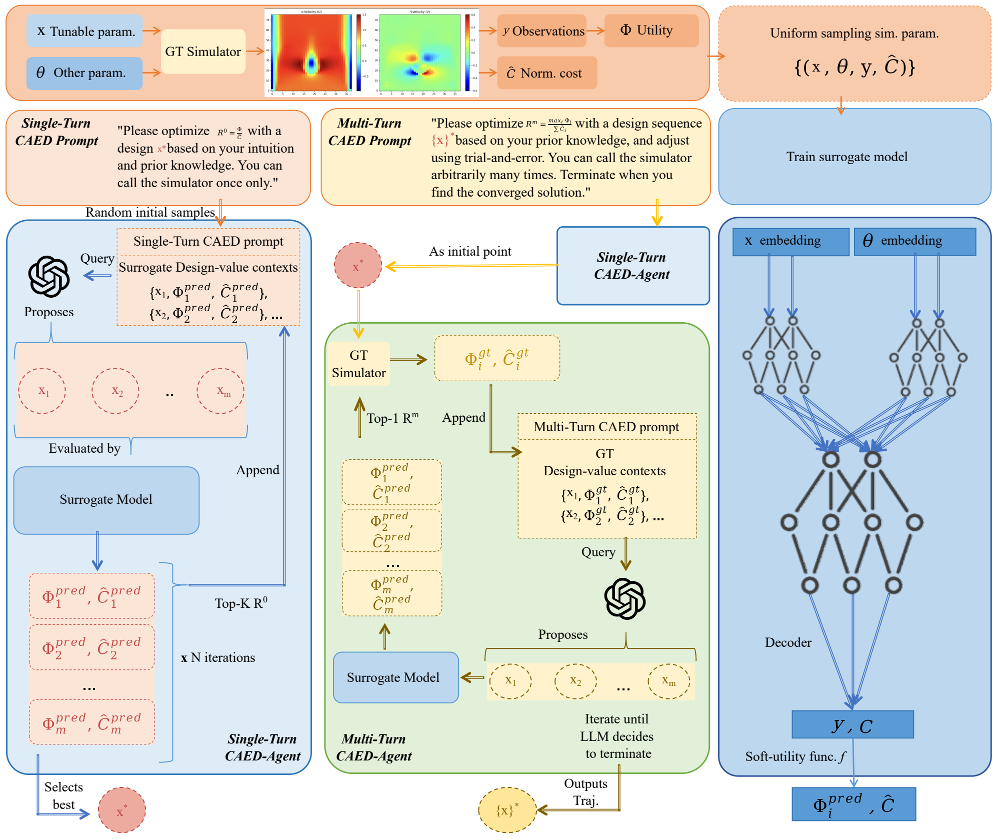


Figure 2: Overview of *CAED-Agent*. For a given simulation-based experimental design problem, *CAED-Agent* samples uniformly within design space to train a neural network surrogate for feedback signals of utility and cost (right). In the Single-Turn CAED setting (left), the LLM agent proposes an ensemble of candidate designs, calls the simulator to obtain feedback, and extend the design-score pairs history for the next iteration of candidate ensemble generation. Finally the LLM selects the best design at the last iteration based on surrogate signals. In the multi-turn CAED setting (middle), the LLM sends the best design based on surrogate signal for actual evaluation at each decision step, receives actual evaluation feedback alongside surrogate signals, and output the trajectories of designs sent for ground-truth evaluation as solution.

The signal neural network $\mathcal{S}$ learns the soft utility signal $f(y)$ in the place of $\Phi(y)$. We provide the following proposition that any soft utility function $f$ guarantees an incremental performance over binary utility functions when integrated into our framework, and a well-designed $f$ will lead to more significant improvements. Refer to Appendix D for our design of $f$ and proofs of the proposition.

**Definition 3.4** (Policies)**.** Recall that $R^0(x, y, \theta)$ and $\mathcal{R}^m(\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_n\}, \theta)$ are respectively single-turn and multi-turn reward defined in Eq.1). For a task instance $\theta$ , we define two policies:

(i) **Binary-utility policy** $\pi_{\text{bin}}(x_t \mid \theta, h_{t-1})$: at step $t$, given history $h_{t-1} = \{(x_s, y_s, b(y_s, \theta))\}_{s=1}^{t-1}$, sample the next design $x_t$; denote the induced distribution over the final design by $x \sim \pi_{\text{bin}}(\cdot \mid \theta)$.

(ii) **Soft-utility policy** $\pi_f(x_t \mid \theta, h_{t-1})$: replace $b$ with any soft utility $f$ from Definition 3.3, i.e., the history stores $(x_s, y_s, f(y_s, \theta))$. Denote the resulting final-design distribution by $x \sim \pi_f(\cdot \mid \theta)$.

**Proposition 3.5** (Soft utility dominates binary utility in expected reward)**.** *Fix a base model and any soft utility $f$ in Definition 3.3, the expected reward under the soft-utility policy is no worse than under the binary-utility policy:*

$$\mathbb{E}_\theta \, \mathbb{E}_{x \sim \pi_f^0(\cdot|\theta)} \big[ R^0(x, \theta) \big] \;\geq\; \mathbb{E}_\theta \, \mathbb{E}_{x \sim \pi_{\text{bin}}^0(\cdot|\theta)} \big[ R^0(x, \theta) \big] .$$

$$\mathbb{E}_{\theta}\,\mathbb{E}_{\{x\}\sim\pi_f^m(\cdot|\theta)}[R^m(\{x\},\theta)] \;\geq\; \mathbb{E}_{\theta}\,\mathbb{E}_{\{x\}\sim\pi_{\mathrm{bin}}^m(\cdot|\theta)}[R^m(\{x\},\theta)]\,.$$

In summary, for a given simulation-based experimental design task, we train a lightweight network $\mathcal{S} : \mathcal{X} \times \Theta \to \mathbf{D}_{\Phi} \times \mathbb{R}$ to predict a certain design's utility and cost; in cases where utility function $\Phi$ is sparse and less informative, we substitute it with soft utility function $f$ and learn soft utility signals, i.e. we learn $\mathcal{S} : \mathcal{X} \times \Theta \to \mathbf{D}_y \times \hat{\mathbf{C}}$, where $\mathbf{D}_y$ is the range of $f$. The trained network $\mathcal{S}$ provides feedback for the following agent's self-refinement.

**Agentic Framework.** The agent leverages Optimization by PROmpting (OPRO) as the base LLM in-context optimization method, and use the signal network's feedback as in-context examples. We note that expanding to other inference-time scaling methods is straightforward and requires no change or re-training of the signal neural network. Pseudocode for our agent implementation is provided in Appendix B.

For Single-Turn *CAED*, the agent starts with 5 (a hyper-parameter to adjust based on inference budget) uniformly-sampled tuples of (design variable, utility, efficiency) evaluated by surrogate neural network. Then the agent iteratively proposes ensembles of candidate design choices, receives neural network feedback for the entire ensemble, and append them to the example pool. The example pool is managed as a priority queue with key (utility, efficiency) and presented to the model in ascending order.The example pool only keeps top-10 samples (also a hyper-parameter) to concise the context. The process is repeated for a fixed number of iterations, and the best design in the example pool is chosen for the final design. The fixed number of iterations is another hyperparameter reflecting the allowed LLM inference budget.

To solve Multi-Turn *CAED*, we warm-start with Single-Turn *CAED* solution for the first round of ground truth simulator evaluation, and then append the results to the pool. This process is repeated for each iteration to find the most promising proposals for simulator evaluation. In short, the Single-Turn *CAED* works as an acquisition function for each of the multi-turn steps. The loop terminates when either the LLM decides that a satisfactory solution is found or the computation cap is reached.

## 4 Experiments

### 4.1 Experimental Environment

We demonstrate the ability of *CAED-Agent* on three physics simulators: (1) 1D heat conduction equation with mixed boundary conditions, (2) 1D compressible inviscid flow with Euler equation, and (3) 2D transient incompressible Navier–Stokes equation, referred to as *Heat 1D* , *Euler 1D* and *NS Transient 2D* respectively, for brevity. Appendix A contains details on the design variable space $\mathcal{X}$, observation space $\mathcal{Y}$, and parameter space $\Theta$. We focus on spatial resolution tuning tasks, where the tunable parameter governs the spatial resolution of the simulation, creating a trade-off between simulation accuracy and computational cost. The tunable parameters in our experiments are:

1. The number of grid numbers ($n\_space$) for *Heat 1D* and *Euler 1D*
2. The grid resolution along X-axis ($resolution$) for *NS Transient 2D*

We design three precision level goals $\delta$ for each task, reflecting moderate to stringent accuracy requirements in real-world experiments. For each task and each precision level, we evaluate the methods on around 25 settings varying in environmental parameters.

