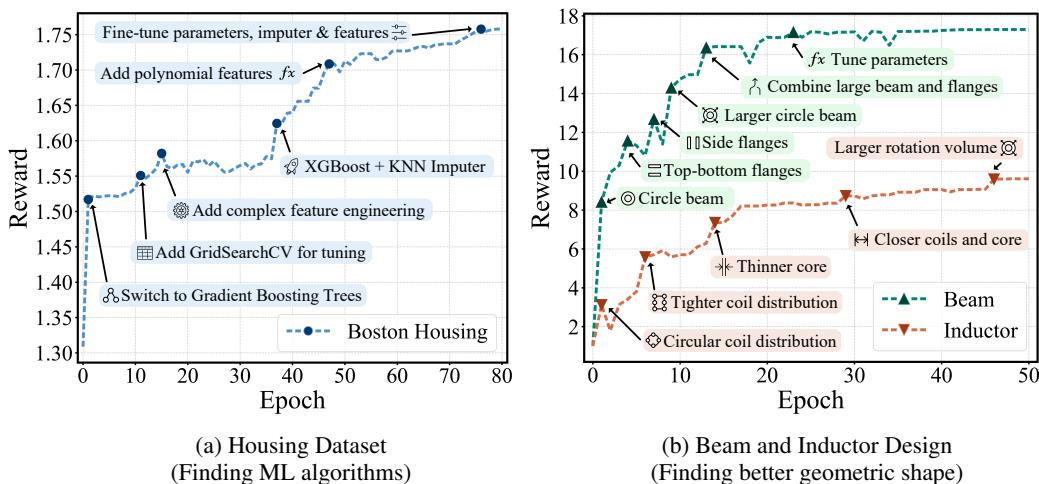


# 000 001 002 003 004 005 HELIX: EVOLUTIONARY REINFORCEMENT LEARNING 006 FOR OPEN-ENDED SCIENTIFIC PROBLEM SOLVING 007 008 009

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

## 012 ABSTRACT 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027

028 Large language models (LLMs) with reasoning abilities have demonstrated growing  
029 promise for tackling complex scientific problems. Yet such tasks are inherently  
030 domain-specific, unbounded and open-ended, demanding exploration across vast  
031 and flexible solution spaces. Existing approaches, whether purely learning-based  
032 or reliant on carefully designed workflows, often suffer from limited exploration  
033 efficiency and poor generalization. To overcome these challenges, we present **HE-  
034 LIX**—a Hierarchical Evolutionary reinforcement Learning framework with In-  
035 context eXperiences. HELIX introduces two key novelties: (i) a diverse yet high-  
036 quality pool of candidate solutions that broadens exploration through in-context  
037 learning, and (ii) reinforcement learning for iterative policy refinement that pro-  
038 gressively elevates solution quality. This synergy enables the discovery of more  
039 advanced solutions. On the circle packing task, HELIX achieves a new state-of-  
040 the-art with a sum of radii of 2.635983 using only a 14B model. Across standard  
041 machine learning benchmarks, HELIX further surpasses GPT-4o with a carefully  
042 engineered pipeline, delivering an average F1 improvement of 5.95 points on the  
043 Adult and Bank Marketing datasets and a 40.5% reduction in RMSE on Boston  
044 Housing.



045 Figure 1: The figure demonstrates how our framework progressively discovers new insights  
046 and refines solutions over iterations. **(a):** Reward curve for the housing dataset optimization, where  
047 improvements are achieved through iterative adoption of better models, parameter tuning, and  
048 feature engineering, with the final reward of 1.758 corresponding to an RMSE of 1.747. **(b):** Reward  
049 curves for the beam and inductor design tasks, where the algorithm explores novel geometries and  
050 combines favorable structural features to enhance performance.

054  
055 

## 1 INTRODUCTION

  
056  
057058 Solving complex scientific problems with large language models (LLMs) is an important and in-  
059 creasingly active research direction (Forootani, 2025). By leveraging and enhancing their reasoning  
060 capabilities, LLMs have demonstrated promising results in tackling challenging scientific tasks,  
061 such as symbolic regression (Shojaee et al., 2024), molecular generation (Liu et al., 2024), and diffi-  
062 cult mathematical optimization problems (Ahmed & Choudhury, 2024). Addressing such problems  
063 holds the potential to advance the boundaries of human knowledge and reshape scientific discovery.064 While LLMs have shown promising applications, complex scientific problems remain particularly  
065 challenging due to three intrinsic characteristics. First, they are *domain-specific*, with unique envi-  
066 ronments and problem-specific constraints that differ across various tasks. Second, they are *open-  
067 ended*, requiring exploration of vast and flexible solution spaces. Third, they are *unbounded*, often  
068 with no known or guaranteed global optimum.069 To address these challenges, we argue that a powerful LLM for solving complex scientific prob-  
070 lems must possess three corresponding key abilities: **(1) learning from experience**, i.e., it should  
071 enable task-specific policy adaptation by incorporating feedback from previous trials, addressing  
072 the *domain-specific* nature of each problem. **(2) Balancing quality and diversity**, i.e., it should  
073 maintain a diverse population to thoroughly explore the vast and flexible solution spaces inherent in  
074 *open-ended* tasks. **(3) Exploration based on the shoulder of giants**, i.e., it should iteratively build  
075 upon existing high-quality solutions to extend the known limits of *unbounded* problems.076 However, recent works largely lack the capabilities outlined above, limiting their effectiveness on  
077 complex scientific problems. Existing approaches fall into two categories. *Post-training methods*  
078 (e.g., SFT, RLVR) fine-tune LLMs on domain-specific datasets, as in AlphaCode (Li et al., 2022) and  
079 Deepseek-R1 (Ren et al., 2025), achieving strong results in code generation and mathematical rea-  
080 soning. Yet such methods often suffer from entropy collapse (Cui et al., 2025) and, as shown in Yue  
081 et al. (2025), rarely move beyond the base model’s capabilities. This makes it difficult to discover  
082 fundamentally new solutions, especially when sparse rewards further limit exploration. *Workflow-  
083 driven approaches* embed LLMs in predefined pipelines to improve task-specific performance. Ex-  
084 amples include integrating genetic algorithms with LLMs for enzyme discovery (Nana Teukam et al.,  
085 2025), establishing LLM-driven evolutionary loops such as LLaMEA (van Stein & Bäck, 2024), or  
086 applying evolutionary strategies to prompts (Agrawal et al., 2025). While effective on narrow tasks,  
087 these systems are highly sensitive to workflow design and rely on static pretrained knowledge, mak-  
088 ing it hard to reuse past discoveries to guide iterative search. Both categories thus struggle to gen-  
089 eralize in open-ended scientific domains where efficient exploration and continual refinement are  
090 essential.091 To this end, we propose **HELIX**—a **H**ierarchical **E**volutionary **L**earning framework  
092 with **I**n-context **e**Xperiences. **First, to learn from experience**, **HELIX** **updates the LLM policy using**  
093 **reward signals by reinforcement learning to progressively improve solution quality**. Meanwhile  
094 candidate solutions explored by the model forms a population for evolving algorithms. **Secondly,**  
095 to balance the quality and diversity, we propose to rank and select samples using both diversity and  
096 reward, inspired by a classic multi-objective evolutionary algorithm named NSGA-II(Deb et al.,  
097 2002). Specifically, to better measure the novelty of a solution, we compute diversity using a pre-  
098 trained language embedding model and estimate the diversity by kNN. Finally, we enable the model  
099 to stand on the shoulder of giants by adding a prompt constructed by the best solutions in the popu-  
100 lation to guide the model to generate new solutions. By using the in-context learning paradigm, we  
101 seamlessly unify and integrate evolutionary learning with reinforcement learning to explore the vast  
solution space in complex scientific problems.102 In experiments, we evaluated HELIX on **20** tasks across five diverse categories. Compared with  
103 strong task-specific baselines and advanced proprietary models such as GPT-4o, HELIX achieves  
104 superior performance on **17** tasks, demonstrating its ability to iteratively refine solutions and update  
105 its policy towards better results. Further analysis via ablation studies confirms that each component  
106 of HELIX contributes critically to performance. Notably, success on these unbounded and open-  
107 ended tasks suggests that iterative, diversity-aware exploration can provide useful insights for other  
scientific and engineering problems.

108 **2 RELATED WORK**

110 **Reinforcement learning of LLMs.** Training LLMs or LLM-based agents with reinforcement  
 111 learning (RL) has recently attracted significant attention. This includes reinforcement learning  
 112 from human feedback (RLHF) to align models with human preferences, as well as RL with veri-  
 113 fiable rewards (RLVR) to enhance reasoning, mathematical problem-solving, and coding capabili-  
 114 ties. Beyond improving reasoning, RLVR-style training can also elicit new capabilities such as  
 115 tool use (Feng et al., 2025) and information retrieval (Jin et al., 2025). A representative method is  
 116 GRPO (Shao et al., 2024), which normalizes rewards within groups of samples. Variants such as  
 117 DAPO (Yu et al., 2025) and Dr.GRPO (Liu et al., 2025) further improve GRPO through refined data  
 118 sampling strategies and advantage estimation techniques. While RL can improve generalization in  
 119 specific domains, the training process often suffers from decreasing entropy and diversity over time,  
 120 hindering effective exploration. Some approaches, such as KL-Cov (Cui et al., 2025), attempt to  
 121 address this limitation **by applying KL penalty solely to tokens with high covariance to preserve**  
 122 **entropy.** However, for complex scientific problems, **these memory-less RL methods—where the**  
 123 **sampling context for the same problem remains fixed—struggle to leverage solutions that have al-**  
 124 **ready been discovered, making it difficult to build upon prior explorations.**

125 **Evolutionary algorithms.** Evolutionary algorithms are a classic approach for tackling complex  
 126 optimization problems. They use “gene” to represent a solution for the problem and use random  
 127 mutation to explore the whole solution space. Recently, AlphaEvolve (Novikov et al., 2025) treats  
 128 code as the “gene” and applies LLM-driven mutations, successfully integrating LLM agents with  
 129 evolutionary algorithms—opening the door to solving complex scientific problems. Since then,  
 130 many works have adopted similar agent-based workflows to address scientific tasks such as CUDA  
 131 code optimization (Lange et al., 2025), drug discovery (Gao et al., 2025), and complex scientific  
 132 software usage (Fan et al., 2025; Pham et al., 2025). However, such methods typically require  
 133 highly problem-specific workflow logic and prompt design, which greatly limit their effectiveness  
 134 in solving more general and complex problems.

135 **3 PROPOSED METHOD**

136 **3.1 OVERVIEW**

137 To tackle the challenges of applying large language models (LLMs) to complex scientific discovery  
 138 tasks, we propose HELIX, a hybrid framework that integrates reinforcement learning with evolu-  
 139 tionary search. The goal is to enable LLMs to efficiently explore large and flexible solution spaces  
 140 while maintaining diversity and exploiting previously discovered high-quality solutions. The frame-  
 141 work is composed of three complementary modules: (1) **A reinforcement learning framework**  
 142 that updates the policy parameters based on verifiable reward, allowing the model to *learn from*  
 143 *experience* and progressively improve its reasoning capability. (2) **A multi-objective evolution-  
 144 ary mechanism** that *balancing solution quality and diversity*, ensuring that the population retains  
 145 both high-performing and diverse candidates for further expansion. (3) **An in-context learning**  
 146 **mechanism** that incorporates multiple past trials into the prompt, enabling the model to build upon  
 147 previously discovered solutions and *expand its exploration on the shoulder of giants*.

148 We consider the task as an optimization problem that has a solution space of code. Let  $s \in \mathcal{S}$  denote a  
 149 candidate solution, represented as code written in a domain-specific language (e.g., Python, YAML,  
 150 or other DSLs). We define an objective reward function  $R(\cdot)$  which only depends on the current  
 151 solution (state). The optimization objective is to find a valid  $s \in \mathcal{S}$  to maximize the reward:

$$\max_{s \in \mathcal{S}} R(s). \quad (1)$$

152 To explore and search for new solutions, we use an LLM policy  $\pi_\theta$  that **iteratively mutates(improves)**  
 153 **current solutions. Given timestep  $t$ , we sample a solution  $s_t$  from  $\mathcal{P}_t$ , the set of candidate solutions**  
 154 **at  $t$ -th step. The LLM will output an action  $a_t \in \mathcal{A}$ , which is an edit or modification applied to  $s_t$ ,**  
 155 **to obtain a new solution  $s_{t+1} = T(s_t, a)$ , where  $T$  is the transition function. Our goal is to improve**

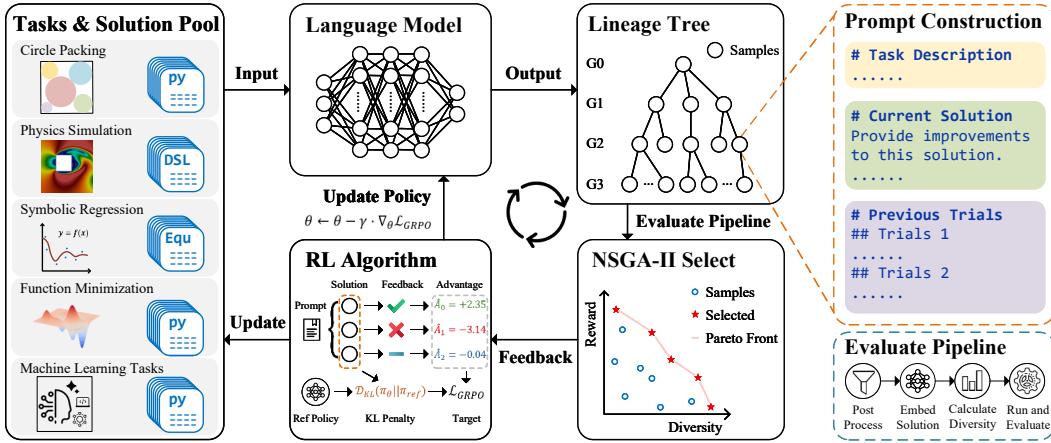


Figure 2: Illustration of HELIX framework. The workflow begins with a dataset containing task descriptions and a pool of initial solutions, which are taken by LLM as inputs. The LLM will modify and update the original solution and generate a new one, represented as descendants in lineage tree. After the evaluation pipeline, samples will be selected by NSGA-II algorithm to construct promising yet diverse candidate solutions for population evolution. The resulting reward-labeled solutions will also be used to update policy parameters via reinforcement learning.

the policy’s ability to find better solutions. The objective is defined as follows,

$$\max_{\theta} \mathbb{E}_{s_t \sim \mathcal{P}_t, a_t \sim \pi_{\theta}(\cdot | q, s_t)} [R(s_t, a_t)], \quad (2)$$

where  $q$  is the prompt constructed in equation 5 and  $R(s_t, a_t) = R(s_{t+1})$  is the reward of the new solution with a slight abuse of notations. We leverage GRPO (Shao et al., 2024), a reinforcement learning algorithm, to update LLM policy  $\pi_{\theta}$ . By maximize the reward in equation 2, the LLM will learn to enhance current solution  $s_t$  towards higher reward, which will finally leads to improvement in equation 1.

To address the exploration–exploitation trade-off and prevent entropy collapse in RL, we incorporate evolutionary algorithm in selection of candidate solutions. Suppose  $\mathcal{D}_t = \{s_t\}$  is the set of all solutions generated in the  $t$ -th iteration and  $\mathcal{D}_0 = \{s_0\}$  is the set of initial solution, the candidate solution for  $t$ -th step can be constructed as

$$\mathcal{P}_t = \text{SelectTop}_{\text{NSGA-II}}\left(\bigcup_{s=0}^t \mathcal{D}_s\right), \quad (3)$$

where NSGA-II (Deb et al., 2002) is a sample selection strategy widely adopt in evolutionary algorithms, which ensures retention of high-reward and diverse candidates. This formulation allows the model to iteratively improve its policy while exploiting previously found high-quality solutions as starting points for further exploration. Figure 2 provides a brief summary of our method and the formalized algorithm can be found in Appendix G.

### 3.2 POLICY OPTIMIZATION ALIGNED WITH EVOLUTIONARY SEARCH

As the evolutionary process unfolds, updating the model parameters becomes crucial: it enables the policy to learn from both successful and failed trials, generate higher-quality solutions, and dynamically adapt to the shifting input distribution induced by the evolutionary search. Reinforcement learning is particularly suitable in this scientific setting, since open-ended scientific tasks lack standard answers and typically provide only sparse reward feedback. Motivated by the design of GRPO (Shao et al., 2024), we develop a reinforcement learning–based policy update mechanism tailored to our framework. GRPO has proven effective in enhancing LLM reasoning on mathematical and programming tasks (Guo et al., 2025), and its multi-sample generation naturally provides diverse reasoning-driven outputs that enrich the evolutionary dataset, making it a natural inspiration for our method.