For each problem instance characterized by $\theta$, we first obtain a (near-)optimal design $z_\theta$ via brute-force search that guarantees successful convergence, e.g. through iteratively doubling the parameter until successful, serving as a reference point for both accuracy and cost. This is solely for the evaluation of our method and not necessary in practice. We then define the success of the simulation through the following utility function:

$$\Phi(\mathcal{F}(x,\theta),\theta) = \mathbf{1}\{\, ||\mathcal{F}(x,\theta) - \mathcal{F}(z_\theta,\theta)||_2 \;\leq\; \delta\,\}\,, \tag{9}$$

Where $\mathbf{1}$ is the indicator function, $||\cdot||_2$ is the root mean square error across dimensions of the observation space, and $\delta$ is a tolerance parameter reflecting various precision needs in real-world applications. The success rate is defined as the ratio of successful simulations where $\Phi(\mathcal{F}(x,\theta)) = 1$. The cost $\mathbf{C}$ is defined as previously introduced in our problem formulation.

## 4.2 Baselines and Setting

We compare our results against the following baselines. **Bayesian Optimization** (*BO*): We use a classic implementation (Nogueira, 2014) with Gaussian Process (GP) (Rasmussen, 2004) and Upper Confidence Bound (UCB) (Berk et al., 2020). We used consistent training samples for the signal neural network *CAED-Agent* and for the GP regressor to achieve a fair comparison. **Direct query to LLM medels** (*Direct Query*) and the original **Optimization by Prompting** (*OPRO*) (Yang et al., 2023) are LLM-based approaches. For all LLM-based methods (including our *CAED-Agent*), we design a shared set of prompts explaining the Physics scenario, optimization target and simulator calling APIs; refer to E for examples. Notably, *OPRO* requires repeated evaluations of the ground-truth simulator; therefore, we restrict its use to the Multi-Turn setting.

For the implementation of *CAED-Agent*, we trained a lightweight neural network for each task (*Heat 1D* , *Euler 1D* , and *NS Transient 2D* ) separately, each with approximately 10k parameters and trained on about 4k sampled points per problem. The networks outputs are the RMSE to the reference solution and the cost. At inference time, we map the predicted RMSE to a utility signal using the soft utility functions described in Definition 3.3; we also compare using the binary utility $\Phi$ in ablation studies. See Appendix C for details.

## 4.3 Metrics

For ease of future reference, we denote the optimization targets in 5 and 6 as respectively $R^0$ and $R^m$, referring to them as Single-Turn or Multi-Turn *Reward Functions*. We also report success rates $P^0$ and $P^m$ to help us better understand the qualities of proposed solutions.

$$R^0 = \frac{\Phi(\mathcal{F}(x,\theta),\theta)}{\hat{\mathbf{C}}(x,\theta)}, \quad R^m = \frac{\max_i \Phi(\mathcal{F}(x_i,\theta),\theta)}{\sum_i \hat{\mathbf{C}}(x_i,\theta)}$$

## 4.4 Analysis

We refer readers to Table 2 of F for complete results in three scenarios; here we report the following findings that help understand and verify the efficacy of our method.

**Our method outperforms most baselines in terms of $R^0$ and $R^m$.** As shown in Figure 3 and Figure 1, our method outperforms all comparisons in the Single-Turn setting and all but a few exceptions in the Multi-Turn setting. We argue that these sub-optimal cases are due to the inferior reasoning ability of open-source models, causing them to occasionally fail to refine their solutions based on feedback. Note that in many cases, especially in the easier scenarios *Heat 1D* and *Euler 1D* , OPRO and BO are significantly worse than Direct Query, whereas our method is significantly better. This is because convergence is relatively easy in such scenarios, so the additional ground-truth simulator calls used by OPRO and BO incur extra cost without meaningfully improving the solution. Our method does not require additional ground-truth simulator queries.

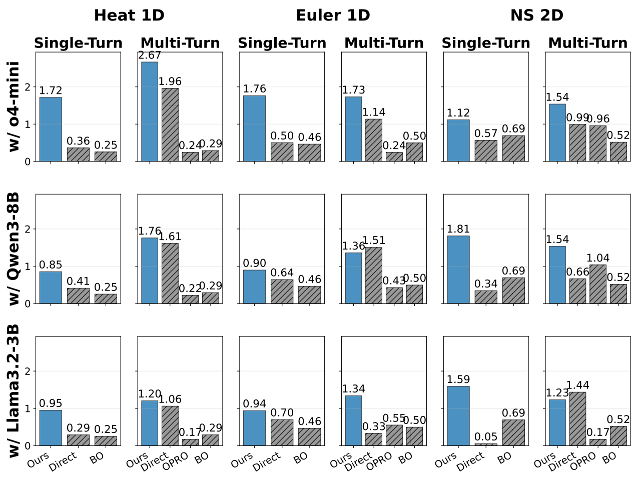


Figure 3: **Comparison of Single-Turn and Multi-Turn rewards for all methods.** Each bar shows the mean reward, averaged over all precision levels of a task, for methods on a given base model. As discussed in definition 4.2, OPRO is only considered in the Multi-Turn scenarios. BO is plotted alongside LLM methods for clarity of comparison.

**Our method delivers substantial reward gains over Direct Query, especially on medium- and easy-difficulty tasks; on harder tasks, it consistently improves success rate.** As shown in Figure 4, reward improvements are most pronounced in easier scenarios (*Heat 1D* ; low-precision *Euler*

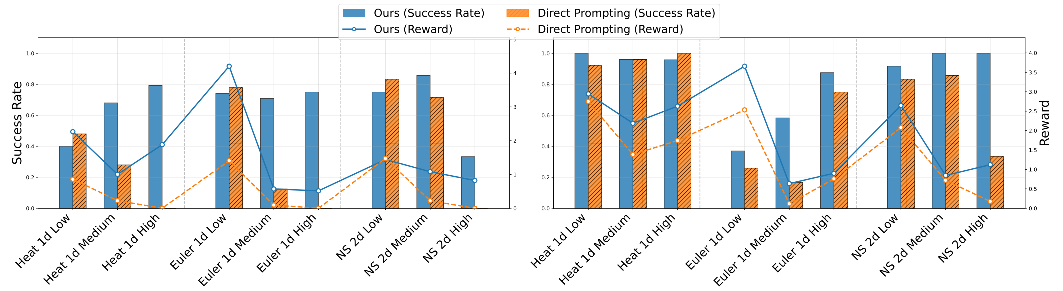


Figure 4: **Reward** ($R^0$, $R^m$) **and success rate** ($P^0$, $P^m$) **across all difficulty levels** in the **Single-Turn setting (left)** and the **Multi-Turn setting (right)**. Tasks are ordered by increasing difficulty: *Heat 1D* , *Euler 1D* , *NS Transient 2D* . Our methods improvements in reward are largest on easy-to-medium tasks and remain present on hard tasks.

*1D* ). In harder scenarios (medium- to high-precision *Euler 1D* ; *NS Transient 2D* ), reward gains are smaller, but success rate improves steadily. This pattern suggests an intrinsic optimization behavior: for unfamiliar questions, *CAED-Agent* first optimizes correctness, and then optimizes efficiency.

## 4.5 Ablations

We present ablation studies for the two main components of *CAED-Agent*: the surrogate neural network and the LLM agent. All ablation studies are preformed on the same set of problems, *Euler 1D* with medium precision level, with base model OpenAI o4-mini (OpenAI et al., 2024).

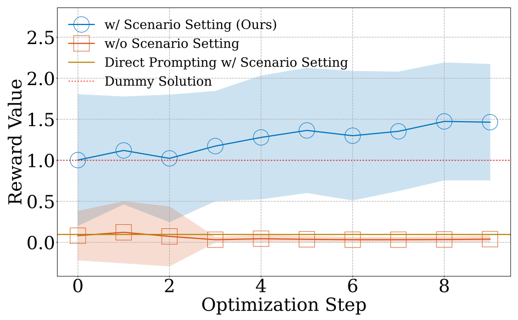


Figure 5: Mean Reward over optimization Steps for *CAED-Agent*, with or without scenario setting description in prompt.

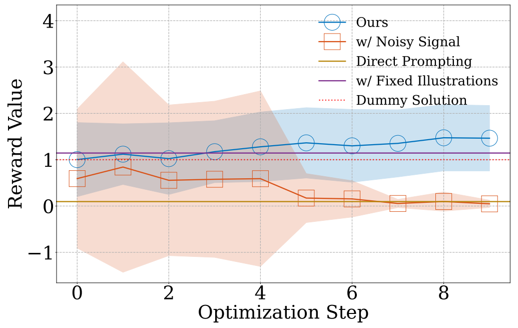


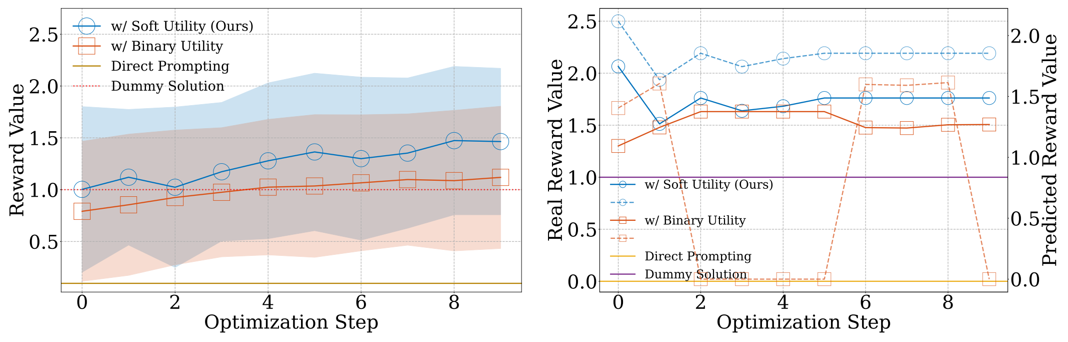
Figure 6: Mean Reward over optimization Steps for *CAED-Agent*, either with surrogate signal, noisy signal or fixed illustrations.

**Physics prior knowledge is necessary to achieve in-context optimization in our tasks.** Figure 5 presents an ablation study on whether the scenario setting is included in the LLM's prompt. We argue that the merits of utilizing LLM in our framework lie in both their in-context optimization abilities and their prior domain knowledge. For the alternative setting (orange lines in 5), we only prompt the model to solve the problem as a numerical optimization problem; see the prompts in E. Figure 5 shows that *CAED-Agent* (blue lines), with physics prior knowledge, can consistently improve reward to surpass baselines, whereas the trajectory without scenario description fails to achieve improvements and converges to a low-reward local optimum. This behavior is also visible in a case study illustrated in 11a.

**Feedback signals are important for agent optimization in our tasks.** We study the effects of our surrogate signal network and present the results in figure 6. We compare *CAED-Agent*with (1) in-context optimization with a fixed set of ground-truth examples for all problems, and (2) our agent equipped with noisy signal from a poorly fitted surrogate model. We experiment on both Single-Turn and Multi-Turn settings in *Euler 1D* 's medium precision level with base model GPT-4o-mini. As shown in Figure 6 and 11b, in both the dataset-level pattern and the case study, our method starts from a worse point than that of fixed illustrations', but surpasses it in later optimization steps; the noisy signal fails to guide the model's optimization after the first few steps, highlighting the importance of an effective signal model.

**Soft surrogate signals significantly improve optimization performance compared to binary surrogate signals.** We verify the effectiveness of the soft utility (Definition 3.3). Specifically, we compare Single-Turn results of our framework under two variants: (a) integrating surrogates with the original binary utility function, and (b) our approach that uses a soft utility function in the surrogate signals. As shown in Figure 7, the soft-utility variant achieves significantly better performance at the dataset level and exhibits a steadier upward trend in the case study.

We also present a case study in 7b, which plots the predicted reward **(dashed lines)** of the step-wise optimal design for both methods besides the real reward in **solid lines**. As shown by the orange dashed line in 7b, once the model receives a zero-utility signal from the surrogate at step 3, it stops refining and remains at a local optimum. By contrast, the blue line shows that although the model proposes the same point at step 3, the non-zero soft-utility signal it receives enables it to continue refining the solution.



(a) Mean Reward over Optimization Steps for *CAED-Agent*, using different functions for surrogate signal.

(b) Case study. An exemplar optimization trajectory in Single-Turn setting. Notations explained in 4.5.

Figure 7: **Study on soft utility functions vs. binary signals for surrogate signal** for o4-mini.

Complete ablation study results are presented in Table 3 of Appendix F. We show that each component of *CAED-Agent*, including the physics prior, the signal NN, and the prompt design all contribute to the final performance. *CAED-Agent* achieves the Pareto optima of success rate and efficiency for all settings, as shown in Figure 1.

## 5 Conclusion

We presented the **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign Agent, a LLM Agent framework for experimental design that focuses on cost-efficiency. Through experiments on three physics simulator environments, each with varying environmental setting and precision requirements, we demonstrated that *CAED-Agent* consistently outperforms both classical Bayesian optimization baselines and state-of-the-art LLM-based optimizers. Our results highlight its ability to achieve high success rates and favorable cost-efficiency trade-offs, even when direct evaluations are prohibitively expensive. Our method introduces the novel contribution of utilizing a low-dimensional cost-efficiency signal neural network, which through our ablation studies we show significantly improves utility of both single-turn and multi-turn experiment design. These findings suggest that *CAED-Agent* provides a practical and scalable path toward deploying agentic frameworks in experiment design in scientific discovery pipelines.

Our approach has the following limitations for exploration in future work. The accuracy of *CAED-Agent* depends on the fidelity of the surrogate, which may under-fit in highly complex or noisy experimental landscapes, and requires some degree of human tuning. Moreover, our data sampling strategy does not guarantee the minimization of sampling size while the model converges. Future work can aim to address these limitations by exploring richer surrogate models, adaptive sampling strategies, and tighter coupling between surrogate predictions and target function evaluation to improve the quality of feedback to LLM. Extending *CAED-Agent* to multi-objective, higher-dimensional, or real-world experimental systems will further test its scalability and practical utility, paving the way toward more autonomous and cost-efficient experimental design agents.

## REPRODUCIBILITY

We evaluate this work on three physics-solver environments that we implemented: *Heat 1D* , *Euler 1D* , and *NS Transient 2D* which include solvers, reference solutions, problem sets, and evaluation pipelines. We plan to extend and organize these into a benchmark to aid the open-source community in solving **C**ost-**A**ware Simulation-Based **E**xperimental **D**esign (*CAED*) better. As the benchmark is still in progress, our solvers, evaluation pipeline, etc. may not yet be robust enough for convenient reproduction. Therefore, we consider it appropriate to open-source the code for this work after acceptance, including not only a (subset) of the aforementioned benchmark but also the neural-network training, the main framework, and the plotting components.

## ETHICS STATEMENT

This work studies cost-aware experimental design agents for physics simulations (e.g., 1D Heat Conduction and Euler equations) and does not involve human subjects, personal data, or sensitive attributes. All data are synthetic or standard simulation benchmarks; no personally identifiable information is used or created. We comply with licenses and usage terms for third-party software and models; any proprietary APIs were accessed under their respective terms.

Potential risks are limited. As our method can improve search efficiency, there is a generic risk of misuse to optimize unsafe physical systems. To mitigate this, we focus on pedagogical and widely used benchmark scenarios with explicit constraints and provide documentation intended for scientific replication rather than domain-specific exploitation.

Fairness and demographic bias considerations are not applicable to our setting. The environmental impact is modest: we train lightweight surrogates on small datasets and use limited inference budgets; we report hardware and runtime details to enable carbon accounting. For reproducibility, we will release code, configurations, and seeds, and follow standard reporting checklists. We declare no conflicts of interest and no concurrent submissions related to this work.

## THE USE OF LARGE LANGUAGE MODELS

In this work, Large Language Models are primarily used for assisting in polishing the mathematical formulation in 3.3, explaining the results in 4.5 and generating the plotting code for Figure 1, 3, 4, 5, 6 and 7.

They are also used for polishing text in some sections. They were NOT used in research ideation and/or writing.

## References

X. Huan, J. Jagalur, and Y. Marzouk. Optimal experimental design: Formulations and computations. *Acta Numerica*, 33:715840, July 2024. ISSN 1474-0508. doi: 10.1017/s0962492924000023. URL `http://dx.doi.org/10.1017/S0962492924000023`.

J. D. Anderson, J. Wendt, and . others. *Computational fluid dynamics*, volume 206. Springer, 1995.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Y. Yao, F. Liu, J. Cheng, and Q. Zhang. Evolve cost-aware acquisition functions using large language models. In *International Conference on Parallel Problem Solving from Nature*, pages 374–390. Springer, 2024.

Y. S. Perera, D. Ratnaweera, C. H. Dasanayaka, and C. Abeykoon. The role of artificial intelligence-driven soft sensors in advanced sustainable process industries: A critical review. *Engineering Applications of Artificial Intelligence*, 121:105988, 2023.