216 Formally, given a prompt  $q$ , the model will generate  $G$  rollout sequences  $\{a_j\}_{1 \leq j \leq G}$  with policy  
 217  $\pi_{\theta_{\text{old}}}$ . The GRPO objective is then defined as:  
 218

$$219 \mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{s_t \sim \mathcal{P}_t, \{a_j\}_{j=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q, s_t)} \\ 220 \left[ \frac{1}{G} \sum_{j=1}^G \frac{1}{|a_j|} \sum_{k=1}^{|a_j|} \left( \min(r_{j,k}(\theta) \hat{A}_{j,k}, \text{clip}(r_{j,k}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{j,k}) - \beta D_{\text{KL}}(\pi_{\theta} \| \pi_{\text{ref}}) \right) \right], \\ 221 \quad (4)$$

222 where  $r_{j,k}(\theta) = \frac{\pi_{\theta}(a_{j,k} | q, a_{j,<k})}{\pi_{\theta_{\text{old}}}(a_{j,k} | q, a_{j,<k})}$  is the token-level policy ratio,  $\hat{A}_{j,k} = \frac{R(s_t, a_j) - \text{mean}_j\{R(s_t, a_j)\}}{\text{std}_j\{R(s_t, a_j)\}}$  is  
 223 the token-level advantage,  $\epsilon$  is the clipping parameter, and  $\beta$  controls the KL divergence penalty  
 224 against a reference policy  $\pi_{\text{ref}}$ .  
 225

226 In order to fully leverage the in-context learning ability of LLMs, enabling the model to learn from  
 227 feedback of previous trials and propose advanced solutions, we construct the prompt  $q$  in the fol-  
 228 lowing manner:  
 229

$$230 q = \text{ConstructPrompt}(\{p\} \cup \{s_t, R(s_t), F(s_t)\} \cup \{f^{(k)}(s_t), R(f^{(k)}(s_t)), F(f^{(k)}(s_t))\}_{1 \leq k < n}), \quad (5)$$

231 where  $p$  is the problem description,  $f^{(k)}(s_t)$  is the  $k$ -th ancestor of  $s_t$  in lineage tree (a historical  
 232 trace of the solution  $s_t$ 's iterative refinement),  $R(\cdot)$  represents the reward function and  $F(\cdot)$  denotes  
 233 the auxiliary feedback (e.g., textual or structured evaluations) provided by the evaluator to guide  
 234 future refinements. By constructing prompts using memory of previous feedback and rewards along  
 235 a lineage tree, it ensures the model effectively explores across challenging solution spaces.  
 236

### 237 3.3 EVOLUTIONARY MECHANISM FOR BALANCING QUALITY AND DIVERSITY

238 In unbounded scientific research tasks, it is crucial to explore multiple promising ideas or directions.  
 239 Thus, the optimization process must balance *quality*, i.e., high-reward solutions that serve as strong  
 240 starting points for refinement, with *diversity*, which sustains broad exploration across the solution  
 241 space. We design the evolutionary search algorithm to be a multi-objective optimization that natu-  
 242 rally achieves a trade-off by maintaining a population that simultaneously improves in reward and  
 243 preserves diverse candidates. Specifically, we innovatively adopt NSGA-II Deb et al. (2002), which  
 244 is a powerful multi-objective optimization algorithm, to filter high quality and diverse samples on the  
 245 Pareto front of reward and diversity for subsequent expansion. To further encourage more diverse  
 246 exploration and enable more accurate diversity computation, we propose to compute the diversity  
 247 score based on its semantic embedding similarity using a pretrained language embedding model.  
 248

249 **Diversity measurement.** To quantify the diversity of candidate solutions, we first normalize each  
 250 solution into a canonical code format and encode it into an embedding vector using a pretrained  
 251 embedding model. Let  $\mathcal{D} = \bigcup_{0 \leq s \leq t} \mathcal{D}_s$  represents the union of all solutions,  $E(s) \in \mathbb{R}^d$  denote the  
 252 embedding of solution  $s \in \mathcal{D}$ . For any solution  $s_i$ , its diversity score is computed by measuring the  
 253 average similarity to its  $k$  nearest neighbors in the embedding space:  
 254

$$255 \text{Div}(s_i) = 1 - \frac{1}{k} \sum_{j \in \mathcal{N}_k(i)} \frac{E(s_i) \cdot E(s_j)}{\|E(s_i)\| \|E(s_j)\|}, \quad (6)$$

256 where  $\mathcal{N}_k(i)$  denotes the indices of the  $k$  nearest neighbors of  $s_i$  in  $\mathcal{D}$ , measured by cosine similarity.  
 257 A higher  $\text{Div}(s_i)$  indicates that  $s_i$  is more distinct from other solutions, thereby contributing to  
 258 population diversity.  
 259

260 **NSGA-II based selection.** Given both reward score  $R(s)$  and diversity score  $\text{Div}(s)$ , each can-  
 261 didate solution can be mapped to a two-dimensional objective space. We then adopt the NSGA-  
 262 II (Deb et al., 2002) algorithm to select high-quality and diverse samples. NSGA-II first applies a  
 263 nondominated sorting procedure to partition solutions into multiple fronts based on Pareto domi-  
 264 nance, where a solution  $s_a$  dominates  $s_b$  if  $R(s_a) \geq R(s_b)$  and  $\text{Div}(s_a) \geq \text{Div}(s_b)$  with at least  
 265 one strict inequality. To further ensure diversity preservation within each front, NSGA-II computes  
 266

270 a crowding-distance measure and selects representative samples that are well spread in the objective  
 271 space.

272 By combining nondominated sorting with diversity preservation, the resulting population  $\mathcal{P}$  retains  
 273 candidates that are both high-reward and diverse. This mechanism allows the model to continuously  
 274 exploit promising solutions while sustaining exploration across multiple distinct solution trajec-  
 275 tories.

## 277 4 EXPERIMENT

280 In this section, we first introduce the experimental setup, including the tasks we selected for bench-  
 281 marking the model’s ability to solve open-ended scientific problems. Then, we present extensive  
 282 experiments demonstrating that HELIX effectively enhances model capability, integrates historical  
 283 experience, and balances reward with diversity, leading to significant improvements over existing  
 284 baselines in solving unbounded and open-ended scientific challenges. Finally, the ablation studies  
 285 reveal how different components of the framework work together in a complementary manner.

### 286 4.1 EXPERIMENT SETTING

288 **Tasks.** To comprehensively evaluate the model’s capacity for complex scientific reasoning, we  
 289 design experiments on five representative categories of tasks. These tasks are particularly suited for  
 290 our study because they are *unbounded*, lacking a guaranteed global optimum, *open-ended*, requiring  
 291 exploration over vast and flexible solution spaces and *domain-specific*, containing unique constraints  
 292 and complex background. Success in these tasks not only demonstrates the model’s ability to search  
 293 beyond local optima, but also provides insights that can inspire solutions in broader scientific and  
 294 engineering domains.

- 296 1. **Machine Learning Tasks.** We selected three representative datasets: Adult in-  
 297 come (Becker & Kohavi, 1996), Bank marketing (Moro et al., 2014) and Boston hous-  
 298 ing (Harrison Jr & Rubinfeld, 1978) dataset to evaluate the model’s ability to solve machine  
 299 learning tasks. These tasks reflects the open-ended challenge of combining ML algorithms  
 300 for novel applications, with potential implications for autonomous scientific workflows.
- 301 2. **Physics Simulation Tasks.** These tasks combine geometric structures design and optimiza-  
 302 tion in multi-physics environments in distinct fields. The design space of these problems  
 303 has a very high degree of freedom with few global optimal solution.
- 304 3. **Circle Packing Problems.** The objective of these tasks is to maximize the sum of radii of  
 305 circles packed within given shapes. It allows multiple feasible arrangements and there is  
 306 no proved global optimum solution currently.
- 307 4. **Function Minimization.** It requires LLM to write a code to find the global minimum point  
 308 of given functions. Agents can search freely for new mathematical optimization methods  
 309 in code space.
- 310 5. **Symbolic Regression.** A benchmark (Shojaee et al., 2025) evaluates the ability of LLMs  
 311 to hypothesize underlying expressions for noisy data. The model needs to search among a  
 312 vast possible expression set and utilize domain specific knowledge to find solution.

314 **Models.** We selected the DeepSeek-R1-Distill-Qwen model family for our experiment due to its  
 315 strong reasoning capabilities and manageable size, which is critical for performing complex sci-  
 316 entific tasks under computational constraints. Among the model family, the 14B version offers an  
 317 optimal balance between efficiency and performance, and was selected as the model in the main  
 318 results. For physics simulation tasks that require strong geometric reasoning ability and physical  
 319 prior knowledge, we utilize the 32B version of the model.

321 **Baselines.** We compare our approach against three key baselines:

- 323 1. **Direct Prompt (Test-Time Scaling):** Queries the model directly and selects the best out-  
 324 come from multiple samples to establish a performance upper bound of base model.

324  
 325 Table 1: Results of main experiments. All values correspond to the best outcome obtained across all  
 326 attempts. We use  $\uparrow$  to indicate that larger values correspond to better performance, and  $\downarrow$  represents  
 327 the opposite. We highlighted the best results in each task in **bold**. "NA" denotes non-convergence  
 328 or unsuitability for given case.

		Task Specific Methods		Direct Prompt		Open Evolve		Ours
Tasks		LightGBM	RRL	Qwen	GPT-4o	Qwen	GPT-4o	-
Machine Learning	Adult Income $\uparrow$	80.36	80.72	73.72	76.91	76.90	72.27	<b>82.07</b>
	Bank Marketing $\uparrow$	75.28	76.32	0.00	76.91	75.66	78.54	<b>80.65</b>
	Boston Housing $\downarrow$	3.258	3.966	3.149	3.031	2.937	2.937	<b>1.747</b>
	Transparent Conductors $\downarrow$	0.060	NA	0.060	0.059	0.059	0.056	<b>0.049</b>
Tasks		Parameter Scan	Topology Opt	Qwen	GPT-4o	Qwen	GPT-4o	-
Physics Simulation	Inductor $\uparrow$	6.111	6.248	2.584	0.001	1.637	1.652	<b>9.609</b>
	Beam Bending $\uparrow$	4.771	NA	5.407	4.005	10.793	6.352	<b>17.298</b>
	Magnetic Torque $\uparrow$	10.273	NA	0.323	1.201	3.488	1.607	<b>11.045</b>
	Periodic Heat $\uparrow$	1.206	NA	1.258	1.255	1.233	1.266	<b>1.278</b>
	Demultiplexer $\uparrow$	18.322	<b>23.555</b>	3.364	4.532	12.341	8.645	14.260
Tasks		SLSQP	Genetic Algo	Qwen	GPT-4o	Qwen	GPT-4o	-
Circle Packing	Packing in Unit Square $\uparrow$	2.519	2.345	1.673	1.900	1.586	2.611	<b>2.636</b>
	Packing in Unit Disk $\uparrow$	4.522	3.896	4.608	3.290	4.604	3.984	<b>4.664</b>
Tasks		SLSQP	Trust-constr	Qwen	GPT-4o	Qwen	GPT-4o	-
Function Minimization	Eggholder $\uparrow$	0.705	0.688	<b>1.000</b>	0.959	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>
	Mishras Bird $\uparrow$	0.814	0.764	<b>1.000</b>	0.996	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>
	Kanees Bump 10d $\uparrow$	0.714	0.692	0.886	0.987	<b>1.000</b>	0.997	<b>1.000</b>
	Kanees Bump 20d $\uparrow$	0.603	NA	0.794	0.657	0.596	0.983	<b>1.000</b>
	Kanees Bump 30d $\uparrow$	0.594	NA	0.923	0.625	0.677	0.668	<b>0.994</b>
Tasks		LLM-SR	LaSR	Qwen	GPT-4o	Qwen	GPT-4o	-
Symbolic Regression	Chemistry $\downarrow$	4.12e-6	9.11e-5	2.66e-5	<b>2.44e-6</b>	1.59e-5	9.52e-6	7.32e-6
	Biology $\downarrow$	3.06e-6	1.53e-4	1.26e-4	7.52e-5	1.64e-4	5.31e-5	<b>2.98e-8</b>
	Physics $\downarrow$	7.62e-5	9.94e-4	2.71e-4	1.13e-4	2.76e-5	1.22e-4	<b>2.76e-5</b>
	Material Science $\downarrow$	<b>3.21e-9</b>	9.23e-6	7.14e-6	1.85e-6	6.99e-7	1.94e-6	4.46e-6

353  
 354 2. **Open Evolve** (Sharma, 2025): An open-source implementation of the AlphaE-  
 355 evolve (Novikov et al., 2025) framework, which uses an evolutionary algorithm with multi-  
 356 ple LLM roles (e.g., proposing code mutations, evaluating fitness) to iteratively generate,  
 357 test, and evolve code or solutions across generations.

358 3. **Task-Specific Methods:** Represents results from established algorithms designed for each  
 359 specific problem. Details of these methods can be found in Appendix C.

## 361 4.2 MAIN RESULTS

363 Table 1 presents the results of our methods compared to various baselines. The best results in  
 364 each task are highlighted in **bold**. Since we selected multiple heterogeneous tasks, their evaluation  
 365 metrics are not the same. The detailed definitions and specific evaluation criteria are deferred to  
 366 Appendix B.

367 Across the 20 benchmark tasks, our method achieves the best performance on 17 tasks, surpassing  
 368 all competing baselines. Compared under the same model settings, our framework consistently out-  
 369 performs Direct Prompting across all benchmarks. Against OpenEvolve—the open-source version  
 370 of AlphaEvolve—it achieves superior results on 19 tasks. These results clearly highlight the strength  
 371 of our framework in solving open-ended scientific problems among various domains compared to  
 372 other approaches.

373 Notably, we observe that the base Qwen models perform relatively poorly on certain tasks such as  
 374 Bank Marketing and Magnetic Torque, exhibiting low rewards even in the best of 64 direct trials.  
 375 However, our framework significantly improves performance in these cases by leveraging parameter  
 376 updates and in-context learning to effectively incorporate feedback from the exploration process.  
 377 This demonstrates that our approach can partially overcome the limitations of weaker base models  
 by iteratively evolving toward superior solutions.

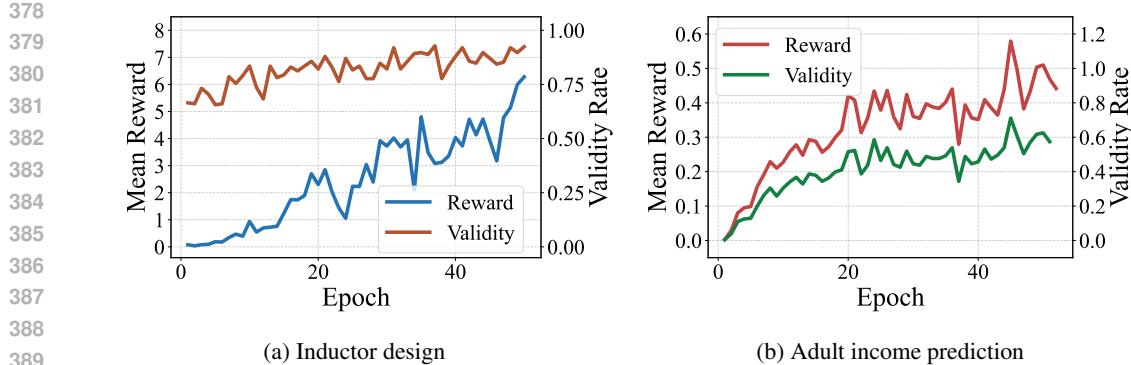


Figure 3: Convergence analysis on the Inductor and Adult tasks. The curves show the progressive improvement of average reward and validity during training, demonstrating that our framework effectively leverages reinforcement learning feedback and evolutionary dynamics to produce increasingly valid and high-quality solutions.

To further assess the competitiveness of our approach against state-of-the-art scientific discovery systems, we compared it with GPT-4o, one of the most advanced closed-source models. Remarkably, our method outperforms GPT-4o on **18** tasks, regardless of whether GPT-4o is equipped with multi-role collaborative reasoning frameworks. These results highlight that our framework can fully exploit the prior knowledge of smaller models through reinforcement learning, enabling cost-efficient and effective solutions to complex scientific problems.