I. Char, Y. Chung, W. Neiswanger, K. Kandasamy, A. O. Nelson, M. Boyer, E. Kolemen, and J. Schneider. Offline contextual bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

B. Trabucco, X. Geng, A. Kumar, and S. Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022.

J. C. Fromer and C. W. Coley. An algorithmic framework for synthetic cost-aware decision making in molecular design. *Nature Computational Science*, 4(6):440–450, 2024.

A. Bharti, D. Huang, S. Kaski, and F. Briol. Cost-aware simulation-based inference. *arXiv preprint arXiv:2410.07930*, 2024.

. OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, . Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, . Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. Avila Belbute Peres, M. Petrov, H. P. Oliveira Pinto, . Michael, . Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle,

N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, . Guangyi, . Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng,

O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, . Yu, . Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

. DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W. L. Xiao, W. Zeng, W. Zhao, W. An, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X. Shen, X. Chen, X. Zhang, X. Chen, X. Nie, X. Sun, X. Wang, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yu, X. Song, X. Shan, X. Zhou, X. Yang, X. Li, X. Su, X. Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Y. Zhang, Y. Xu, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Yu, Y. Zheng, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Tang, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Wu, Y. Ou, Y. Zhu, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Zha, Y. Xiong, Y. Ma, Y. Yan, Y. Luo, Y. You, Y. Liu, Y. Zhou, Z. F. Wu, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Huang, Z. Zhang, Z. Xie, Z. Zhang, Z. Hao, Z. Gou, Z. Ma, Z. Yan, Z. Shao, Z. Xu, Z. Wu, Z. Zhang, Z. Li, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Gao, and Z. Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

S. Ren, P. Jian, Z. Ren, C. Leng, C. Xie, and J. Zhang. Towards scientific intelligence: A survey of llm-based scientific agents, 2025. URL https://arxiv.org/abs/2503.24047.

M. Zhong, C. An, W. Chen, J. Han, and P. He. Seeking neural nuggets: Knowledge transfer in large language models from a parametric perspective, 2024. URL https://arxiv.org/abs/2310.11451.

Q. Lv, T. Liu, and H. Wang. Exploiting edited large language models as general scientific optimizers. *arXiv preprint arXiv:2503.09620*, 2025.

Y. Roohani, A. Lee, Q. Huang, J. Vora, Z. Steinhart, K. Huang, A. Marson, P. Liang, and J. Leskovec. Biodiscoveryagent: An ai agent for designing genetic perturbation experiments, 2025. URL https://arxiv.org/abs/2405.17631.

T. Liu, N. Astorga, N. Seedat, and M. Schaar. Large language models to enhance bayesian optimization, 2024. URL https://arxiv.org/abs/2402.03921.

Y. Chen, X. Song, C. Lee, Z. Wang, R. Zhang, D. Dohan, K. Kawakami, G. Kochanski, A. Doucet, M. Ranzato, and . others. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

X. Song, O. Li, C. Lee, B. Yang, D. Peng, S. Perel, and Y. Chen. Omnipred: Language models as universal regressors. *arXiv preprint arXiv:2402.14547*, 2024.

E. Tang, B. Yang, and X. Song. Understanding llm embeddings for regression, 2025. URL https://arxiv.org/abs/2411.14708.

D. Wu, J. Wang, Y. Meng, Y. Zhang, L. Sun, and Z. Wang. Catp-llm: Empowering large language models for cost-aware tool planning. *arXiv preprint arXiv:2411.16313*, 2024.

B. Lyu, Y. Cao, D. Watson-Parris, L. Bergen, T. Berg-Kirkpatrick, and R. Yu. Adapting while learning: Grounding llms for scientific problems with intelligent tool usage adaptation. *arXiv preprint arXiv:2411.00412*, 2024.

C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

F. Nogueira. Bayesian optimization: Open source constrained global optimization tool for python. `https://github.com/bayesian-optimization/BayesianOptimization`, 2014. Accessed: 2025-08-25.

B. Smucker, M. Krzywinski, and N. Altman. Optimal experimental design. *Nat. Methods*, 15(8): 559–560, 2018.

M. D. Knudsen, L. Georges, K. S. Skeie, and S. Petersen. Experimental test of a black-box economic model predictive control for residential space heating. *Applied energy*, 298:117227, 2021.

P. Pandita. *Bayesian Optimal Design of Experiments for Expensive Black-Box Functions Under Uncertainty*. PhD thesis, Purdue University, 2019.

G. Daras, H. Chung, C. Lai, Y. Mitsufuji, J. C. Ye, P. Milanfar, A. G. Dimakis, and M. Delbracio. A survey on diffusion models for inverse problems. *arXiv preprint arXiv:2410.00083*, 2024.

H. Zheng, W. Chu, B. Zhang, Z. Wu, A. Wang, B. T. Feng, C. Zou, Y. Sun, N. Kovachki, Z. E. Ross, and . others. Inversebench: Benchmarking plug-and-play diffusion priors for inverse problems in physical sciences. *arXiv preprint arXiv:2503.11043*, 2025a.

B. Wang, G. Duan, W. Lv, Y. Tao, H. Xiong, D. Zhang, G. Yang, and F. Shu. Design and experimental realization of triple-band electromagnetically induced transparency terahertz metamaterials employing two big-bright modes for sensing applications. *Nanoscale*, 15(45):18435–18446, 2023.

C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

T. Zheng, Z. Deng, H. T. Tsang, W. Wang, J. Bai, Z. Wang, and Y. Song. From automation to autonomy: A survey on large language models in scientific discovery. *arXiv preprint arXiv:2505.13259*, 2025b.

D. Wang, Y. Wang, X. Jiang, Y. Zhang, Y. Pang, and M. Zhang. When large language models meet optical networks: paving the way for automation. *Electronics*, 13(13):2529, 2024.

A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

K. M. Jablonka, P. Schwaller, A. Ortega-Guerrero, and B. Smit. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, 6(2):161–169, 2024.

J. S. Chan, N. Chowdhury, O. Jaffe, J. Aung, D. Sherburn, E. Mays, G. Starace, K. Liu, L. Maksin, T. Patwardhan, L. Weng, and A. Mdry. Mle-bench: Evaluating machine learning agents on machine learning engineering, 2025. URL `https://arxiv.org/abs/2410.07095`.

H. Hao, X. Zhang, and A. Zhou. Large language models as surrogate models in evolutionary algorithms: A preliminary study. *Swarm and Evolutionary Computation*, 91:101741, 2024.

P. Ghafariasl, A. Mahmoudan, M. Mohammadi, A. Nazarparvar, S. Hoseinzadeh, M. Fathali, S. Chang, M. Zeinalnezhad, and D. A. Garcia. Neural network-based surrogate modeling and optimization of a multigeneration system. *Applied Energy*, 364:123130, 2024.

D. Hou and R. Evins. A protocol for developing and evaluating neural network-based surrogate models and its application to building energy prediction. *Renewable and Sustainable Energy Reviews*, 193:114283, 2024.

C. E. Rasmussen. *Gaussian Processes in Machine Learning*, pages 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-28650-9. doi: 10.1007/978-3-540-28650-9_4. URL `https://doi.org/10.1007/978-3-540-28650-9_4`.

J. Berk, S. Gupta, S. Rana, and S. Venkatesh. Randomised gaussian process upper confidence bound for bayesian optimisation, 2020. URL `https://arxiv.org/abs/2006.04296`.

## A Experimental Environment

**Heat Transfer 1D. (*Heat 1D* )** This solver addresses the 1D heat conduction equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

using explicit finite difference methods with natural convection boundary conditions at $x = 0$ and adiabatic conditions at $x = L$. The tunable parameters include the spatial resolution (`n_space`) and the CFL number (`cfl`) that determines the simulation time step by:

$$\Delta t = \texttt{cfl} \times \frac{(\Delta x)^2}{2\alpha},$$

where $\alpha$ is the thermal diffusivity. The computational cost follows the relationship $C = \texttt{n\_space} \times \texttt{n\_t}$, where `n_t` is the number of time steps accumulated in the solver. The metric for convergence is the RMSE of the heat flux at the convection boundary at the final time step. This simulation has 25 different profiles with varying initial uniform temperatures and physical properties, generating 148 tasks in total, counting both Single-Turn and Multi-Turn settings.

**Euler 1D. (*Euler 1D* )** This solver implements the 1D Euler equations for compressible flow:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0$$

using the MUSCL-Roe method with superbee limiter for high-resolution shock capturing. The tunable parameters include the CFL number (`cfl`) that determines the simulation time step by:

$$\Delta t = \texttt{cfl} \times \frac{\Delta x}{|\lambda|_{\max}},$$

where $|\lambda|_{\max}$ is the maximum eigenvalue of the flux Jacobian, the spatial resolution (`n_space`), the limiter parameter `beta` for generalized minmod flux limiter, and the blending parameter `k` between 0-th and 1-st order interpolation scheme. The computational cost follows the relationship $C = \texttt{n\_space} \times \texttt{n\_t}$, where `n_t` is the number of time steps accumulated in the solver. Convergence is evaluated through multiple criteria: RMSE of the solution fields, positivity preservation of density and pressure, and shock consistency validation. The dataset encompasses 3 classical benchmark profiles (Sod shock tube, Lax problem, and Mach 3), generating a total of 134 tasks, counting both Single-Turn and Multi-Turn settings.

**Transient Navier-Stokes 2D. (*NS Transient 2D* )** This solver implements the 2D transient incompressible Navier-Stokes equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

where $u, v$ are velocity components, $p$ is pressure, and $Re$ is the Reynolds number. The tunable parameters include the spatial resolution (`resolution`) that determines the computational grid size, the CFL number (`cfl`) controlling time step stability through $\Delta t = \texttt{cfl} \times \Delta x$, the relaxation factor (`relaxation_factor`) for pressure correction convergence, and the residual threshold (`residual_threshold`) for pressure solver convergence. The computational cost follows the relationship $C = 2 \times \texttt{resolution}^2 \times (\texttt{num\_steps} + \texttt{total\_pressure\_iterations})$, where the factor of 2 accounts for the fixed aspect ratio domain configuration with $x\_resolution = 2 \times \texttt{resolution}$. Convergence is evaluated through normalized velocity RMSE criteria, with temporal evolution tracked throughout the simulation. The dataset encompasses 18 benchmark profiles across 6 different boundary conditions (simple circular obstacles, complex geometries, random obstacle fields, dual inlet/outlet configurations, dense obstacle arrays, and dragon-shaped obstacles) tested at three Reynolds numbers (Re=1000, 3000, 6000), generating a total of 44 tasks across different precision levels and geometric complexities.

**Dummy Solution Search.** For each task, we find optimal solutions that meet both accuracy requirements and have the lowest cost using brute-force search. Given our parameters have a monotonic relationship between cost and accuracy (i.e., they are spatial resolution), we start with a coarse value and Multi-Turnly refine it with fixed ratios (e.g., halve the time step size, double the spatial resolution) until the distance between adjacent runs is within the accuracy threshold. For single-turn tasks, we set the reference cost to the optimal cost found by brute-force search. For multi-turn tasks, we set the reference cost to the accumulated cost incurred during the brute-force search.

## B Algorithmic Diagram

**Algorithm 1** $Solve$, Single-Turn *CAED-Agent* Framework

```
1: Input: Forward experimental process $\mathcal{F}$, design space $\mathcal{X}$, environment parameters $\theta$, neural
   surrogate $\mathcal{S}$, number of iteration $N$, history context length $K$, initial sample size $m$.
2: Initialize LLM design-value history as a priority queue $\mathcal{M}$
3: Push to $\mathcal{M}$ uniformly sampled initial design-value pairs $\{(x_j, \Phi_j^{pred}, \hat{\mathbf{C}}_j^{pred})\}_{j=1}^m$, evaluated by
   $\mathcal{S}$
4: repeat
5:     LLM proposes candidate designs $\{x_i\}_{i=1}^k$
6:     Evaluate candidates with neural surrogate: $(\Phi_i^{pred}, \hat{\mathbf{C}}_i^{pred}) \leftarrow \mathcal{S}(x_i, \theta)$ for $i = 1, \ldots, k$
7:     Push $\{(x_i, \Phi_i^{pred}, \hat{\mathbf{C}}_i^{pred})\}$ to $\mathcal{M}$, keeping only top-$K$ samples.
8: until Number of iterations $N$ reached
9: Output: $x^* = \arg\max_{x_i} \frac{\Phi(\mathcal{S}(x_i,\theta),\theta)}{\mathbf{C}'(x_i,\theta)}$ from design-value history.
```

**Algorithm 2** Multi-Turn *CAED-Agent* Framework

```
1: Input: Forward experimental process $\mathcal{F}$, design space $\mathcal{X}$, environment parameters $\theta$, neural
   surrogate $\mathcal{S}$, number of iteration for Single-Turn solution $N$, history context length $K$, initial
   sample size $m$, maximum allowed number of ground-truth evaluation $T$.
2: Obtain Single-Turn solution $x = Solve(\mathcal{F}, \mathcal{X}, \theta, \mathcal{S}, \mathcal{N}, K, m)$
3: Initialize solution sequence as a queue $\mathcal{A} = \{x_0\}$
4: Initialize LLM ground-truth design-value history as a priority queue $\mathcal{M}$
5: Evaluate with ground-truth simulator $(\Phi_0^{gt}, \hat{\mathbf{C}}_0^{gt}) \leftarrow \mathcal{F}(\S_\prime, \theta)$
6: Push $(x_0, \Phi_0^{gt}, \hat{\mathbf{C}}_0^{gt})$ to $\mathcal{M}$
7: repeat
8:     LLM agent proposes candidate designs $\{x_i\}_{i=1}^k$
9:     Evaluate candidates with neural surrogate: $(\Phi_i^{pred}, \hat{\mathbf{C}}_i^{pred}) \leftarrow \mathcal{S}(x_i, \theta)$ for $i = 1, \ldots, k$
10:    Add top surrogate-evaluated pair $(x_i, \Phi_i^{pred}, \hat{\mathbf{C}}_i^{pred})$ to solution sequence $\mathcal{A}$
11:    Evaluate with ground-truth simulator $(\Phi_i^{gt}, \hat{\mathbf{C}}_i^{gt}) \leftarrow \mathcal{F}(x_i, \theta)$
12:    Push $\{(x_i, \Phi_i^{gt}, \hat{\mathbf{C}}_i^{gt})$ to $\mathcal{M}$, keeping only top-K samples
13: until LLM outputs $should\_stop = True$ or number of iterations reaches $T$
14: Outputs $\mathcal{A}$
```

## C Neural Network Training

We train one neural-network for each problem (*Heat 1D* , *Euler 1D* , *NS Transient 2D* )'s all precision levels; each network's input and output dimension are as described in 3.2.

We uniformly sample design and environmental parameters on coarse grids. We specifically include environmental parameters to enable interpolation across conditions while avoiding training and tracking multiple network instances for different environment combinations. We provide the range of inputs (environmental parameters and tunable parameters) as follows, from which we performed uniform sampling, and statistics of sampled targets in Table 1. We stress that while our target dimensions have drastically different ranges and high variance, we perform in-dimension normalization as shown in Figure 9, therefore achieving satisfactory training results shown in Figure 8.

```
Heat 1D:
  Environmental Parameters:
    L: [0.1, 0.3]  # Wall thickness [m] - uniform random in range
    k: [0.5, 1.0]  # Thermal conductivity [W/m-K] - uniform random in
    ↪  range
    h: [0.1, 10000]  # Convection coefficient [W/mš-K] - log-uniform
    ↪  random in range
    rho: [1000, 2000]  # Density [kg/mş] - uniform random in range
    cp: [800, 1000]  # Specific heat [J/kg-K] - uniform random in range
    T_inf: [-40, 40]  # Ambient temperature [řC] - uniform random in
    ↪  range
    T_init: [0, 30]  # Initial temperature [řC] - uniform random in
    ↪  range
    record_dt: 10.0  # Time interval between recordings [s] - fixed
    end_frame: 24  # Simulation end frames - fixed

  Tunable Parameters:
    n_space: [64, 2048]  # Number of spatial points (iterative search:
    ↪  initial=64, factor=2, max_iter=6)

Euler 1D:
  Environmental Parameters:
    L: 1.0  # Domain length - fixed
    gamma: 1.4  # Ratio of specific heats - fixed
    case: {"sod", "lax", "mach_3"}  # Initial condition name - 3
    ↪  discrete values across profiles
    record_dt: {0.02, 0.012, 0.009}  # Time interval between recordings
    ↪  - specific values per case
    end_frame: 10  # Simulation end frames - fixed

  Tunable Parameters:
    n_space: [256, 4096]  # Number of grid cells (iterative search:
    ↪  initial=256, factor=2, max_iter=7)

NS Transient 2D:
  Environmental Parameters:
    boundary_condition: {1, 2, 3, 4, 5, 6}  # 6 boundary condition types
    ↪  across 18 profiles
    reynolds_num: {1000, 3000, 6000}  # Reynolds number - 3 discrete
    ↪  values
    vorticity_confinement: 0.0  # Fixed across profiles
    total_runtime: 1.0  # Fixed across profiles - fixed
    no_dye: False  # Fixed across profiles
    cpu: False  # Fixed across profiles
    visualization: 0  # Fixed across profiles
    advection_scheme: "cip"  # Fixed across profiles

  Tunable Parameters:
    resolution: [50, 400]  # Grid resolution (iterative search:
    ↪  initial=50, factor=2, max_iter=4)
```