In comparison with task-specific methods, which are typically crafted by human experts for particular domains, our framework still achieves superior performance on **17** tasks. Specifically, in the circle packing task, we establish a new world record 2.635983 using only a 14B model. **For the Transparent Conductors dataset, derived from a human-participation competition (Ziletti et al., 2017), our framework attains the second-highest score on the participants’ leaderboard.** This highlights its ability to iteratively evolve within open-ended solution spaces and to autonomously uncover novel solutions that go beyond manually designed approaches.

To provide further evidence that our framework effectively integrates reinforcement learning and evolutionary algorithms, we analyze its convergence behavior on two representative cases: inductor design and adult income prediction. Figure 3 plots the average reward and validity of model outputs during training. Both metrics exhibit a clear upward trend: the validity rate rises steadily, showing that the model increasingly generates outputs that satisfy task constraints, while the average reward improves, reflecting higher-quality solutions. This dual improvement demonstrates that reinforcement learning progressively strengthens the model’s intrinsic reasoning ability. It also indicates that the quality of the evolving population keeps improving, enabling the model to leverage in-context feedback as well as intuitions from high-reward solutions to generate better outputs.

### 4.3 ABLATION STUDY

#### 4.3.1 EFFECTIVENESS OF FRAMEWORK COMPONENTS

To better understand the contribution of each component in our framework, we conduct ablation studies on the Boston Housing and Circle Packing tasks. We design several controlled variants by selectively disabling or simplifying parts of the algorithm: **TopScore**, where only the highest-reward candidate in the dataset is selected for further evolution; **TopDiv**, where selection relies solely on diversity without considering reward; **Random**, where candidates are sampled randomly from the population; **EvoOnly**, where the model parameters are kept fixed and only the evolutionary pipeline is applied; and **TrainOnly**, which removes the evolutionary mechanism and in-context prompting, reducing the framework to pure GRPO reinforcement learning. These variants allow us to disentangle the relative importance of reward-driven selection, diversity maintenance, evolutionary population updates, and reinforcement learning in driving overall performance.

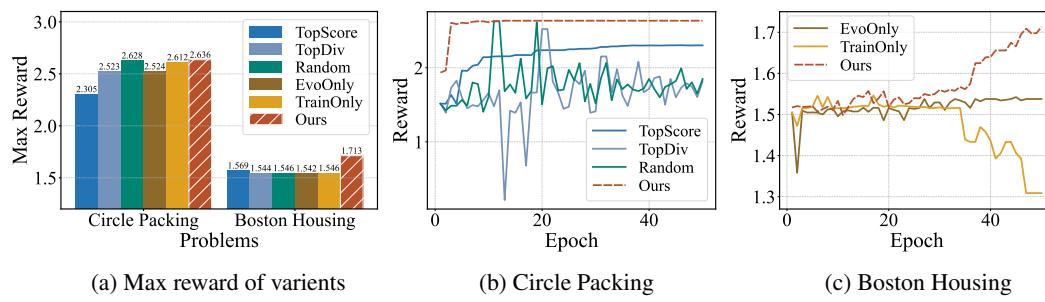


Figure 4: Ablation analysis of framework components. **(a):** Maximum reward achieved by different ablation variants. **(b):** Curve of epoch-wise maximum reward on the Circle Packing task, highlighting the critical role of balancing diversity and quality for stable optimization. **(c):** Curve of epoch-wise maximum reward on the Boston Housing task, showing the necessity of combining reinforcement learning with evolutionary guidance.

Figure 4a reports the maximum reward achieved under different ablation settings. Across both tasks, all variants perform worse than our full framework, confirming the necessity of each component. We next analyze the results task by task.

For the **Circle Packing** problem, high-quality solutions rely on diverse initial starting points for optimization algorithms. As shown in Figure 4b, eliminating diversity (TopScore) significantly reduces reward, since the search quickly collapses into narrow solution modes. In contrast, Random and TopDiv maintain higher diversity, enabling the model to extend from a richer set of initial states. However, focusing solely on diversity also leads to instability—visible in the large variance of TopDiv and Random—whereas TopScore and our full method (Ours) remain relatively stable. This instability disrupts training and prevents the model from finding strong solutions in later epochs. [Moreover, we conducted a detailed analysis on the performance gap between OpenEvolve and HELIX in Appendix E, which demonstrate the effectiveness of explicitly combining diversity with embedding model in our framework.](#) These results highlight that balancing diversity and solution quality is critical for solving such problems.

For the **Boston Housing** task, strong performance requires careful parameter tuning and complex feature engineering, which typically emerge from iteratively learning from past experience. As shown in Figure 4c, disabling either reinforcement learning or evolution severely limits performance. With EvoOnly, the model remains bounded by its initial capacity and fails to break through training bottlenecks. Conversely, with TrainOnly, the model cannot effectively accumulate knowledge in context and collapses during training. These results demonstrate that both parameter updates and in-context evolutionary guidance are indispensable for helping the model accumulate expertise and progressively refine its solutions.

#### 4.3.2 SCALING EXPERIMENTS

Here, we discuss the impact of base model size on task performance. We evaluate our framework on two representative tasks, Magnetic Torque Maximization and Inductor Design, using the DeepSeek-R1-Distill-Qwen model family with 1.5B, 7B, 14B, and 32B parameters. As shown in Figure 5, for the magnetic torque task, the reward steadily increases with model size, indicating stronger reasoning ability and more effective exploration. For the inductor design task, we observe a reward plateau around 9.6. However, the mean reward continues to grow as model size increases, suggesting that larger models generate more valid and higher-quality candidates. These results demonstrate that our framework exhibits scaling property: as the underlying LLM grows, the system can push the boundaries of scientific discovery by enabling more efficient and higher-quality exploration.

## 5 CONCLUSION

In this work, we proposed HELIX, a hierarchical evolutionary reinforcement learning framework with in-context experiences. By integrating reinforcement learning, evolutionary selection, and in-

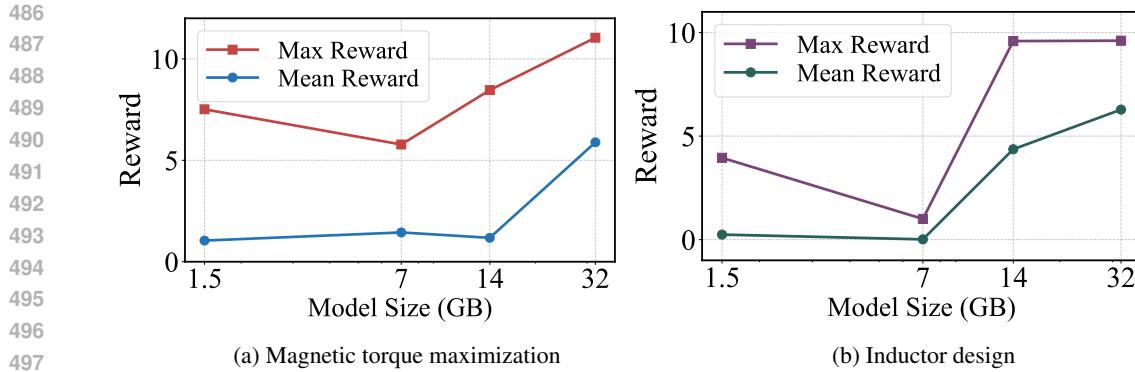


Figure 5: Scaling analysis of model parameter scale on (a): magnetic torque maximization and (b): inductor design tasks.

context trial incorporation, HELIX effectively balances exploration and exploitation, enables task-specific adaptation, and iteratively refines solutions. Extensive experiments across 20 tasks in five diverse categories demonstrate that HELIX consistently outperforms strong task-specific baselines and advanced proprietary models. Overall, HELIX shows strong potential for advancing open-ended scientific discovery by enabling iterative, diversity-aware exploration. Looking ahead, it could provide a foundation for broader applications in engineering, optimization, and autonomous research systems.

## ETHICS STATEMENT

This work focuses on developing a hybrid reinforcement learning and evolutionary framework for solving complex scientific problems. It does not involve human subjects, sensitive personal data, or proprietary datasets, and thus raises no direct ethical or privacy concerns. All datasets used are publicly available and widely adopted in prior research. All authors have reviewed and agree to abide by the ICLR Code of Ethics as linked above, and affirm that this submission complies with the principles of honesty, transparency, fairness, and responsible conduct.

## REPRODUCIBILITY STATEMENT

Detailed descriptions of the experimental setup, task definitions, and evaluation metrics are provided in Appendix A and Appendix B.

Source code will be available at <https://anonymous.4open.science/r/HELIX-1829/>.

## REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Tasnim Ahmed and Salimur Choudhury. Lm4opt: Unveiling the potential of large language models in formulating mathematical optimization problems. *INFOR: Information Systems and Operational Research*, 62(4):559–572, 2024.

Oliver A Bauchau and James I Craig. Euler-bernoulli beam theory. In *Structural analysis*, pp. 173–221. Springer, 2009.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: <https://doi.org/10.24432/C5XW20>.

COMSOL AB. Comsol multiphysics®, 2024. URL [www.comsol.com](http://www.comsol.com).

540 Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.  
 541

542 Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen  
 543 Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for  
 544 reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

545 Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *Inter-  
 546 national Conference on Learning Representations (ICLR)*, 2024.  
 547

548 Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multi-  
 549 objective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):  
 550 182–197, 2002.

551 E Fan, Weizong Wang, and Tianhan Zhang. Chatcf: an end-to-end cfd agent with domain-specific  
 552 structured thinking. *arXiv preprint arXiv:2506.02019*, 2025.

553 Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang,  
 554 Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms.  
 555 *arXiv preprint arXiv:2504.11536*, 2025.

556 Ali Forootani. A survey on mathematical reasoning and optimization with large language models.  
 557 *arXiv preprint arXiv:2503.17726*, 2025.

558 Bowen Gao, Yanwen Huang, Yiqiao Liu, Wenxuan Xie, Wei-Ying Ma, Ya-Qin Zhang, and Yanyan  
 559 Lan. Pharmagents: Building a virtual pharma with large language model agents. *arXiv preprint  
 560 arXiv:2503.22164*, 2025.

561 Arya Grayeli, Atharva Sehgal, Omar Costilla Reyes, Miles Cranmer, and Swarat Chaudhuri. Sym-  
 562 bolic regression with a learned concept library. *Advances in Neural Information Processing Sys-  
 563 tems*, 37:44678–44709, 2024.

564 Daya Guo, Dejian Yang, Huawei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,  
 565 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms  
 566 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

567 David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air.  
 568 *Journal of environmental economics and management*, 5(1):81–102, 1978.

569 Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and  
 570 Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement  
 571 learning. *arXiv preprint arXiv:2503.09516*, 2025.

572 Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-  
 573 Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural  
 574 information processing systems*, 30, 2017.

575 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph  
 576 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model  
 577 serving with pagedattention. In *Proceedings of the 29th symposium on operating systems prin-  
 578 ciples*, pp. 611–626, 2023.

579 Robert Tjarko Lange, Aaditya Prasad, Qi Sun, Maxence Faldor, Yujin Tang, and David Ha. The ai  
 580 cuda engineer: Agentic cuda kernel discovery, optimization and composition. Technical report,  
 581 Technical report, Sakana AI, 02 2025, 2025.

582 Charles L Lawson and Richard J Hanson. *Solving least squares problems*. SIAM, 1995.

583 Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom  
 584 Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation  
 585 with alphacode. *Science*, 378(6624):1092–1097, 2022.

586 Xianggen Liu, Yan Guo, Haoran Li, Jin Liu, Shudong Huang, Bowen Ke, and Jiancheng Lv.  
 587 Drugllm: Open large language model for few-shot molecule generation. *arXiv preprint  
 588 arXiv:2405.06690*, 2024.

594 Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee,  
 595 and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint*  
 596 *arXiv:2503.20783*, 2025.

597 S. Moro, P. Rita, and P. Cortez. Bank Marketing. UCI Machine Learning Repository, 2014. DOI:  
 598 <https://doi.org/10.24432/C5K306>.

600 Yves Gaetan Nana Teukam, Federico Zipoli, Teodoro Laino, Emanuele Criscuolo, Francesca  
 601 Grisoni, and Matteo Manica. Integrating genetic algorithms and language models for enhanced  
 602 enzyme design. *Briefings in bioinformatics*, 26(1):bbae675, 2025.

603 Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt  
 604 Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian,  
 605 et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint*  
 606 *arXiv:2506.13131*, 2025.

607 Thang D Pham, Aditya Tanikanti, and Murat Keçeli. Chemgraph: An agentic framework for com-  
 608 putational chemistry workflows. *arXiv preprint arXiv:2506.06363*, 2025.

610 ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang,  
 611 Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical rea-  
 612 soning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*,  
 613 2025.

614 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,  
 615 Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathemati-  
 616 cal reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

617 Asankhaya Sharma. Opennevolve: an open-source evolutionary coding agent, 2025. URL <https://github.com/codelion/openevolve>.

620 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,  
 621 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*  
 622 *arXiv: 2409.19256*, 2024.

623 Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy.  
 624 Llm-sr: Scientific equation discovery via programming with large language models. *arXiv*  
 625 *preprint arXiv:2404.18400*, 2024.

627 Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and  
 628 Chandan K Reddy. Llm-srbench: A new benchmark for scientific equation discovery with large  
 629 language models. *arXiv preprint arXiv:2504.10415*, 2025.

630 Niki van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for  
 631 automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 2024.

633 Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning  
 634 for interpretable classification. *Advances in Neural Information Processing Systems*, 34:30479–  
 635 30491, 2021.

636 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian  
 637 Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system  
 638 at scale. *arXiv preprint arXiv:2503.14476*, 2025.

639 Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does re-  
 640inforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv*  
 641 *preprint arXiv:2504.13837*, 2025.

642 Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright,  
 643 Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully  
 644 sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

646 Angelo Ziletti, Chris, Maggie, and Will Cukierski. Nomad2018 predicting  
 647 transparent conductors. <https://kaggle.com/competitions/nomad2018-predict-transparent-conductors>, 2017. Kaggle.

## 648 A TRAINING AND EVALUATION DETAILS

649  
 650 **Training.** We primarily use DeepSeek-Distill-Qwen-14B and 32B as the backbone models in our  
 651 experiments. The models are fine-tuned with the VERL framework (Sheng et al., 2024) under the  
 652 GRPO algorithm. Each model is trained for 80 epochs with a fixed learning rate of  $1 \times 10^{-6}$ ,  
 653 updating all parameters. We set the KL coefficient in GRPO to  $1 \times 10^{-3}$  and the number of rollouts  
 654 to 16. The rollouts are generated via VLLM (Kwon et al., 2023) backend with temperature equals  
 655 to 1.0 and top\_p equals to 0.95. Training was conducted using eight A100 GPUs for 14B models  
 656 and sixteen H100 GPUs for 32B models. For training efficiency, we use Pytorch FSDP (Zhao et al.,  
 657 2023) with parameter offload and optimizer offload. Gradient checkpoint and Flash-Attention (Dao,  
 658 2024) are used by default.

659  
 660 **Evaluation.** The evaluation is performed on a Slurm Workload Manager system. For each job,  
 661 we allocate 4 Intel(R) Xeon(R) Platinum 8168 CPUs for execution and impose time limits for each  
 662 task: five minutes for physics simulation, two minutes for machine learning and function minimiza-  
 663 tion, and one minute for circle packing and symbolic regression. The execution time includes the  
 664 time for task-dependent evaluators to calculate reward. For the detailed evaluate metric and reward  
 665 calculation, please refer to Appendix B.

## 666 B DEFINITION AND EVALUATION OF PROBLEMS

667 In this section we explain the detailed problem definition and evaluation metrics of all the tasks used  
 668 in the experiment.

### 669 B.1 MACHINE LEARNING

670 We selected 3 classic machine learning datasets, and the model has to write Python code to max-  
 671 imize the F1 score for classification tasks and minimize the rooted mean square error (RMSE) for  
 672 regression tasks. The details are described below.

#### 673 B.1.1 ADULT INCOME

674 The Adult income dataset (Becker & Kohavi, 1996) is a well-known binary classification task. The  
 675 goal is to predict whether a person's income exceeds \$50,000 per year based on various demographic  
 676 features such as age, education, marital status, and occupation. The dataset is sourced from the 1994  
 677 U.S. Census and contains both categorical and numerical features, with some missing values.

678 The dataset itself contains a separate train and test split. We then load the train set for model's  
 679 training and evaluate its result on the test set. The reward is the Macro F1 score, defined as:

$$680 R = \frac{1}{C} \sum_{c=1}^C \frac{2 \cdot P_c \cdot R_c}{P_c + R_c} \quad (7)$$

681 where  $P_c, R_c$  are the precision and recall for class  $c$ , and  $C = 2$  is the total number of classes.