Table 1: Dataset Statistics.

| | *RMSE Loss* | *Cost* | *N.* samples |
|---|---|---|---|
| *Heat 1D* | $4.47e^{-4} \pm 9.50e^{-4}$ | $8.33e^{7} \pm 1.27e^{8}$ | 4440 |
| *Euler 1D* | $3.48e{-}2 \pm 3.60e^{-2}$ | $2.76e^{6} \pm 2.42e^{6}$ | 4020 |
| *NS Transient 2D* | $2.55e^{-1} \pm 1.90e^{-1}$ | $2.11e^{8} \pm 1.94e^{8}$ | 1320 |

For all problems, we train neural-network with the same structure as shown in 9; to achieve optimal results for individual problems, we compare the training results with three sets of structures for each problem and choose the one with the best test loss. Specifically, we experiment with the combinations of :

```
h: {2, 3, 4, 6}
d: {64, 128, 256}
```

Where $h, d$ follow the notation in 9, and the hyper-parameters we used are shown as follows:

```
activation_mod: ReLU
layer_norm: False
res_connection: False

batch: 16
epochs: 40
steps_per_epoch: 200

peak_lr: 1e-3
weight_decay: 1e-4
warmup_steps: 100
decay_steps: 1000
gnorm_clip: 1.0
accumulation_steps: 100
```

We show the results of our best checkpoints for the three problems in 8.

## D Soft Utility Function

*Proof of Proposition 3.5.* Let $b(y, \theta) := \mathbf{1}\{\Phi(y, \theta) = 1\}$ be the binary utility and let $s_f(y, \theta) := f(y, \theta)$ be any soft utility satisfying Definition 3.3. By *normalization* (Def. 3.3(ii)), $f(y, \theta) \in [0, 1]$, and by *feasibility calibration* (Def. 3.3(i)), $f(y, \theta) = 1$ iff $y \in \mathcal{G}_\theta = \{y : \Phi(y, \theta) = 1\}$ and $\sup_{y \notin \mathcal{G}_\theta} f(y, \theta) < 1$. Hence the postprocessing map

$$\tau : [0, 1] \to \{0, 1\}, \qquad \tau(u) := \mathbf{1}\{u = 1\}$$

is well-defined (by normalization) and satisfies $b(y, \theta) = \tau(f(y, \theta))$ for all $(y, \theta)$ (by feasibility calibration). Thus the binary signal is a deterministic garbling of the soft signal.

Fix a base model and budget $T \geq 1$, and write the histories $h^{\text{bin}}_{t-1} = \{(x_s, y_s, b(y_s, \theta))\}_{s=1}^{t-1}$ and $h^{f}_{t-1} = \{(x_s, y_s, f(y_s, \theta))\}_{s=1}^{t-1}$; then $h^{\text{bin}}_{t-1} = \tau(h^{f}_{t-1})$ coordinate-wise. Given any binary-utility policy $\pi_{\text{bin}}$, define a soft-signal policy $\tilde{\pi}_f$ that *simulates* it via

$$\tilde{\pi}_f(\,\cdot \mid \theta, h^{f}_{t-1}) \;:=\; \pi_{\text{bin}}(\,\cdot \mid \theta, \tau(h^{f}_{t-1})).$$

Under identical environment randomness, $\tilde{\pi}_f$ induces the same trajectory distribution as $\pi_{\text{bin}}$, hence

$$\mathbb{E}_{x_T \sim \tilde{\pi}_f(\cdot|\theta)}\big[R^0(x_T, \theta)\big] \;=\; \mathbb{E}_{x_T \sim \pi_{\text{bin}}(\cdot|\theta)}\big[R^0(x_T, \theta)\big] \quad \text{for all } \theta.$$

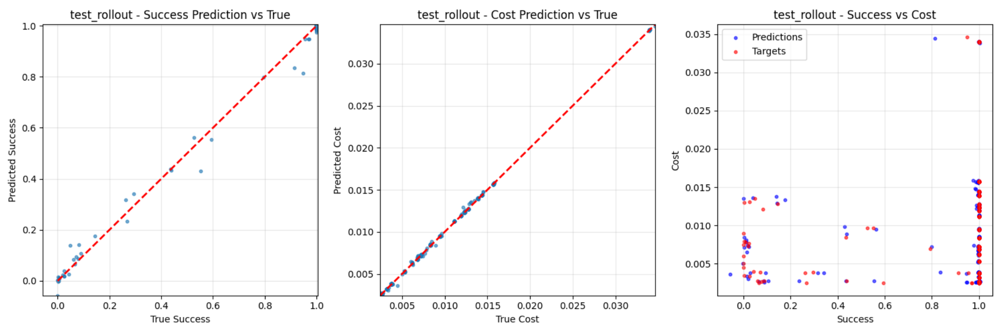
Taking expectation over the task distribution yields equality in expectation.

By *monotone alignment* (Def. 3.3(iii)), if $\Phi(y_1, \theta) \preceq \Phi(y_2, \theta)$ then $f(y_1, \theta) \leq f(y_2, \theta)$; hence ranking by $f$ is orderpreserving with respect to $\Phi$. Since $R^0(x, \theta)$ (Eq. (4)) is nondecreasing in $\Phi$ (its numerator) and $f = 1$ iff $\Phi = 1$ (by feasibility calibration), using $f$ to refine decisions cannot decrease the expected reward relative to $\tilde{\pi}_f$, and is strictly better whenever such refinements occur with positive probability.

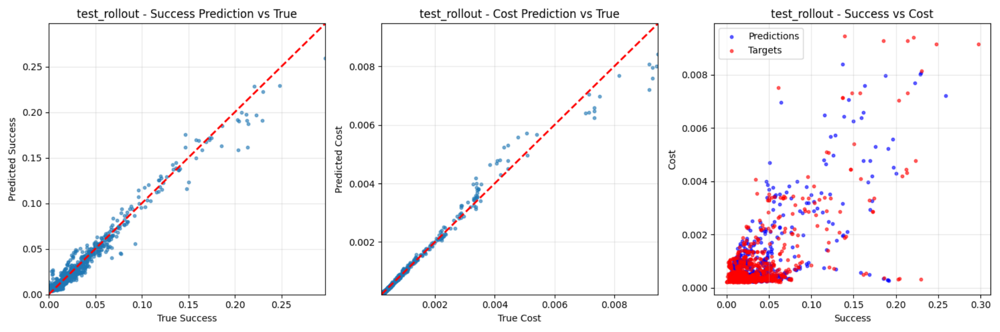
Now let $\pi_f$ denote any soft-signal policy produced by our framework. Since $\pi_f$ can always ignore the extra information and implement $\tilde{\pi}_f$, we have

$$\mathbb{E}_\theta\, \mathbb{E}_{x_T \sim \pi_f(\cdot|\theta)}\big[R^0(x_T, \theta)\big] \;\geq\; \mathbb{E}_\theta\, \mathbb{E}_{x_T \sim \tilde{\pi}_f(\cdot|\theta)}\big[R^0(x_T, \theta)\big]. \tag{$*$}$$

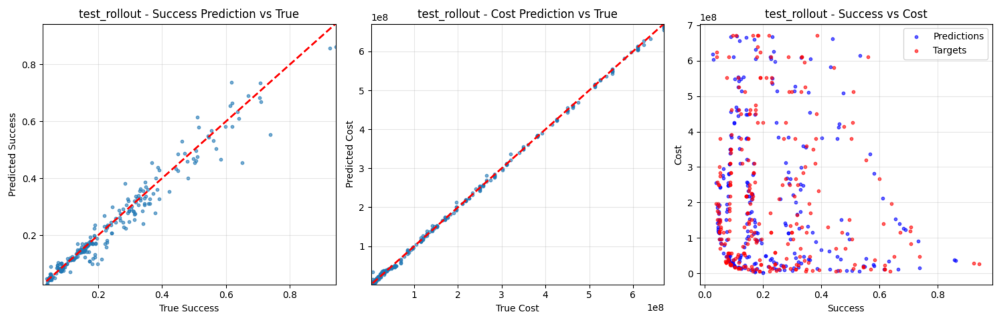
The case $T = 1$ (zero-shot) follows verbatim by replacing $x_T$ with the single-step $x$. □



(a) *Heat 1D*



(b) *Euler 1D*



(c) *NS Transient 2D*

Figure 8: Test results of our best neural network for each task. The plots from left to right respectively mean: (left) soft utility signal of true RMSE loss *vs.* soft utility signal of predicted RMSE loss, (middle) true cost *vs.* predicted cost, (right) distribution in the cost-utility space of predicted *vs.* true points.