#### 682 B.1.2 BANK MARKETING

683 The Bank marketing dataset (Moro et al., 2014) is another binary classification problem. It includes  
 684 data from a Portuguese bank's direct marketing campaigns, where the objective is to predict whether  
 685 a client will subscribe to a term deposit. This dataset is characterized by a high number of categor-  
 686 ical features and a significant class imbalance, making it a good benchmark for evaluating model  
 687 performance under challenging real-world conditions.

688 To ensure a robust evaluation, we use a 5-fold cross-validation strategy with StratifiedKFold in  
 689 sklearn to handle the class imbalance. The data is randomly split into five folds, maintaining the  
 690 same class distribution in each fold as in the original dataset. The model is trained and evaluated  
 691 five times, with each fold serving as the test set once. The final reward is the average of the Macro  
 692 F1 scores obtained from all five folds. If a task fails to produce a result in any fold, its reward is  
 693 considered to be 0 for that fold. The final result is the Macro F1 score, as defined in equation 7.

702 B.1.3 BOSTON HOUSING  
703

704 The Boston housing dataset (Harrison Jr & Rubinfeld, 1978) is a classic regression problem. The  
705 task is to predict the median value of owner-occupied homes in Boston suburbs, based on 13 features.  
706 These features include per capita crime rate, a number of rooms per dwelling, and the proportion  
707 of non-retail business acres. While the original dataset is no longer widely used for research due to  
708 ethical concerns, it remains a common benchmark for teaching and evaluating regression models.

709 To evaluate model performance, we use a 5-fold cross-validation strategy with KFold, splitting the  
710 data into five folds. The model is trained and evaluated five times, with each fold serving as the test  
711 set once. The final reward for this task is the average of the scores from all five folds. The reward is  
712 calculated using the following formula:

$$713 R = 2 - \log_{10}(\text{RMSE} + 10^{-10}), \quad (8)$$

714 where:

$$715 \text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (9)$$

716 This reward metric is designed to penalize larger RMSE values while rewarding smaller ones. If a  
717 task fails in any fold, its reward is considered to be 0 for that fold.

718 B.1.4 TRANSPARENT CONDUCTORS  
719

720 The Transparent Conductors dataset (Ziletti et al., 2017) is motivated by the need for accelerated  
721 discovery of materials that simultaneously exhibit optical transparency and electrical conductivity—two properties that are typically at odds. Such materials are central to modern technologies  
722 including photovoltaic cells, LEDs, sensors, touch screens, and display panels. Despite their im-  
723 portance, only a limited number of compounds are currently known to meet the desired trans-  
724 parency-conductivity trade-off, making data-driven exploration an appealing alternative to costly  
725 experimental or quantum-mechanical searches.

726 The dataset contains computationally derived information for 3,000 candidate materials belong-  
727 ing to the sesquioxide alloy family  $(Al_xGa_yIn_z)_{2N}O_{3N}$ , where the compositional ratios satisfy  
728  $x + y + z = 1$  and the total number of atoms in the unit cell ranges from 5 to 100. These materials  
729 are of particular interest due to their large bandgaps, chemical stability, and relatively low produc-  
730 tion cost. Each entry includes crystallographic descriptors (e.g., space group, lattice parameters),  
731 compositional ratios, and structural characteristics, offering a rich feature space for modeling.

732 The task is to predict two key target properties for each material: (1) formation energy, which reflects  
733 thermodynamic stability, and (2) bandgap energy, which determines visible-range transparency. Acc-  
734 curate prediction of these quantities enables efficient screening of new transparent conductor candi-  
735 dates without the need for expensive density-functional theory (DFT) calculations.

736 Model performance is evaluated using the root mean squared logarithmic error (RMSLE), computed  
737 column-wise for the two target properties. For a single target, the RMSLE is defined as:

$$738 \text{RMSLE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}, \quad (10)$$

739 where  $n$  is the number of samples,  $p_i$  denotes the predicted value, and  $a_i$  the ground-truth value.  
740 The final reward for model training is:

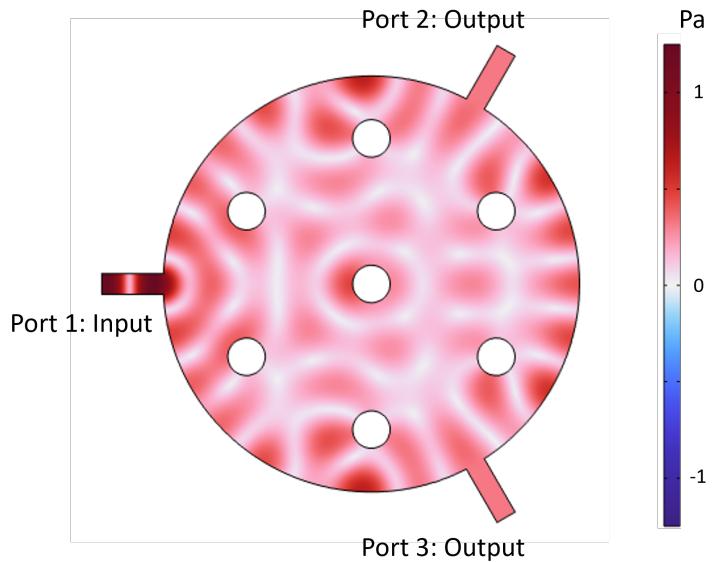
$$741 R = 1 - \text{RMSLE} \quad (11)$$

750 B.2 PHYSICS SIMULATION  
751

752 To test the model’s capacity for geometric reasoning and ability to utilize physics prior knowledge  
753 to discover better designs, we proposed the following physics simulation tasks. These tasks mainly  
754 require the model to generate a yaml representation of a complex geometry under certain constraints  
755 to maximize the reward. We utilize COMSOL Multiphysics® (COMSOL AB, 2024), a commercial  
756 FEA software for industrial multiphysics simulations, for the evaluation backend.

756 B.2.1 ACOUSTIC DEMULTIPLEXER  
757

758 This task aims to design an acoustic demultiplexer. The demultiplexer is a data distributing device  
759 which takes acoustic energy from the input port and distributes different frequency bands to the  
760 specific output port. The model is asked to propose the cavity geometry within a circular domain as  
761 seen in Fig. 6 to maximize the acoustic pressure at output port 2 while minimizing the pressure at  
762 output port 3. The input acoustic pressure level is set to 1 Pa at port 1, and the frequency level is set  
763 to 7500 Hz.



783 Figure 6: The RMS pressure field of an acoustic demultiplexer at frequency level 7500 Hz. The  
784 RMS pressure field in log scale is proportional to the acoustic power.  
785

786 The model is guided by the following reward  $R$  where  $P_i$  is the power output at port  $i$ ,  $p_{rms}$  is the  
787 Root Mean Square (RMS) pressure field,  $\rho$  is fluid density, and  $c$  is sound speed.  
788

$$P = \int_{port} \frac{p_{rms}^2}{\rho c} dl \quad (12)$$

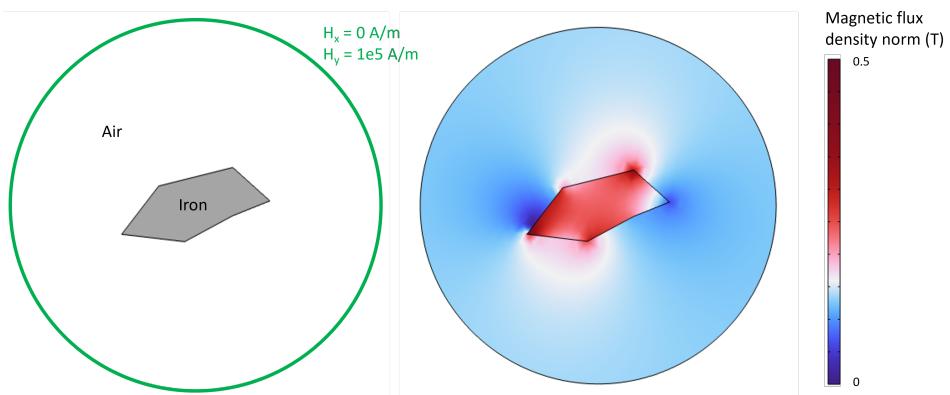
$$R = \frac{\log_{10}(P_2) - \log_{10}(P_3)}{0.292}$$

794 We use a value of 0.292 on the denominator of Eq. 12 to normalize the reward. And Fig. 6 shows  
795 a symmetric design with 7 circular cavities in the computation domain, producing equal acoustic  
796 pressure at the two output ports and thus  $R = 0$ . Notice that LLM is not limited by the circular  
797 cavity pattern, and is prompted to freely explore any viable cavity geometries within the computation  
798 domain.

799 B.2.2 MAGNETIC TORQUE  
800

801 This task aims to design the geometry of an iron core that generates large torque when subjected  
802 to a uniform magnetic field. Fig. 7 shows the problem setting, an example iron core geometry  
803 and the corresponding magnetic flux density norm field. A uniform magnetic field intensity of  
804  $\mathbf{H} = [0, 1e5]$  A/m is applied to the circular boundary. The iron core possesses a large permeability  
805  $\mu \gg \mu_0$  distorts the magnetic flux density field  $\mathbf{B}$  within the circular air domain. The distorted  $\mathbf{B}$   
806 thus applies a torque on the iron core, which can be obtained from Comsol by solving the static  
807 Maxwell's equations.

808 To guide the model reinforcement learning and evolutionary search, the following reward  $R$  is com-  
809 puted as below where  $\mathbf{T}$  is Maxwell stress tensor,  $\mathbf{r}$  is position vector, and  $\tau$  represents magnetic  
torque which is simplified to  $\tau_z$  in 2D simulations:



823  
824 Figure 7: The magnetic flux density field generated by an iron core subject to a uniform magnetic  
825 field boundary condition. The distorted magnetic flux density field then applies a torque on the iron  
826 core.  
827  
828

$$829 \quad \mathbf{T} = \frac{1}{\mu_0} (\mathbf{B} \mathbf{B} - \frac{1}{2} B^2 \mathbf{I}) \\ 830 \quad R = \frac{||\boldsymbol{\tau}||}{9241.99 \cdot A} = \frac{1}{9241.99 \cdot A} \left\| \int_S \mathbf{r} \times (\mathbf{T} \cdot \hat{\mathbf{n}}) dA \right\| \quad (13)$$

$$831 \\ 832 \\ 833$$

834  
835 We use a value of 9241.99 on the denominator of Eq. 13 to normalize the reward. Notice that a  
836 perfectly symmetric iron core (for instance a circle) would have  $\tau_z = 0$ . Therefore, we expect to  
837 train and evolve the LLM to produce a highly irregular iron core geometry to generate large magnetic  
838 torque values. We set a minimum area of  $2e^{-4} \text{ m}^2$  to avoid naive designs.  
839  
840

### B.2.3 BEAM BENDING

841 This task aims to design the cross section geometry of a cantilever beam subject to a superposed  
842 loading of bending moments  $M_x$  and  $M_y$ , shear forces  $T_x$  and  $T_y$  along the two in-plane directions,  
843 and twisting moment  $T_z$  along the out-of-plane direction. The cantilever beam is assumed to be  
844 linear elastic with Young's modulus 1 GPa and Poisson's ratio 0.3. Fig. 8 shows an example beam  
845 cross section design and the von Mises stress distribution as calculated from Eq. 14, solved using  
846 the Beam Cross Section module in Comsol. As the cross section stays in the x-y plane,  $\sigma_{xx}$ ,  $\sigma_{yy}$ ,  
847 and  $\tau_{xy}$  take 0 values.  
848

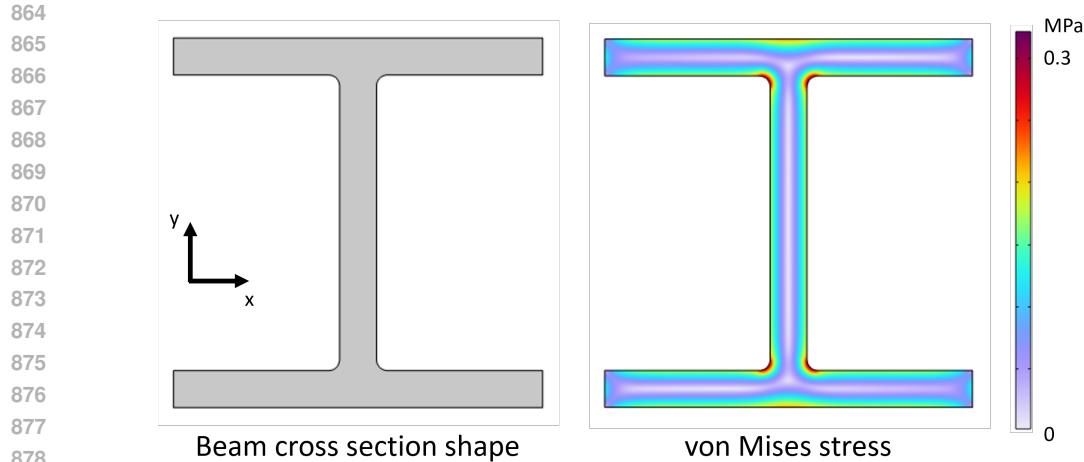
$$849 \quad \sigma_{vm} = \sqrt{\frac{1}{2} [(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{xx} - \sigma_{zz})^2 + (\sigma_{yy} - \sigma_{zz})^2] + 3 \cdot (\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2)} \quad (14)$$

$$850 \\ 851$$

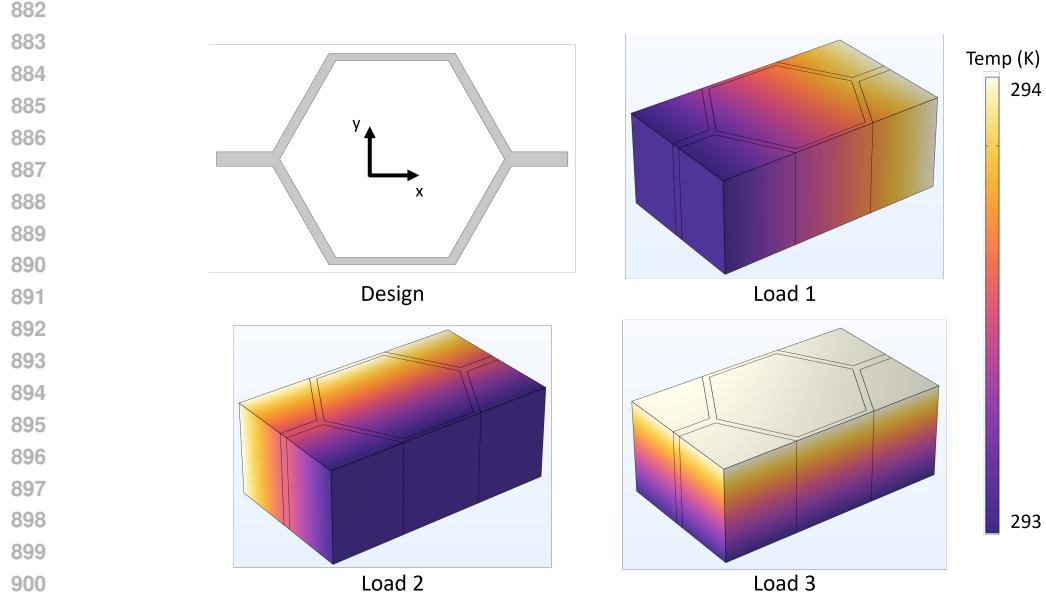
852 The reward is set to be  $R = \frac{I_1^{0.8} \cdot I_2^{0.2}}{1.32e^{-3} \cdot A}$  where  $A$  is the cross section area,  $I_1$  is the largest second  
853 moment of inertia,  $I_2$  is the smallest second moment of inertia. We use a value of  $1.32e^{-3}$  on the  
854 denominator to normalize the reward.  $I_1$  and  $I_2$  represent the beam's largest and smallest resistance  
855 over different bending loading directions, and can be calculated from the stress field following the  
856 classical beam bending theory (Bauchau & Craig, 2009). We set a minimum area of  $2e^{-3} \text{ m}^2$  to  
857 avoid naive designs.  
858

### B.2.4 PERIODIC HEAT

859 This task aims to design the unit cell geometry of a periodic meta-material for best effective thermal  
860 conductivity. The base material is assumed to be aluminum with density  $2700 \text{ kg/m}^3$  and thermal  
861 conductivity  $238 \text{ W/mK}$ . Fig. 9 shows an example 2D unit cell geometry which will be extruded  
862 in the z direction to form the 3D unit cell. The resultant temperature distribution and effective  
863



880 Figure 8: The von Mises stress field generated by applying bending moment, shear force, and twisting  
881 moment on a cantilever beam cross section design.



902 Figure 9: Temperature distribution of the meta-material under three loading conditions. The effec-  
903 tive properties are calculated based on temperature distributions according to the homogenization  
904 theory.

905  
906  
907 properties are solved using Comsol based on the homogenization theory. The results are calculated  
908 from a 1 K temperature difference boundary conditions along x, y, and z directions.

909  
910  
911  
912

$$R = \frac{\text{trace}(\mathbf{k}_{eff})}{0.178 \cdot \rho_{eff}} \quad (15)$$

913

914 where  $\mathbf{k}_{eff}$  is the homogenized effective thermal conductivity matrix, and  $\rho_{eff}$  is the effective  
915 density, which simply equals to the percentage of volume filled by aluminum. We use a value of  
916 0.178 on the denominator to normalize the reward. This objective function targets to maximize the  
917 thermal conductivity along x, y, and z directions under limited material usage. We set a maximum  
918 effective density  $\rho_{eff} \leq 2000 \text{ kg/m}^3$  to avoid naive designs.

918  
919

## B.2.5 INDUCTOR

920  
921  
922  
923  
924  
925  
926  
927

This task aims to design an inductor which is a critical component in power electronics. Fig. 10 shows an example inductor consisting of an iron core and coil windings in a cylindrical coordinate. A sinusoidal current excitation is supplied to the coils at a frequency of 1000 Hz and magnitude 500 A. The iron core possesses a nonlinear magnetization curve with an initial permeability of 663 H/m and saturates at 5 T. The resultant magnetic field is calculated using Comsol by solving the Maxwell's equations in frequency domain. The model is asked to propose the optimal iron core geometry as well as the placement of the coil windings (coil shapes are fixed) to produce the maximum inductance with limited material usage.

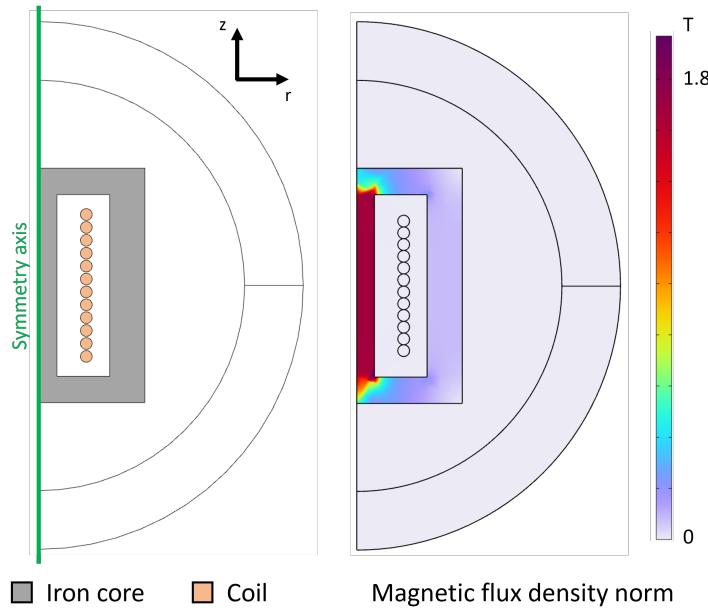
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948949  
950  
951  
952

Figure 10: The magnetic flux density norm field generated by an inductor. The copper coils are excited by a 500 A, 1000 Hz sinusoidal current.

953  
954  
955

$$R = \frac{L}{43.11 \cdot V} = \frac{0.5 \cdot \int_{\Omega} (B_r \cdot \bar{H}_r + B_{\phi} \cdot \bar{H}_{\phi} + B_z \cdot \bar{H}_z) dV}{43.11 \cdot V} \quad (16)$$

956  
957  
958  
959  
960  
961  
962  
963

The reward calculation is shown in Eq. 16 where  $B_r$ ,  $B_{\phi}$ , and  $B_z$  are cylindrical components of magnetic flux density field, and  $H_r$ ,  $H_{\phi}$ ,  $H_z$  are components of magnetic intensity field. Both fields take complex values for frequency domain response. We use a value of 43.11 on the denominator to normalize the reward. The numerator stands for the inductance which is a volume integral of magnetic energy. We set a minimum iron core volume of  $1e^{-3} m^3$  to avoid naive designs.

## B.3 CIRCLE PACKING

964  
965  
966  
967  
968  
969  
970  
971

The objective of these tasks is to pack a fixed number of circles in a specific domain and maximize the sum of the radii of these circles. The circles cannot overlap with each other or exceed the domain boundary. All the centers and radii can change as long as the constraints are satisfied.

Formally, let  $n = 26$  be the number of circles,  $\{x_i\}_{i \leq n}$ ,  $\{y_i\}_{i \leq n}$  be the coordinates of centers and  $\{r_i\}_{i \leq n}$  be the radii. The objective can be written as:

$$R = \sum_{i=1}^n r_i, \quad (17)$$

972 while the constraint is  
 973

$$\begin{aligned}
 974 \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} &\geq r_i + r_j, \quad \forall 1 \leq i < j \leq n \\
 975 \quad x_i - r_i &\geq 0, \quad \forall 1 \leq i \leq n \\
 976 \quad x_i + r_i &\leq 1, \quad \forall 1 \leq i \leq n \\
 977 \quad y_i - r_i &\geq 0, \quad \forall 1 \leq i \leq n \\
 978 \quad y_i + r_i &\leq 1, \quad \forall 1 \leq i \leq n,
 \end{aligned} \tag{18}$$

980 for the packing in a unit square, and  
 981

$$\begin{aligned}
 982 \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} &\geq r_i + r_j, \quad \forall 1 \leq i < j \leq n \\
 983 \quad \sqrt{x_i^2 + y_i^2} + r_i &\leq 1, \quad \forall 1 \leq i \leq n,
 \end{aligned} \tag{19}$$

985 for the packing in a unit disk.  
 986

#### 987 B.4 FUNCTION MINIMIZATION 988

989 These tasks require the model to find an effective algorithm to locate the global minimum of a  
 990 complex function with various local minima. For a given function  $f(\mathbf{x}^*)$  and the model's prediction  
 991  $\hat{\mathbf{x}}^*$ , The evaluation metric is defined as:

$$992 \quad R = \frac{|f(\mathbf{x}^*)|}{|f(\mathbf{x}^*)| + |f(\hat{\mathbf{x}}^*) - f(\mathbf{x}^*)|}. \tag{20}$$

996 This metric is suitable for distinct functions with varying scales of  $|f(\mathbf{x}^*)|$ . It satisfies  $0 \leq R \leq 1$   
 997 and if the model successfully finds the global minimum, the reward will be  $R = 1.0$ .  
 998

##### 999 B.4.1 EGHOLDER FUNCTION

1000 The Eggholder function is a classical task for evaluating evolutionary optimization algorithms with  
 1001 various local minima. It can be defined as:

$$1003 \quad f(\mathbf{x}) = -(x_2 + 47) \sin(\sqrt{|(x_2 + 47) + \frac{x_1}{2}|}) - x_1 \sin(\sqrt{|x_1 - (x_2 + 47)|}), \tag{21}$$

1004 with constraint  $-512 \leq x_1, x_2 \leq 512$  and a global minimum  $f((512, 404.2319)) \approx -959.6407$   
 1005 under such constraint.  
 1006

1007 Figure 11 illustrates the landscape and the global minimum point of the Eggholder function.  
 1008

##### 1009 B.4.2 MISHRA'S BIRD FUNCTION

1010 The Mishra's Bird function is a classic test function used in optimization to evaluate the performance  
 1011 of algorithms. It is known for having a unique "bird-shaped" landscape with multiple local minima  
 1012 and a single global minimum. It's often used to test an algorithm's ability to avoid getting stuck in  
 1013 suboptimal solutions.  
 1014

1015 The function is defined as:

$$1017 \quad f(\mathbf{x}) = \sin(x_2)e^{(1-\cos(x_1))^2} + \cos(x_1)e^{(1-\sin(x_2))^2} + (x_1 - x_2)^2 \tag{22}$$

1019 with the constraints:

$$\begin{aligned}
 1020 \quad -10 \leq x_1 &\leq 0 \\
 1021 \quad -6.5 \leq x_2 &\leq 0 \\
 1022 \quad x_1^2 + x_2^2 &\geq 25.
 \end{aligned} \tag{23}$$

1024 The global minimum is  $f((-3.1302, -1.5822)) \approx -106.7645$ .  
 1025

Figure 12 shows the landscape of the Mishra's Bird function and its global minimum point.

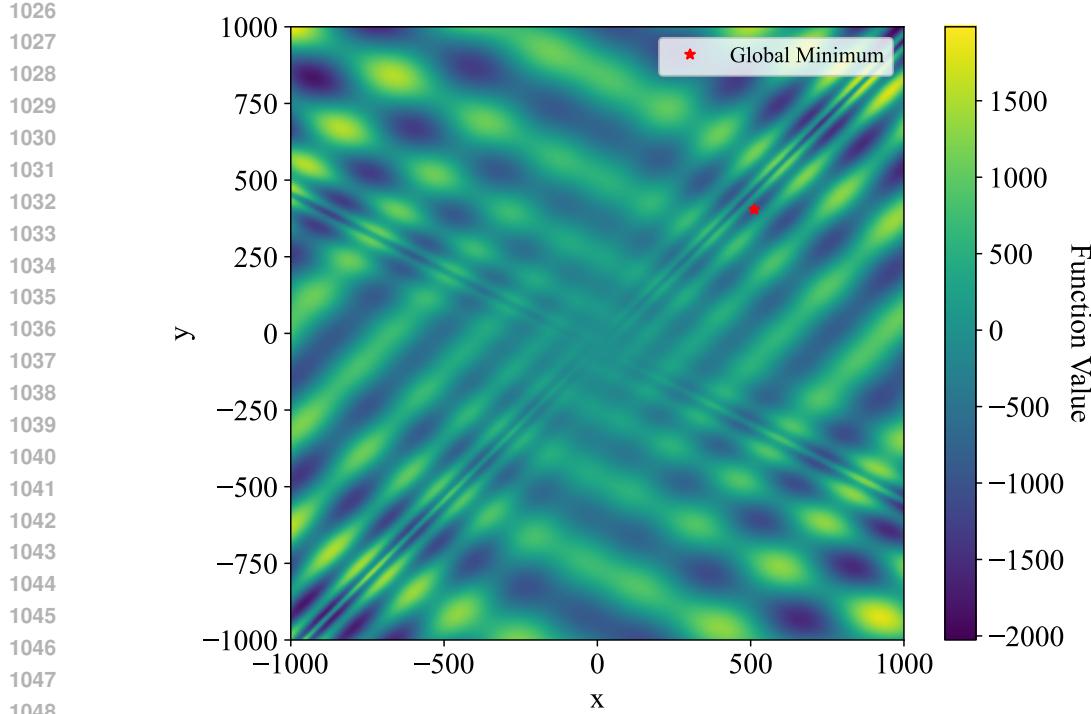


Figure 11: The landscape and global minimum point of Eggholder function with constraints  $-512 \leq x, y \leq 512$

#### B.4.3 KEANES BUMP FUNCTION

The Keanes Bump function is a challenging, non-convex test function commonly used to evaluate the performance of optimization algorithms in handling high-dimensional problems with complex constraints. The function's landscape is highly irregular, containing numerous local minima, and its feasible region is a small, irregular subset of the search space.

Let  $d$  be the dimension of variables and  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ , the function is defined as:

$$f(\mathbf{x}) = \frac{-|\sum_{i=1}^d \cos^4(x_i) - 2 \prod_{i=1}^d \cos^2(x_i)|}{\sqrt{\sum_{i=1}^d i x_i^2}} \quad (24)$$

with the following constraints:

$$\begin{aligned} 1065 \quad 0 < x_i &\leq 10, \quad \forall 1 \leq i \leq d \\ 1066 \quad \sum_{i=1}^d x_i &\leq 7.5d \\ 1067 \quad \prod_{i=1}^d x_i &\geq 0.75. \end{aligned} \quad (25)$$

The global minimum is located within the feasible region, which is a small, bounded area defined by these constraints. The image in this document, Figure 13, shows a two-dimensional visualization of the function's landscape. However, for our experiments, we tested the function in its 10-D, 20-D, and 30-D versions, where the complexity increases significantly. The global minima and their corresponding function values for these dimensions are listed below.

- **10-D Version:** The global minimum value is approximately  $-0.747310362$ .
- **20-D Version:** The global minimum value is approximately  $-0.803619104$ .
- **30-D Version:** The global minimum value is approximately  $-0.818056222$ .

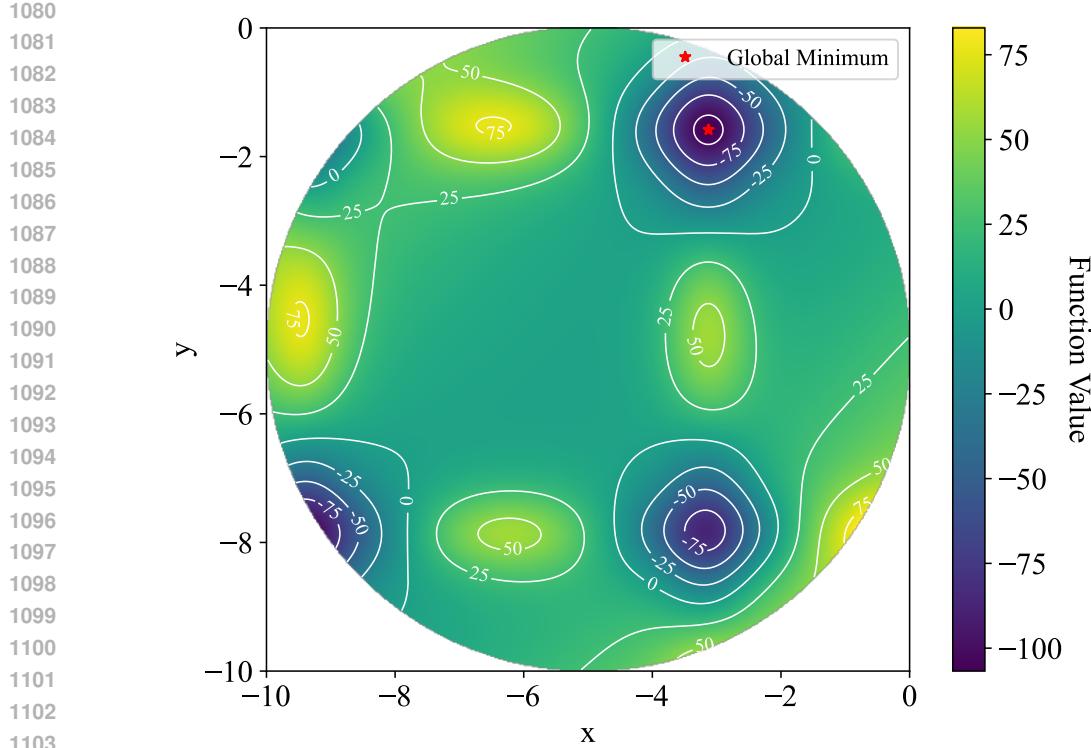


Figure 12: The landscape and global minimum point of Mishra’s Bird function with constraints  $x_1 \in [-10, 0]$ ,  $x_2 \in [-6.5, 0]$  and  $x_1^2 + x_2^2 \geq 25$ .

## B.5 SYMBOLIC REGRESSION

In this task, the model has to uncover symbolic mathematical expressions from observational data. The benchmark and baselines are provided by Shojaee et al. (2025), which includes equations and data in chemistry, biology, physics and material science domains. In each category, several cases are created, each containing its own train and test sets generated by the same underlying equation. The model trained on the train set has to propose an expression to minimize the normalized mean square error (NMSE) on the test set, which is defined as:

$$\text{NMSE} = \frac{\sum_{i=1}^N (\hat{y}_i - y_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}, \quad (26)$$

where  $N$  is the number of observations in the test set.

To ensure a fair and robust comparison with the benchmark paper’s results, we use the median of the NMSE calculated across all tasks within the same category  $c$ :

$$\text{NMSE}_c = \text{median}(\text{NMSE}_{c,1}, \text{NMSE}_{c,2}, \dots, \text{NMSE}_{c,n}). \quad (27)$$

The reward we used for reinforcement learning for category  $c$  is then set to:

$$R_c = -\log_{10}(\text{NMSE}_c). \quad (28)$$

In the benchmark, all the methods have a limit of 1000 trials for each single case, and we obey the same rule in our experiments, adjusting the number of training steps accordingly.