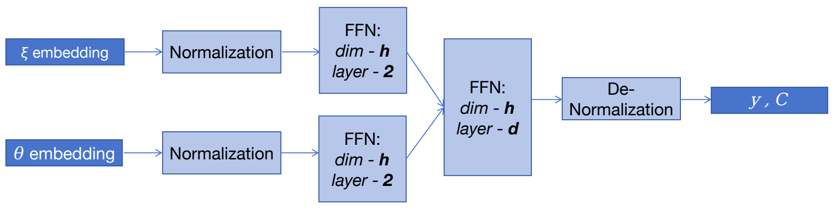


Figure 9: Neural-Network Structure

In this work, we define the soft utility function $f(r)$ as follows:

$$f(r) = \begin{cases} 1.0 & \text{if } d \leq \epsilon \\ \alpha e^{-\beta(r-1)^\gamma} + (1-\alpha)\left(\frac{1}{1+\omega(r-1)^\delta}\right) & \text{if } d > \epsilon, \end{cases} \tag{10}$$

where $r = \frac{d}{\epsilon}$. The parameters are set to $\alpha = 0.6$, $\beta = 0.43$, $\gamma = 1.5$, $\omega = 0.3$, and $\delta = 2.2$.

This function is designed so that the utility value drops to approximately 0.5 when the distance $d$ is double the tolerance $\epsilon$ (i.e., $r = 2$), and it decays rapidly towards zero as the distance increases further, becoming negligible for distances approaching $10\epsilon$ (i.e., $r = 10$). A plot of $f(r)$ is shown in Figure 10.

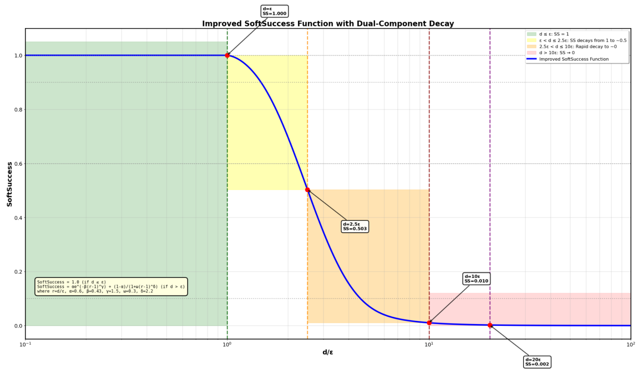


Figure 10: Plot of the soft utility function $f(r)$. The function maintains a maximum utility of 1.0 for normalized distances $r \leq 1$, drops to approximately 0.5 at $r = 2$, and rapidly decays towards zero for larger values of $r$.

## E PROMPTS USED IN AGENT FRAMEWORK

**Prompt Example for *Euler 1D* Single-Turn w/. Scenario Setting**

**Instruction**

Your task is to find the optimal parameter, solving the 1D Euler equations for compressible inviscid flow, using a 2nd order MUSCL scheme with Roe flux and generalized superbee limiter. This serves as a simplified model for compressible fluid dynamics. You should try to minimize the total cost incurred by function calls, but your primary goal is to successfully meet the convergence criteria. You should always use the tool call function to finish the problem.

Workflow: n_space (Number of grid cells) determines the spatial discretization resolution: $\Delta x = L/n\_space$ where L is the domain length. You may **only** change ‘n_space‘. The value of k is **-1.0**, beta is **1.0**, cfl is **0.25**. **You must not change them!** You have only one opportunity to choose an optimal value for n_space. No trial-and-error or iterative optimization is permitted. Your goal is to select a value that provides adequate spatial resolution while keeping computational cost reasonable.

Step 1: Make your best **one-shot** guess for n_space.
Step 2: Call the Convergence Test Function and check if converged.
Step 3: Output final answer with no further tool calls.

**Input**

QID: 1
Problem: Euler 1D Equations with 2nd Order MUSCL-Roe Method
This simulation solves the 1D Euler equations for compressible inviscid flow, using a 2nd order MUSCL scheme with Roe flux and generalized superbee limiter:
Conservative form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0$$

Where the conservative variables and flux are:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{pmatrix}$$

Primitive variables:

- $\rho$ = density
- $u$ = velocity
- $p$ = pressure
- $E$ = specific total energy

Equation of state:

$$p = (\gamma - 1)\rho \left( E - \frac{u^2}{2} \right)$$

where $\gamma$ is the ratio of specific heats.

Spatial Discretization: The spatial discretization uses MUSCL reconstruction with blending parameter $k$:

$$\mathbf{U}^L_{j+\frac{1}{2}} = \mathbf{U}_j + \frac{1+k}{4}\psi(r_j)(\mathbf{U}_{j+1} - \mathbf{U}_j)$$

$$\mathbf{U}^R_{j+\frac{1}{2}} = \mathbf{U}_{j+1} - \frac{1+k}{4}\psi(r_{j+1})(\mathbf{U}_{j+2} - \mathbf{U}_{j+1})$$

where $k$ is a blending coefficient between central ($k = 1$) and upwind ($k = -1$) scheme, and $\psi(r)$ is the slope limiter function.

Slope Limiting: The slope limiter uses a generalized superbee limiter:

$$\psi(r) = \max\left[0, \max\left[\min(\beta r, 1), \min(r, \beta)\right]\right]$$

where $\beta$ is the limiter parameter controlling dissipation.

The slope ratio $r$ at interface $j$ is defined as:

$$r_j = \frac{\mathbf{U}_{j+1} - \mathbf{U}_j}{\mathbf{U}_{j+2} - \mathbf{U}_{j+1}}$$

This ratio indicates the local non-smoothness, which will be the input into the slope limiter to achieve the TVD condition.

Flux Computation: The interface flux is computed using the Roe approximate Riemann solver:

$$\mathbf{F}_{j+\frac{1}{2}} = \frac{1}{2}\left[\mathbf{F}(\mathbf{U}^L) + \mathbf{F}(\mathbf{U}^R)\right] - \frac{1}{2}|\mathbf{A}|(\mathbf{U}^R - \mathbf{U}^L)$$

where $|\mathbf{A}|$ is the Roe matrix with Roe-averaged quantities.

Initial condition cases:

- sod: Left: $\rho = 1.0, u = 0.0, p = 1.0$; Right: $\rho = 0.125, u = 0.0, p = 0.1$
- lax: Left: $\rho = 0.445, u = 0.6977, p = 3.528$; Right: $\rho = 0.5, u = 0.0, p = 0.571$
- mach_3: Left: $\rho = 3.857, u = 0.92, p = 10.333$; Right: $\rho = 1.0, u = 3.55, p = 1.0$

Parameter Information:

- cfl: Courant-Friedrichs-Lewy number, $CFL = \frac{(|u|+c)\Delta t}{\Delta x}$ where $c = \sqrt{\gamma p/\rho}$ is the speed of sound
- beta: Limiter parameter for generalized superbee
- k: Blending parameter between central and upwind fluxes
- n_space: Number of grid cells for spatial discretization, determines spatial resolution: $\Delta x = L/n\_space$

Physical Parameters:

- Domain length: 1.0

- Gamma (ratio of specific heats): 1.4
- Case: sod

Convergence Check:

- Errors between the simulation based on your solution and the simulation based on the self-refined solution are computed to assess convergence.
- Convergence is confirmed if the following validation criteria are satisfied.

Validation Criteria:

- **Current Problem Precision Level**: HIGH
- **Required RMSE Tolerance**: $\leq 0.01$
- Relative RMSE must meet this tolerance compared to self-refined solution
- Positivity preservation: pressure and density must remain positive at all times
- Shock speed consistency: pressure gradients should not exceed physical bounds

**Available functions:**

Function Name: euler_1d_check_converge_n_space
Description: Conduct a 1D Euler PDE simulation and evaluate its spatial convergence by doubling n_space. It returns the following results:

- `RMSE`: float
- `is_converged`: boolean
- `accumulated_cost`: integer
- `The cost of the solver simulating the environment`: integer
- `The cost of the solver verifying convergence (This will not be included in your accumulated_cost)`: integer
- `metrics1`: object
- `metrics2`: object

Parameters:

- `cfl` (float): CFL number
- `beta` (float): Limiter parameter for generalized superbee
- `k` (float): Blending parameter for MUSCL reconstruction
- `n_space` (integer): Current number of grid cells for spatial discretization