## C DESCRIPTION OF TASK SPECIFIC BASELINES

In this section, we introduce the task-specific baseline methods and describe their implementation details.

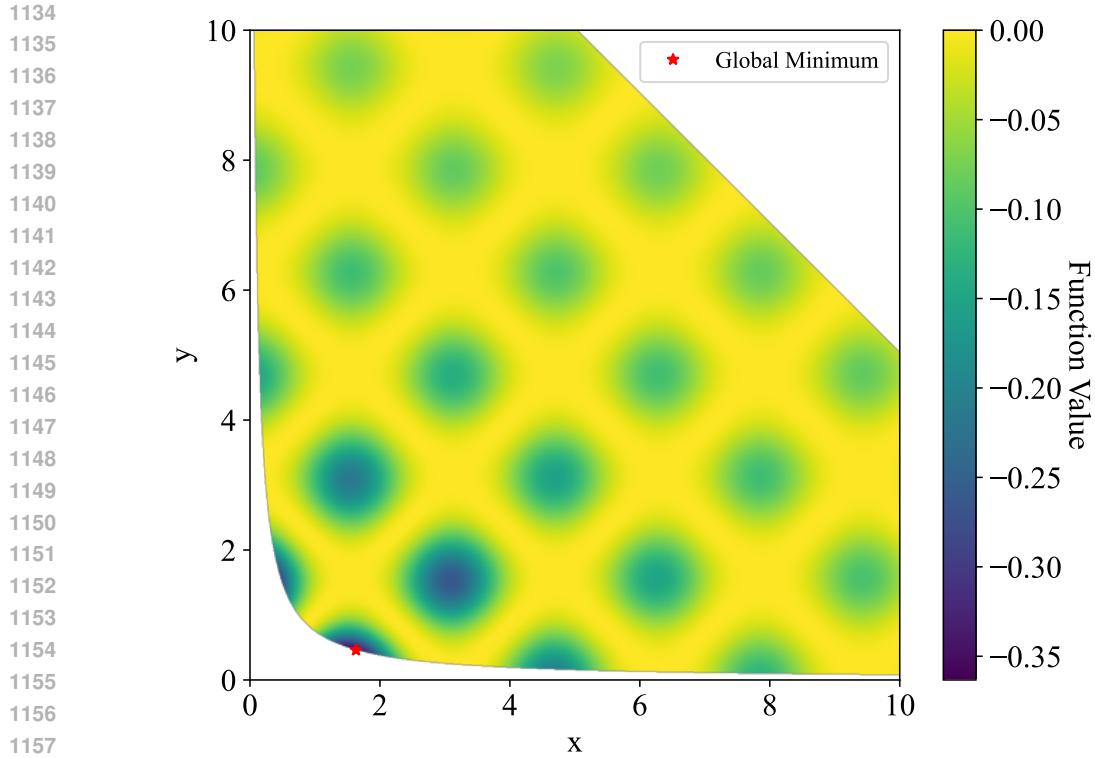


Figure 13: The landscape and global minimum point of the 2-D Kanes Bump function. The feasible region, a small part of the search space, is the only area with finite function values.

**Machine Learning.** For machine learning benchmarks, we evaluate two interpretable yet competitive models: LightGBM (Ke et al., 2017), a gradient boosting framework widely adopted in practice, and Rule-based Representation Learner (RRL) (Wang et al., 2021), which learns discrete non-fuzzy rules via gradient grafting to achieve both scalability and interpretability.

**Physics Simulation.** For physics-related optimization problems, we use two widely adopted modules in COMSOL Multiphysics: parameter search and topology optimization. For parameter search, we first parameterize the geometry based on initial solutions provided by human experts, and then optimize within the search space defined by these parameters. For topology optimization, human experts specify deformable geometric regions, while COMSOL applies its built-in topology optimization solvers to iteratively refine the structure.

**Circle Packing.** We consider two strong baselines: Sequential Least Squares Programming (SLSQP) (Lawson & Hanson, 1995) and a Genetic Algorithm (GA). SLSQP formulates circle packing as a constrained optimization problem, maximizing the sum of radii subject to boundary and non-overlap constraints. The GA baseline encodes circle positions and radii, evolves a feasible population with selection, crossover, and mutation, and evaluates fitness by the total radii.

**Function Minimization.** We adopt two standard constrained optimization solvers from `scipy.optimize`: Sequential Least Squares Programming (SLSQP) and the `trust-constr` method (Conn et al., 2000). Both are widely used gradient-based methods that provide strong task-specific baselines for function minimization.

**Symbolic Regression.** For symbolic regression tasks, we directly use results reported in LLM-SRBench (Shojaee et al., 2025), obtained by GPT-4o-mini running two recent methods: LaSR (Grayeli et al., 2024), which enhances evolutionary search with LLM-guided concept discovery, and

1188 LLM-SR (Shojaee et al., 2024), which combines LLM scientific priors with evolutionary equation  
 1189 search. These represent competitive state-of-the-art baselines for symbolic regression.  
 1190

## D EXAMPLE OF PROMPTS USED IN EACH EXPERIMENT

1194 In this section, we will demonstrate the prompts we used in our experiments.  
 1195

### D.1 MACHINE LEARNING

#### Prompt for task Adult Income

1199 You are an expert software developer tasked with iteratively improving  
 1200 ↳ a codebase.  
 1201 Your job is to analyze the current program and suggest improvements  
 1202 ↳ based on feedback from previous attempts.  
 1203 Focus on making targeted changes that will increase the program's  
 1204 ↳ performance metrics.  
 1205 Respond in the following format: <think>  
 1206 ...  
 1207 </think>  
 1208 <answer>  
 1209 ...  
 1210 </answer>.  
**# Problem Description**  
 1211 You are an expert in traditional machine learning.  
 1212 Your task is to build a predictive model using the **\*\*Adult Income**  
 1213 ↳ **Dataset\*\*** (also known as the "Census Income" dataset).  
 1214 This dataset contains demographic and employment-related attributes  
 1215 ↳ collected from the 1994 U.S. Census database.  
 1216 The goal is to **\*\*predict whether a client will subscribe to a term**  
 1217 ↳ **deposit\*\*** (`y` column: yes/no) based on demographic and  
 1218 ↳ marketing-related features.  
 1219 The goal is to **\*\*predict whether a person's income exceeds \\$50K per**  
 1220 ↳ **year\*\*** (income column: >50K / <=50K) based on individual and  
 1221 ↳ employment features.  
**## Dataset Features**  
**### Target variable:**  
 1223 - **\*\*income\*\***: `>50K`, `<=50K`. (Parsed to 1/0 in Program)  
**### Input variables:**  
 1226 1. **\*\*age\*\*** \*(numeric)\*  
 1227 Age of the individual.  
 1228 2. **\*\*workclass\*\*** \*(categorical)\*  
 1229 Type of employment:  
 1230 ↳ `Private`, `Self-emp-not-inc`, `Self-emp-inc`, `Federal-gov`,  
 1231 ↳ `Local-gov`,  
 1232 ↳ `State-gov`, `Without-pay`, `Never-worked`.  
 1233 3. **\*\*fnlwgt\*\*** \*(numeric)\*  
 1234 Final sampling weight | indicates the number of people represented  
 1235 ↳ by this record.  
 1236 4. **\*\*education\*\*** \*(categorical)\*  
 1237 Education level:  
 1238 ↳ `Bachelors`, `Some-college`, `11th`, `HS-grad`, `Prof-school`,  
 1239 ↳ `Assoc-acdm`, `Assoc-voc`, `9th`, `7th-8th`, `12th`, `Masters`,  
 1240 ↳ `1st-4th`, `10th`, `Doctorate`, `5th-6th`, `Preschool`.  
 1241

```

1242
1243
1244 5. **education-num** *(numeric)*
1245   Encodes years of education.
1246
1247 6. **marital-status** *(categorical)*
1248   `Married-civ-spouse`, `Divorced`, `Never-married`, `Separated`,
1249   `Widowed`, `Married-spouse-absent`, `Married-AF-spouse`.
1250
1251 7. **occupation** *(categorical)*
1252   `Tech-support`, `Craft-repair`, `Other-service`, `Sales`,
1253   `→ Exec-managerial`,
1254   `Prof-specialty`, `Handlers-cleaners`, `Machine-op-inspct`,
1255   `→ Adm-clerical`,
1256   `Farming-fishing`, `Transport-moving`, `Priv-house-serv`,
1257   `→ Protective-serv`,
1258   `Armed-Forces`.
1259
1260 8. **relationship** *(categorical)*
1261   `Wife`, `Own-child`, `Husband`, `Not-in-family`, `Other-relative`,
1262   `→ Unmarried`.
1263
1264 9. **race** *(categorical)*
1265   `White`, `Asian-Pac-Islander`, `Amer-Indian-Eskimo`, `Other`,
1266   `→ Black`.
1267
1268 10. **sex** *(categorical)*
1269   `Female`, `Male`.
1270
1271 11. **capital-gain** *(numeric)*
1272   Income from capital gains.
1273
1274 12. **capital-loss** *(numeric)*
1275   Losses from capital assets.
1276
1277 13. **hours-per-week** *(numeric)*
1278   Number of working hours per week.
1279
1280 14. **native-country** *(categorical)*
1281   `United-States`, `Cambodia`, `England`, `Puerto-Rico`, `Canada`,
1282   `→ Germany`,
1283   `Outlying-US(Guam-USVI-etc)`, `India`, `Japan`, `Greece`, `South`,
1284   `→ China`,
1285   `Cuba`, `Iran`, `Honduras`, `Philippines`, `Italy`, `Poland`,
1286   `→ Jamaica`,
1287   `Vietnam`, `Mexico`, `Portugal`, `Ireland`, `France`,
1288   `→ Dominican-Republic`,
1289   `Laos`, `Ecuador`, `Taiwan`, `Haiti`, `Columbia`, `Hungary`,
1290   `→ Guatemala`,
1291   `Nicaragua`, `Scotland`, `Thailand`, `Yugoslavia`, `El-Salvador`,
1292   `Trinidad&Tobago`, `Peru`, `Hong`, `Holand-Netherlands`.
1293
1294 ## Additional Notes
1295 - You may add, delete, or modify functions arbitrarily, but the
  → program must still contain the run_model() function.
- If you want to use new packages, please import them explicitly.
- Try different **data preprocessing**, **feature engineering**, and
  → **modeling techniques** to improve performance.
- Pay attention to Missing values: represented as "?". Handle them
  → properly.
- Pay attention to Categorical encoding: many features are
  → categorical; choose an effective encoding strategy.
1296
1297 ## Task

```

```

1296
1297
1298 Write a machine learning model to **predict** whether a person's
1299 → income exceeds \$50K.
1300
1301 You will be given a **starter program** in Python.
1302 Your goal is to **improve this program** to maximize the **F1-score**
1303 → on the test set.
1304
1305 Your code execution time should **not exceed 60 seconds**.
1306 You MUST use the exact SEARCH/REPLACE diff format shown below when
1307 → modifying code:
1308
1309 <<<<< SEARCH
1310 # Original code to find and replace (must match exactly)
1311 =====
1312 # New replacement code
1313 >>>>> REPLACE
1314
1315 ## Current Program
1316 Status: {current_status}
1317 ~~`python
1318 {current_program}
1319 ~~`
```

#### Prompt for task Bank Marketing

```

1320
1321 You are an expert software developer tasked with iteratively improving
1322 → a codebase.
1323 Your job is to analyze the current program and suggest improvements
1324 → based on feedback from previous attempts.
1325 Focus on making targeted changes that will increase the program's
1326 → performance metrics.
1327 Respond in the following format: <think>
1328 ...
1329 </think>
1330 <answer>
1331 ...
1332 </answer>.
1333 # Problem Description
1334
1335 You are an expert in traditional machine learning.
1336 Your task is to build a predictive model using the **Bank Marketing
1337 → Dataset**.
1338 This dataset contains information collected from direct marketing
1339 → campaigns conducted by a Portuguese banking institution.
1340 The goal is to **predict whether a client will subscribe to a term
1341 → deposit** ('y' column: yes/no) based on demographic and
1342 → marketing-related features.
1343
1344 ## Dataset Features
1345 Input variables:
1346 ### bank client data:
1347 1 - age (numeric)
1348 2 - job : type of job (categorical: "admin.", "blue-collar",
1349 → "entrepreneur", "housemaid", "management", "retired",
1350 → "self-employed", "services", "student", "technician",
1351 → "unemployed", "unknown")
1352 3 - marital : marital status (categorical: "divorced", "married",
1353 → "single", "unknown"; note: "divorced" means divorced or
1354 → widowed)
```

```

1350
1351     4 - education (categorical: "basic.4y", "basic.6y", "basic.9y",
1352         ↳ "high.school", "illiterate", "professional.course",
1353         ↳ "university.degree", "unknown")
1354     5 - default: has credit in default? (categorical:
1355         ↳ "no", "yes", "unknown")
1356     6 - housing: has housing loan? (categorical: "no", "yes", "unknown")
1357     7 - loan: has personal loan? (categorical: "no", "yes", "unknown")
1358 #### related with the last contact of the current campaign:
1359     8 - contact: contact communication type (categorical:
1360         ↳ "cellular", "telephone")
1361     9 - month: last contact month of year (categorical: "jan", "feb",
1362         ↳ "mar", ..., "nov", "dec")
1363    10 - day_of_week: last contact day of the week (categorical:
1364         ↳ "mon", "tue", "wed", "thu", "fri")
1365    11 - duration: last contact duration, in seconds (numeric).
1366 #### other attributes:
1367    12 - campaign: number of contacts performed during this campaign and
1368        ↳ for this client (numeric, includes last contact)
1369    13 - pdays: number of days that passed by after the client was last
1370        ↳ contacted from a previous campaign (numeric; 999 means client
1371        ↳ was not previously contacted)
1372    14 - previous: number of contacts performed before this campaign and
1373        ↳ for this client (numeric)
1374    15 - poutcome: outcome of the previous marketing campaign
1375        ↳ (categorical: "failure", "nonexistent", "success")
1376 #### social and economic context attributes
1377    16 - emp.var.rate: employment variation rate - quarterly indicator
1378        ↳ (numeric)
1379    17 - cons.price.idx: consumer price index - monthly indicator
1380        ↳ (numeric)
1381    18 - cons.conf.idx: consumer confidence index - monthly indicator
1382        ↳ (numeric)
1383    19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
1384    20 - nr.employed: number of employees - quarterly indicator
1385        ↳ (numeric)
1386
1387 ## Task
1388
1389 Write a machine learning model to **predict** whether a client
1390     ↳ subscribes to a term deposit.
1391
1392 You will be given a **starter program** in Python.
1393 Your goal is to **improve this program** to maximize the **F1-score**
1394     ↳ on the test set.
1395
1396 Your code execution time should **not exceed 60 seconds**.
1397 You MUST use the exact SEARCH/REPLACE diff format shown below when
1398     ↳ modifying code:
1399
1400 <<<<< SEARCH
1401 # Original code to find and replace (must match exactly)
1402 =====
1403 # New replacement code
1404 >>>>> REPLACE
1405
1406
1407 ## Additional Notes
1408 - You may add, delete, or modify functions arbitrarily, but the
1409     ↳ program must still contain the run_model() function.
1410 - If you want to use new packages, please import them explicitly.
1411 - Try different **data preprocessing**, **feature engineering**, and
1412     ↳ **modeling techniques** to improve performance.
1413
1414 ## Current Program

```

```

1404
1405     Status: {current_status}
1406     ````python
1407     {current_program}
1408     ````

1409
1410 Prompt for task Boston Housing
1411
1412     You are an expert software developer tasked with iteratively improving
1413     → a codebase.
1414     Your job is to analyze the current program and suggest improvements
1415     → based on feedback from previous attempts.
1416     Focus on making targeted changes that will increase the program's
1417     → performance metrics.
1418     Respond in the following format: <think>
1419     ...
1420     </think>
1421     <answer>
1422     ...
1423     </answer>.

# Problem Description
1424
1425     You are an expert in traditional machine learning.
1426     Your task is to build a predictive regression model using the
1427     → Boston Housing Dataset.
1428
1429     The Boston Housing Dataset contains information collected by the
1430     → U.S. Census Service concerning housing in the Boston,
1431     → Massachusetts area.
1432     The goal is to predict the median value of owner-occupied homes
1433     → (MEDV, measured in \$1000s) based on various demographic,
1434     → economic, and geographic factors.

## Dataset Features
1435
1436     The dataset contains 13 numerical and categorical features. Some
1437     → of them may have missing values (nan in dataframe)
1438
1439     1. CRIM { Per capita crime rate by town
1440     2. ZN { Proportion of residential land zoned for lots over 25,000
1441     → sq.ft.
1442     3. INDUS { Proportion of non-retail business acres per town
1443     4. CHAS { Charles River dummy variable (1 if tract bounds river; 0
1444     → otherwise)
1445     5. NOX { Nitric oxides concentration (parts per 10 million)
1446     6. RM { Average number of rooms per dwelling
1447     7. AGE { Proportion of owner-occupied units built prior to 1940
1448     8. DIS { Weighted distances to five Boston employment centres
1449     9. RAD { Index of accessibility to radial highways
1450     10. TAX { Full-value property tax rate per \$10,000
1451     11. PTRATIO { Pupil-teacher ratio by town
1452     12. LSTAT { Percentage of lower status population
1453     13. MEDV { Target variable: Median value of owner-occupied
1454     → homes in \$1000s

## Task
1455
1456     You will be provided with a starter Python program.
1457     Your objective is to improve the program to build a more accurate
1458     → regression model for predicting MEDV.
1459     Your improvements should focus on maximizing the RMSE score on the
1460     → test set (RMSE score =  $2 - \log_{10}(\text{RMSE})$ ).

```

```

1458
1459 ## Requirements
1460
1461 * Your code execution time **must not exceed 60 seconds**.
1462 * You MUST use the **SEARCH/REPLACE diff format** exactly as shown
1463 → below when modifying the code:
1464
1465 ```
1466 <<<<< SEARCH
1467 # Original code to find and replace (must match exactly)
1468 =====
1469 # New replacement code
1470 >>>>> REPLACE
1471 ```
1472
1473 ## Additional Notes
1474
1475 * You **may add, delete, or modify functions** as needed, but the
1476 → program **must still contain** the run_model() function.
1477 * If you want to use new packages, please import them explicitly.
1478 → Usable packages: pandas, numpy, sklearn, scipy, statsmodels,
1479 → xgboost, lightgbm, catboost, category_encoders, imbalanced-learn
1480 * Try different **data preprocessing**, **feature engineering**, and
1481 → **modeling techniques** to improve performance.
1482
1483 ## Current Program
1484 Status: {current_status}
1485 ```python
1486 {current_program}
1487 ```
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497 prompt for task Predict Transparent Conductors

```

```

1498 You are an expert software developer tasked with iteratively improving
1499 → a codebase.
1500 Your job is to analyze the current program and suggest improvements
1501 → based on feedback from previous attempts.
1502 Focus on making targeted changes that will increase the program's
1503 → performance metrics.
1504 Respond in the following format: <think>
1505 ...
1506 </think>
1507 <answer>
1508 ...
1509 </answer>.
1510 # Overview
1511
1512 ## Description
1513

```

1512  
 1513 Innovative materials design is needed to tackle some of the most  
 1514 → important health, environmental, energy, social, and economic  
 1515 → challenges of this century. In particular, improving the  
 1516 → properties of materials that are intrinsically connected to the  
 1517 → generation and utilization of energy is crucial if we are to  
 1518 → mitigate environmental damage due to a growing global demand.  
 1519 → Transparent conductors are an important class of compounds that  
 1520 → are both electrically conductive and have a low absorption in the  
 1521 → visible range, which are typically competing properties. A  
 1522 → combination of both of these characteristics is key for the  
 1523 → operation of a variety of technological devices such as  
 1524 → photovoltaic cells, light-emitting diodes for flat-panel displays,  
 1525 → transistors, sensors, touch screens, and lasers. However, only a  
 1526 → small number of compounds are currently known to display both  
 1527 → transparency and conductivity suitable enough to be used as  
 1528 → transparent conducting materials.  
 1529  
 1530 Aluminum (Al), gallium (Ga), indium (In) sesquioxides are some of the  
 1531 → most promising transparent conductors because of a combination of  
 1532 → both large bandgap energies, which leads to optical transparency  
 1533 → over the visible range, and high conductivities. These materials  
 1534 → are also chemically stable and relatively inexpensive to produce.  
 1535 → Alloying of these binary compounds in ternary or quaternary  
 1536 → mixtures could enable the design of a new material at a specific  
 1537 → composition with improved properties over what is current  
 1538 → possible. These alloys are described by the formula  $(Al_x Ga_y$   
 1539 →  $In_z)_{\{2N\}}O_{\{3N\}}$ ; where x, y, and z can vary but are limited  
 1540 → by the constraint  $x+y+z = 1$ . The total number of atoms in the unit  
 1541 → cell,  $N_{\{total\}}=2N+3N$  (where N is an integer), is typically  
 1542 → between 5 and 100. However, the main limitation in the design of  
 1543 → compounds is that identification and discovery of novel materials  
 1544 → for targeted applications requires an examination of enormous  
 1545 → compositional and configurational degrees of freedom (i.e., many  
 1546 → combinations of x, y, and z). To avoid costly and inefficient  
 1547 → trial-and-error of synthetic routes, computational data-driven  
 1548 → methods can be used to guide the discovery of potentially more  
 1549 → efficient materials to aid in the development of advanced (or  
 1550 → totally new) technologies. In computational material science, the  
 1551 → standard tool for computing these properties is the  
 1552 → quantum-mechanical method known as density-functional theory  
 1553 → (DFT). However, DFT calculations are expensive, requiring hundreds  
 1554 → or thousands of CPU hours on supercomputers for large systems,  
 1555 → which prohibits the modeling of a sizable number of possible  
 1556 → compositions and configurations. As a result, potential  $(Al_x$   
 1557 →  $Ga_y In_z)_{\{2N\}}O_{\{3N\}}$  materials remain relatively unexplored.  
 1558 → Data-driven models offer an alternative approach to efficiently  
 1559 → search for new possible compounds in targeted applications but at  
 1560 → a significantly reduced computational cost.  
 1561  
 1562 This competition aims to accomplish this goal by asking participants  
 1563 → to develop or apply data analytics/data mining/machine-learning  
 1564 → models for the prediction of two target properties: the formation  
 1565 → energy (which is an indication of the stability of a new material)  
 1566 → and the bandgap energy (which is an indication of the potential  
 1567 → for transparency over the visible range) to facilitate the  
 1568 → discovery of new transparent conductors and allow for advancements  
 1569 → in the above-mentioned technologies.  
 1570  
 1571 **## Evaluation**  
 1572  
 1573 Submissions are evaluated on the column-wise root mean squared  
 1574 → logarithmic error.  
 1575



```

1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
```
<<<<< SEARCH
# Original code to find and replace (must match exactly)
=====
# New replacement code
>>>> REPLACE
```
## Current Program
Status: {current_status}
```python
{current_program}
```

```

## D.2 PHYSICS SIMULATION

### Prompt for task Inductor

You are a helpful AI Assistant that provides well-reasoned and detailed responses.

You first think about the reasoning process as an internal monologue and then provide the user with the answer.

Respond in the following format: <think>

...

</think>

<answer>

...

</answer>.

**## Task Description**

You are a helpful AI Assistant and scientist with strong physical background and wonderful geometric designing ideas.

You are asked to generate the geometry design of a component using yaml files under certain constraints. You will first create geometries of your design, and then assign functions to the geometries according to the specific requirements.

Your final answer should contain a yaml file enclosed in ```yaml\n(your code)```. The yaml file should have at least two parts: geometry and selection. The specific requirements are as follow:

1. **geometry:** A list of objects with type and type-specific parameters.  
The types and parameters are as follows:
  - Polygon: (2D) You can use it to create rectangles, triangles, etc.
    - table: Ordered list of n vertices as [x, y] points. The polygon is formed by **\*\*connecting consecutive points\*\***  $(p_i \rightarrow p_{i+1})$  and **\*\*automatically closing\*\*** the shape  $(p_n \rightarrow p_1)$ .
    - fillet: (Optional) A list of [i, r] tuples, where i is the index (starting from 1) of a polygon vertex defined in the above table, and r is the fillet radius for that corresponding vertex.
  - Ellipse: (2D) You can use it to create circles.
    - semiaxes: [horizontal, vertical] axis lengths
    - pos: [center\_x, center\_y] center position
    - rot: (Optional) Rotation angle (degree) counterclockwise
    - angle: (Optional) Angular span (degree) counterclockwise. e.g. by setting angle=180 you can draw a upward semicircle.
  - LineSegment: (1D)
    - coord1: [start\_x, start\_y]
    - coord2: [end\_x, end\_y]

```

1674
1675     CircularArc: (1D)
1676         r: Radius
1677         angle1: Start angle (degree) counterclockwise, 0 degree
1678         ↳ represent positive direction of X-axis.
1679         angle2: End angle (degree) counterclockwise
1680
1681     CubicBezier: (1D)
1682         p: Control points as [[x0,x1,x2,x3], [y0,y1,y2,y3]]
1683         w: Weight values as [w0,w1,w2,w3]
1684
1685     InterpolationCurve: (1D)
1686         table: Ordered list of [x,y] points to interpolate through.
1687         ↳ The curve will pass every points smoothly (polynomial
1688         ↳ interpolation for x and y).
1689
1690     ParametricCurve: (1D)
1691         parname: Name of parameter
1692         parmin: Minimum value of parameter
1693         parmax: Maximum value of parameter
1694         coord: Expressions about the parameter like ["expression_x",
1695             ↳ "expression_y"]. Trigonometric functions here use radians
1696
1697     ConvertToSolid: (2D) Geometry formed by end-to-end connected 1D
1698         ↳ curves.
1699         geometries: A dictionary of 1D geometries (using the same
1700             ↳ structure as the top-level geometry section, recursive).
1701             ↳ **They Must connect end-to-end and form a simply connected
1702             ↳ space**.
1703
1704     Union: (2D) Union of 2D geometries.
1705         geometries: A dictionary of geometries (recursive).
1706
1707     Intersection: (2D) Intersection of 2D geometries.
1708         geometries: A dictionary of geometries (recursive).
1709
1710     Difference: (2D) Difference of the 2D geometries.
1711         geometries_add: A dictionary of geometries to keep
1712             ↳ (recursive).
1713         geometries_subtract: A dictionary of geometries to subtract
1714             ↳ (recursive).
1715
1716     After **geometry** was created, the shapes will be splitted into
1717         ↳ **non-overlapping connected regions**.
1718         - Overlapping 2D shapes create new regions (e.g., two intersecting
1719             ↳ circles → 3 regions)
1720         - Enclosed 2D shapes split regions (e.g., circle inside polygon →
1721             ↳ 2 regions: circle interior + polygon-ring)
1722         - 1D curves through 2D shapes create sub-regions (e.g., line
1723             ↳ segment through rectangle → alternating regions)
1724
1725     The **regions** can be represented by the following ways:
1726         - point: You can select an interior point of the region to
1727             ↳ represent it. The point should never be on boundaries/corners.
1728             ↳ One point per region suffices.
1729         - geometry: The 2d shapes you created might be splitted into
1730             ↳ several regions. You can select the geometry to represent all
1731             ↳ the regions in it.
1732
1733     2. selection: After regions are created, you will assign different
1734         ↳ functions to regions using selections.
1735
1736     UnionSelection: Union of all the regions selected below.
1737         points: (Optional) List of [x,y] points representing distinct
1738             ↳ regions.
1739         geometries: (Optional) List of 2d geometry names you created
1740             ↳ above. By listing geometries here, you can select all the
1741             ↳ region this geometry contains.
1742         selections: (Optional) List of other selection names you
1743             ↳ created.
1744
1745     IntersectionSelection: Intersection of all the regions selected
1746         ↳ below.
1747         same parameters as UnionSelection

```

```

1728
1729     DifferenceSelection: Select the regions in Add but not in
1730     ↳ Subtract.
1731     add: same parameters as UnionSelection.
1732     subtract: same parameters as UnionSelection.
1733
1734     Finally a yaml file will be like the following sample:
1735
1736     ---yaml
1737     geometry:
1738         uni1: # Name of this geometry
1739         type: Union
1740         geometries: # create geometries recursively below
1741             uni_ell1: # Name of the first ellipse to union
1742                 type: Ellipse # Type of this geometry
1743                 semiaxes: [2.0, 1.0] # Specific parameters
1744                 pos: [1.0, 1.0]
1745             uni_poly1: # Name of the second polygon
1746                 type: Polygon
1747                 table:
1748                     - [-1.0, -0.3]
1749                     - [2.0, -1.0]
1750                     - [1.0, 1.0]
1751         line1: # This line splits the ellipse into 2 regions.
1752             type: LineSegment
1753             coord1: [1.0, 2.0]
1754             coord2: [3.0, 1.0]
1755
1756         selection:
1757             sel1: # Name of this selection
1758             type: DifferenceSelection
1759             add:
1760                 geometries:
1761                     - uni1 # Select all the regions in uni1
1762             subtract:
1763                 points:
1764                     - [2.5, 1.5] # Remove the region where (2.5, 1.5) in. This
1765                     ↳ region is part of ellipse but splitted by the line
1766                     ↳ segment.
1767
1768     ---
1769
1770     ## Geometric Design of Inductor2d
1771
1772     You are asked to design an inductor. The objective is to maximize the
1773     ↳ inductance value of the inductor which is calculated as
1774     ↳  $\$0.5 * \text{real}(\text{Br} * \text{conj}(\text{Hr}) + \text{conj}(\text{Hphi}) * \text{Bphi} + \text{conj}(\text{Hz}) * \text{Bz}) / V\$$  with B, H to
1775     ↳ be the magnetic induction intensity and magnetic field intensity
1776     ↳ and V to be the volume of the core.
1777
1778     The geometry should be designed inside a semicircle of radius 0.35m
1779     ↳ centered at (0,0), opening in the positive x-direction. Then we
1780     ↳ will generate a 3D geometry by rotating the semicircle around the
1781     ↳ axis x=0.
1782
1783     You are required to give the geometry of the core, and the location of
1784     ↳ the coils. After you create the geometry, you should select the
1785     ↳ regions of the core. **The Name of the selection must be `core`**.
1786     ↳ Finally, you need to give the center of the coils, which are
1787     ↳ circles with radius 0.01m. You don't need to give the geometry of
1788     ↳ the coils.
1789
1790     The constraints are as follow:
1791     1. There's no overlapping of different coils. There must be 12 coils
1792     ↳ in total.

```

```

1782
1783 2. The core and the coils should not overlap or adjacent.
1784 3. The geometry and coils should be placed inside the semicircle of
1785   → radius 0.35m centered at (0,0), opening in the positive
1786   → x-direction.
1787 4. The volume of the core should be more than 0.001 m^3. This means
1788   → cores that are **extremely thin or extremely fine** are not
1789   → allowed.

1790 The reward is calculated as follow:
1791 1. 0 if constraints are violated.
1792 2.  $0.5 * \text{real}(\text{Br} * \text{conj}(\text{Hr}) + \text{conj}(\text{Hphi}) * \text{Bphi} + \text{conj}(\text{Hz}) * \text{Bz}) / V$ , the
1793   → inductance value of the inductor, if constraints are satisfied.

1794 ## Example
1795 An example solution is shown below. You should not copy the example
1796   → solution, but you can refer to it to understand the task and
1797   → create better ones.

1798 ```yaml
1799 geometry:
1800   main:
1801     type: Difference
1802     geometries_add:
1803       outer:
1804         type: Polygon
1805         table:
1806           - [0, 0.2]
1807           - [0.18, 0.2]
1808           - [0.18, -0.2]
1809           - [0, -0.2]
1810     geometries_subtract:
1811       inner:
1812         type: Polygon
1813         table:
1814           - [0.03, -0.155]
1815           - [0.03, 0.155]
1816           - [0.12, 0.155]
1817           - [0.12, -0.155]
1818   selection:
1819     core:
1820       type: UnionSelection
1821       geometries:
1822         - main
1823     coils:
1824       - [0.08, 0.11]
1825       - [0.08, 0.09]
1826       - [0.08, 0.07]
1827       - [0.08, 0.05]
1828       - [0.08, 0.03]
1829       - [0.08, 0.01]
1830       - [0.08, -0.01]
1831       - [0.08, -0.03]
1832       - [0.08, -0.05]
1833       - [0.08, -0.07]
1834       - [0.08, -0.09]
1835       - [0.08, -0.11]
1836 ```

```

## Prompt for task Beam Bending

You are a helpful AI Assistant that provides well-reasoned and detailed responses.  
You first think about the reasoning process as an internal monologue and then provide the user with the answer.  
Respond in the following format: <think>  
...  
</think>  
<answer>  
...  
</answer>.