Required parameters: `cfl`, `beta`, `k`, `n_space` **Design-Value History**

```
Below are some previous n_space values and their simulation accuracy
↪  and efficiency indicators. The values are arranged in ascending
↪  order based on accuracy, where higher values indicate a closer
↪  simulation result to ground truth. The efficiency indicator is
↪  also important, where higher values mean a more cost-efficient
↪  n_space choice.

<n_space> 240 </n_space>
 Accuracy Indicator:
0.9834
 Efficiency Indicator:
1.1479

<n_space> 512 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.2717
```

```
<n_space> 400 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.4255

<n_space> 300 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.7290

<n_space> 288 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.7897

<n_space> 260 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.9707

<n_space> 258 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.9865

<n_space> 257 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.9946

<n_space> 256 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
1.0027

<n_space> 252 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
1.0364

Output final answer in the requested format with a new n_space value
↪  that is different from all values above. You should first ensure
↪  an accurate simulation by achieving 1.0 in accuracy indicator,
↪  then gradually increase efficiency by choosing a coarser n_space
↪  value.
```

**Prompt Example for *Euler 1D* Single-Turn w/. Scenario Setting**

**Instruction**

Your task is to optimize a one-dimensional black-box function with a given parameter. You will be prompted with a list of history of parameter and values, where values include an accuracy indicator and success indicator. You are required to first optimize accuracy until it reaches

1.0, then optimize efficiency for as high as possible. The parameter in history will start with <n_space>and end with </n_space>. Please return a parameter value different from all values given in the history that you think will optimize the function value as requested. Please return your answer by starting with <n_space>and ending with </task>as well. **You may NOT use any form of prior knowledge, and treat all parameter names, function names, etc. as purely arbitrary.**

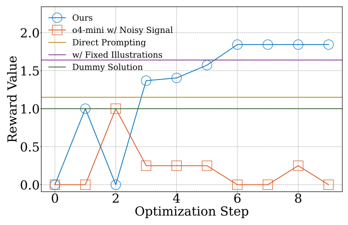
**Input**

**Design-Value History**

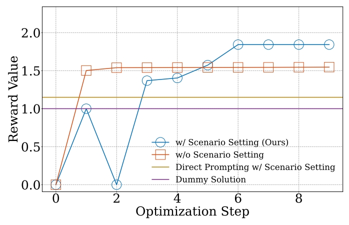
```
Below are some previous n_space values and their simulation accuracy
↪  and efficiency indicators. The values are arranged in ascending
↪  order based on accuracy, where higher values indicate a closer
↪  simulation result to ground truth. The efficiency indicator is
↪  also important, where higher values mean a more cost-efficient
↪  n_space choice.

<n_space> 240 </n_space>
 Accuracy Indicator:
0.9834
 Efficiency Indicator:
1.1479

<n_space> 512 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.2717

<n_space> 400 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.4255

<n_space> 300 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.7290

<n_space> 288 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.7897

<n_space> 260 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.9707

<n_space> 258 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
0.9865

<n_space> 257 </n_space>
 Accuracy Indicator:
```

```
1.0000
 Efficiency Indicator:
0.9946

<n_space> 256 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
1.0027

<n_space> 252 </n_space>
 Accuracy Indicator:
1.0000
 Efficiency Indicator:
1.0364

Output final answer in the requested format with a new n_space value
↪  that is different from all values above. You should first ensure
↪  an accurate simulation by achieving 1.0 in accuracy indicator,
↪  then gradually increase efficiency by choosing a coarser n_space
↪  value.
```

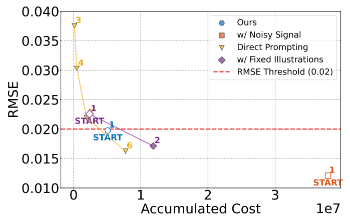
## F Detailed Results



(a) An exemplar optimization trajectory in Single-Turn setting for *CAED-Agent*with vs. without scenario setting.



(b) An exemplar optimization trajectory in Single-Turn setting for *CAED-Agent*with various signals.



(c) An exemplar optimization trajectory in Mingle-Turn setting for *CAED-Agent*with various signals.

Figure 11: **Case studies** for ablations, with base model OpenAI o4-mini.

Full results of the comprehensive benchmark are presented in table 2. The case studies as introduced in 4.5 are shown in Figure 11.

The full ablation results are presented in table 3. Our ablations on surrogate neural network and prior knowledge are conducted on *Euler 1D* 's medium precision level tasks; we report reward $R^0, R^m$ and success rates $P^0, P^m$ for both Single-Turn and Multi-Turn settings. Our base model is fixed as o4-mini. Note that although *CAED-Agent* without Physics prior is achieving a higher mean reward in Single-Turn setting, its success rate is much lower than our method, indicating its frequent choice of coarse designs that leads to high reward in only a few tasks. We argue that this is a form of reward hacking as it contradicts with our expectation to carry out experiments correctly and efficiently.

Table 2: Main evaluation results in both Single-Turn and Multi-Turn settings. Values in each box is the mean of tasks evaluated in three precision levels. Note that we report both reward $R^0$, $R^m$ and for-reference quantities $P^0$ and $P^m$. Values in bold font are the best-achieving ones, and values with ↑ indicate a significant rise compared to direct prompting.

(a) Single-Turn *CAED-Agent*

| | | *Heat 1D* | | *Euler 1D* | | *NS Transient 2D* | |
|---|---|---|---|---|---|---|---|
| Method | Base Model | $R^0$ | $P^0$ | $R^0$ | $P^0$ | $R^0$ | $P^0$ |
| BO (Nogueira, 2014) | – | 0.253 | **1.000** | 0.464 | 0.125 | 0.814 | 0718 |
| Base LLM | Llama3.2-3B-Instruct | 0.288 | 0.347 | 0.698 | 0.174 | 0.052 | 0.151 |
| | Qwen-8B | 0.412 | 0.633 | 0.642 | 0.268 | 0.342 | **1.000** |
| | o4-mini | 0.362 | 0.253 | 0.501 | 0.301 | 0.565 | 0.516 |
| *CAED-Agent* (Ours) | Llama3.2-3B-Instruct | 0.950↑ | 0.773 ↑ | 0.939 ↑ | 0.516 ↑ | 1.591↑ | 0.785↑ |
| | Qwen-8B | 0.853↑ | 0.759 ↑ | 0.897 ↑ | **0.789** ↑ | **1.813**↑ | 0.702 |
| | o4-mini | **1.239** ↑ | 0.679 ↑ | **1.764** ↑ | 0.733 ↑ | 0.842 ↑ | 0.536 |

(b) Multi-Turn *CAED-Agent*

| | | *Heat 1D* | | *Euler 1D* | | *NS Transient 2D* | |
|---|---|---|---|---|---|---|---|
| Method | Base Model | $R^m$ | $P^m$ | $R^m$ | $P^m$ | $R^m$ | $P^m$ |
| BO (Nogueira, 2014) | – | 0.290 | **1.000** | 0.496 | 0.625 | 0.517 | 0.766 |
| Base LLM | Llama3.2-3B-Instruct | 1.060 | 0.837 | 0.328 | 0.531 | 1.232 | 0.448 |
| | Qwen-8B | 1.613 | 0.756 | 1.511 | 0.421 | 0.662 | 0.861 |
| | o4-mini | 1.960 | 0.960 | 1.135 | 0.392 | 0.991 | 0.674 |
| OPRO (Yang et al., 2023) | Llama3.2-3B-Instruct | 0.170 | 0.917 | 0.290 | 0.600 | 0.275 | 0.877 |
| | Qwen-8B | 0.217 | 0.917 | 0.323 | **0.680** | 0.326 | **1.000** |
| | o4-mini | 0.241 | 0.917 | 0.974 | 0.520 | 0.957 | **1.000** |
| *CAED-Agent* (Ours) | Llama3.2-3B-Instruct | 1.204 ↑ | 0.946 ↑ | 1.339 ↑ | 0.572 | 1.435 | 0.925 ↑ |
| | Qwen-8B | 1.760 | 0.900 ↑ | 1.359 | 0.624 ↑ | 1.535 ↑ | 0.944 |
| | o4-mini | **1.981** | 0.986 | **1.571** ↑ | 0.443 | **1.538** ↑ | 0.972 ↑ |

Table 3: Ablation results averaged over all tasks.

| | Single-Turn Setting | | Multi-Turn Setting | |
|---|---|---|---|---|
| Setting | $R^0$ | $P^0$ | $R^m$ | $P^m$ |
| *CAED-Agent* (Ours) | 0.571 | **0.708** | **0.834** | **1** |
| *CAED-Agent* w/ Sparse Surrogate Signal | 0.42 | 0.5 | 0.635 | 0.875 |
| *CAED-Agent* w/ Random Signal | 0.142 | 0.583 | 0.426 | 0.583 |
| *CAED-Agent* w/ In-Context Signal | 0.42 | 0.5 | 0.572 | **0.958** |
| *CAED-Agent* w/o Physics Prior | **0.595** | 0.152 | 0.475 | 0.375 |
| Direct Prompting | 0.096 | 0.125 | 0.116 | 0.167 |