**## Task Description**

You are a helpful AI Assistant and scientist with strong physical background and wonderful geometric designing ideas.  
You are asked to generate the geometry design of a component using yaml files under certain constraints. You will first create geometries of your design, and then assign functions to the geometries according to the specific requirements.

Your final answer should contain a yaml file enclosed in `'''yaml\n(your code)'''`. The yaml file should have at least two parts: geometry and selection. The specific requirements are as follow:

1. **geometry:** A list of objects with type and type-specific parameters.  
The types and parameters are as follows:  
Polygon: (2D) You can use it to create rectangles, triangles, etc.  
table: Ordered list of n vertices as [x, y] points. The polygon is formed by **connecting consecutive points** (p<sub>i</sub>->p<sub>{i+1}</sub>) and **automatically closing** the shape (p<sub>n</sub>->p<sub>1</sub>). **NO Intersections between edges/nodes are allowed**.  
fillet: (Optional) A list of [i, r] tuples, where i is the index (starting from 1) of a polygon vertex defined in the above table, and r is the fillet radius for that corresponding vertex.  
Ellipse: (2D) You can use it to create circles.  
semiaxes: [horizontal, vertical] axis lengths  
pos: [center\_x, center\_y] center position  
rot: (Optional) Rotation angle (degree) counterclockwise  
angle: (Optional) Angular span (degree) counterclockwise. e.g. by setting angle=180 you can draw a upward semicircle.  
LineSegment: (1D)  
coord1: [start\_x, start\_y]  
coord2: [end\_x, end\_y]  
CircularArc: (1D)  
r: Radius  
angle1: Start angle (degree) counterclockwise, 0 degree represent positive direction of X-axis.  
angle2: End angle (degree) counterclockwise  
CubicBezier: (1D)  
p: Control points as [[x<sub>0</sub>, x<sub>1</sub>, x<sub>2</sub>, x<sub>3</sub>], [y<sub>0</sub>, y<sub>1</sub>, y<sub>2</sub>, y<sub>3</sub>]]  
w: Weight values as [w<sub>0</sub>, w<sub>1</sub>, w<sub>2</sub>, w<sub>3</sub>]  
InterpolationCurve: (1D)  
table: Ordered list of [x, y] points to interpolate through.  
The curve will pass every points smoothly (polynomial interpolation for x and y).  
ParametricCurve: (1D)  
parname: Name of parameter  
parmin: Minimum value of parameter  
parmax: Maximum value of parameter

```

1890
1891     coord: Expressions about the parameter like ["expression_x",
1892         ↵ "expression_y"]. Trigonometric functions here use radians
1893     ConvertToSolid: (2D) Geometry formed by end-to-end connected 1D
1894         ↵ curves.
1895     geometries: A dictionary of 1D geometries (using the same
1896         ↵ structure as the top-level geometry section, recursive).
1897         ↵ **They Must connect end-to-end and form a simply connected
1898         ↵ space**.
1899     Union: (2D) Union of 2D geometries.
1900     geometries: A dictionary of geometries (recursive).
1901     Intersection: (2D) Intersection of 2D geometries.
1902     geometries: A dictionary of geometries (recursive).
1903     Difference: (2D) Difference of the 2D geometries.
1904         geometries_add: A dictionary of geometries to keep
1905             ↵ (recursive).
1906         geometries_subtract: A dictionary of geometries to subtract
1907             ↵ (recursive).
1908
1909 After **geometry** was created, the shapes will be splitted into
1910     ↵ **non-overlapping connected regions**.
1911     - Overlapping 2D shapes create new regions (e.g., two intersecting
1912         ↵ circles → 3 regions)
1913     - Enclosed 2D shapes split regions (e.g., circle inside polygon →
1914         ↵ 2 regions: circle interior + polygon-ring)
1915     - 1D curves through 2D shapes create sub-regions (e.g., line
1916         ↵ segment through rectangle → alternating regions)
1917 The **regions** can be represented by the following ways:
1918     - point: You can select an interior point of the region to
1919         ↵ represent it. The point should never on boundaries/corners.
1920         ↵ One point per region suffices.
1921     - geometry: The 2d shapes you created might be splitted into
1922         ↵ several regions. You can select the geometry to represent all
1923         ↵ the regions in it.
1924
1925 2. selection: After regions are created, you will assign different
1926     ↵ functions to regions using selections.
1927     UnionSelection: Union of all the regions selected below.
1928         points: (Optional) List of [x,y] points representing distinct
1929             ↵ regions.
1930         geometries: (Optional) List of 2d geometry names you created
1931             ↵ above. By listing geometries here, you can select all the
1932             ↵ region this geometry contains.
1933         selections: (Optional) List of other selection names you
1934             ↵ created.
1935     IntersectionSelection: Intersection of all the regions selected
1936         ↵ below.
1937             same parameters as UnionSelection
1938     DifferenceSelection: Select the regions in Add but not in
1939         ↵ Subtract.
1940             add: same parameters as UnionSelection.
1941             subtract: same parameters as UnionSelection.
1942
1943 Finally a yaml file will be like the following sample:
1944
1945 ````yaml
1946 geometry:
1947     uni1: # Name of this geometry
1948         type: Union
1949         geometries: # create geometries recursively below
1950             uni_ell1: # Name of the first ellipse to union
1951                 type: Ellipse # Type of this geometry
1952                 semiaxes: [2.0, 1.0] # Specific parameters
1953                 pos: [1.0, 1.0]

```

```

1944
1945     uni_poly1: # Name of the second polygon
1946         type: Polygon
1947         table:
1948             - [-1.0, -0.3]
1949             - [2.0, -1.0]
1950             - [1.0, 1.0]
1951     line1: # This line splits the ellipse into 2 regions.
1952         type: LineSegment
1953         coord1: [1.0, 2.0]
1954         coord2: [3.0, 1.0]
1955
1956     selection:
1957         sel1: # Name of this selection
1958             type: DifferenceSelection
1959             add:
1960                 geometries:
1961                     - unil # Select all the regions in unil
1962             subtract:
1963                 points:
1964                     - [2.5, 1.5] # Remove the region where (2.5, 1.5) in. This
1965                     → region is part of ellipse but splitted by the line
1966                     → segment.
1967
1968     ...
1969
1970     ## Beam Cross Section Geometry Design
1971
1972 You are asked to design the cross section of a beam. The objective is
1973 → to maximize both the largest principal moment of inertia and the
1974 → torsional constant, while keeping the cross section area small.
1975 → The goal can be quantified as  $(I_1^{**0.8} * I_2^{**0.2})/A$  with  $I_1$ 
1976 → being the largest principal moment of inertia,  $I_2$  being the
1977 → smallest principal moment of inertia and  $A$  being the beam cross
1978 → section area.
1979
1980 The beam cross-sectional dimension should not go beyond 0.15m, with
1981 → the center staying close to the origin.
1982
1983 You are required to design the beam cross section. The shape doesn't
1984 → have to be symmetric. After you create the geometry, you should
1985 → select the regions of the beam. **The Name of the selection must
1986 → be `beam`**.
1987
1988 The constraints are as follow:
1989 1. The shape center should be close to the origin.
1990 2. The shape should stay inside the circle boundary with radius 0.2m.
1991 3. The area should not be smaller than  $2e-3 \text{ m}^2$ .
1992
1993 The reward is calculated as follow:
1994 1. 0 if constraints are violated.
1995 2.  $(I_1^{**0.8} * I_2^{**0.2})/A$ , weighted geometry average of the largest
1996 → principal moment of inertia and the smallest principal moment of
1997 → inertia, normalized by the cross section area, if constraints are
1998 → satisfied.
1999
2000     ## Example
2001 An example solution is shown below. You should not copy the example
2002 → solution, but you can refer to it to understand the task and
2003 → create better ones.
2004
2005     ````yaml
2006     geometry:
2007         pol1:
2008             type: Polygon
2009

```

```

1998
1999     table:
2000     - [0.05, 0.05]
2001     - [-0.05, 0.05]
2002     - [-0.05, 0.04]
2003     - [-0.005, 0.04]
2004     - [-0.005, -0.04]
2005     - [-0.05, -0.04]
2006     - [-0.05, -0.05]
2007     - [0.05, -0.05]
2008     - [0.05, -0.04]
2009     - [0.005, 0.04]
2010     - [0.005, 0.04]
2011     - [0.05, 0.04]
2012     - [4, 0.003]
2013     - [5, 0.003]
2014     - [10, 0.003]
2015     - [11, 0.003]
2016     selection:
2017     beam:
2018     - type: UnionSelection
2019     - geometries:
2020     - poll1
2021     ...

```

### Prompt for task Magnetic Torque

2022 You are a helpful AI Assistant that provides well-reasoned and  
 2023 → detailed responses.  
 2024 You first think about the reasoning process as an internal monologue  
 2025 → and then provide the user with the answer.  
 2026 Respond in the following format: <think>  
 2027 ...  
 2028 </think>  
 2029 <answer>  
 2030 ...  
 2031 </answer>.  
**## Task Description**  
 2032 You are a helpful AI Assistant and scientist with strong physical  
 2033 → background and wonderful geometric designing ideas.  
 2034 You are asked to generate the geometry design of a component using  
 2035 → yaml files under certain constraints. You will first create  
 2036 → geometries of your design, and then assign functions to the  
 2037 → geometries according to the specific requirements.  
 2038 Your final answer should contain a yaml file enclosed in  
 2039 → ```yaml\n(your code)```. The yaml file should have at least two  
 2040 → parts: geometry and selection. The specific requirements are as  
 2041 → follow:  
 2042 1. **geometry**: A list of objects with type and type-specific parameters.  
 2043 → The types and parameters are as follows:  
 2044 **Polygon**: (2D) You can use it to create rectangles, triangles, etc.  
 2045 table: Ordered list of n vertices as [x, y] points. The  
 2046 → polygon is formed by **\*\*connecting consecutive points\*\***  
 2047 → (p<sub>i</sub>->p<sub>(i+1)</sub>) and **\*\*automatically closing\*\*** the shape  
 2048 → (p<sub>n</sub>->p<sub>1</sub>).  
 2049 fillet: (Optional) A list of [i, r] tuples, where i is the  
 2050 → index (starting from 1) of a polygon vertex defined in the  
 2051 → above table, and r is the fillet radius for that  
 → corresponding vertex.

```

2052
2053     Ellipse: (2D) You can use it to create circles.
2054         semiaxes: [horizontal, vertical] axis lengths
2055         pos: [center_x, center_y] center position
2056         rot: (Optional) Rotation angle (degree) counterclockwise
2057         angle: (Optional) Angular span (degree) counterclockwise. e.g.
2058             → by setting angle=180 you can draw a upward semicircle.
2059
2060     LineSegment: (1D)
2061         coord1: [start_x, start_y]
2062         coord2: [end_x, end_y]
2063
2064     CircularArc: (1D)
2065         r: Radius
2066         angle1: Start angle (degree) counterclockwise, 0 degree
2067             → represent positive direction of X-axis.
2068         angle2: End angle (degree) counterclockwise
2069
2070     CubicBezier: (1D)
2071         p: Control points as [[x0,x1,x2,x3], [y0,y1,y2,y3]]
2072         w: Weight values as [w0,w1,w2,w3]
2073
2074     InterpolationCurve: (1D)
2075         table: Ordered list of [x,y] points to interpolate through.
2076             → The curve will pass every points smoothly (polynomial
2077                 → interpolation for x and y).
2078
2079     ParametricCurve: (1D)
2080         parname: Name of parameter
2081         parmin: Minimum value of parameter
2082         parmax: Maximum value of parameter
2083         coord: Expressions about the parameter like ["expression_x",
2084             → "expression_y"]. Trigonometric functions here use radians
2085
2086     ConvertToSolid: (2D) Geometry formed by end-to-end connected 1D
2087         → curves.
2088         geometries: A dictionary of 1D geometries (using the same
2089             → structure as the top-level geometry section, recursive).
2090             → They Must connect end-to-end and form a simply connected
2091                 → space**.
2092
2093     Union: (2D) Union of 2D geometries.
2094         geometries: A dictionary of geometries (recursive).
2095
2096     Intersection: (2D) Intersection of 2D geometries.
2097         geometries: A dictionary of geometries (recursive).
2098
2099     Difference: (2D) Difference of the 2D geometries.
2100         geometries_add: A dictionary of geometries to keep
2101             → (recursive).
2102         geometries_subtract: A dictionary of geometries to subtract
2103             → (recursive).

2104
2105
2106     After geometry** was created, the shapes will be splitted into
2107         → non-overlapping connected regions**.
2108         - Overlapping 2D shapes create new regions (e.g., two intersecting
2109             → circles → 3 regions)
2110         - Enclosed 2D shapes split regions (e.g., circle inside polygon →
2111             → 2 regions: circle interior + polygon-ring)
2112         - 1D curves through 2D shapes create sub-regions (e.g., line
2113             → segment through rectangle → alternating regions)
2114
2115     The regions** can be represented by the following ways:
2116         - point: You can select an interior point of the region to
2117             → represent it. The point should never be on boundaries/corners.
2118             → One point per region suffices.
2119         - geometry: The 2d shapes you created might be splitted into
2120             → several regions. You can select the geometry to represent all
2121                 → the regions in it.

2122     2. selection: After regions are created, you will assign different
2123         → functions to regions using selections.
2124         UnionSelection: Union of all the regions selected below.

```

```

2106
2107     points: (Optional) List of [x,y] points representing distinct
2108     ↳ regions.
2109     geometries: (Optional) List of 2d geometry names you created
2110     ↳ above. By listing geometries here, you can select all the
2111     ↳ region this geometry contains.
2112     selections: (Optional) List of other selection names you
2113     ↳ created.
2114     IntersectionSelection: Intersection of all the regions selected
2115     ↳ below.
2116     ↳ same parameters as UnionSelection
2117     DifferenceSelection: Select the regions in Add but not in
2118     ↳ Subtract.
2119     ↳ add: same parameters as UnionSelection.
2120     ↳ subtract: same parameters as UnionSelection.

2121 Finally a yaml file will be like the following sample:
2122
2123 ```yaml
2124 geometry:
2125     uni1: # Name of this geometry
2126     type: Union
2127     geometries: # create geometries recursively below
2128     uni_ell1: # Name of the first ellipse to union
2129     type: Ellipse # Type of this geometry
2130     semiaxes: [2.0, 1.0] # Specific parameters
2131     pos: [1.0, 1.0]
2132     uni_poly1: # Name of the second polygon
2133     type: Polygon
2134     table:
2135     - [-1.0, -0.3]
2136     - [2.0, -1.0]
2137     - [1.0, 1.0]
2138     line1: # This line splits the ellipse into 2 regions.
2139     type: LineSegment
2140     coord1: [1.0, 2.0]
2141     coord2: [3.0, 1.0]

2142 selection:
2143     sel1: # Name of this selection
2144     type: DifferenceSelection
2145     add:
2146     geometries:
2147     - uni1 # Select all the regions in uni1
2148     subtract:
2149     points:
2150     - [2.5, 1.5] # Remove the region where (2.5, 1.5) in. This
2151     ↳ region is part of ellipse but splitted by the line
2152     ↳ segment.
2153
2154 ## Geometric Design of 2D Iron Core
2155
2156 You are asked to design a 2D iron core. The iron core has large
2157     ↳ permeability and is subject to a constant far field magnetic field
2158     ↳ intensity which applies a magnetic torque (pointing out of the 2D
2159     ↳ plane) on the iron core. The objective is to maximize the magnetic
     ↳ torque while keeping the iron core small. The goal can be
     ↳ quantified as  $|T_z|/A$  where  $T_z$  is the torque in the out of plane
     ↳ direction and  $A$  is the area of the core.

```

```

2160
2161 The iron core should be designed inside a circle air domain of radius
2162 → 0.05m centered at the origin (0,0). The boundary of the circle
2163 → domain is subject to a constant magnetic field intensity of
2164 → [0,1e5] A/m.
2165
2166 You are required to give the geometry of the iron core. The shape
2167 → doesn't have to be symmetric. After you create the geometry, you
2168 → should select the regions of the core. **The Name of the selection
2169 → must be `core`**.
2170
2171 The constraints are as follow:
2172 1. The core center should be close to the origin and inside the circle
2173 → air domain of radius 0.05m centered at (0,0).
2174 2. The boundary of the core should stay at least 0.02m away from the
2175 → circle air domain.
2176 3. The Area of the core should between 2e-4 and 2e-3 m^2.
2177
2178 The reward is calculated as follow:
2179 1. 0 if constraints are violated.
2180 2.  $|\mathbf{T}_z|/A$ , the absolute value of magnetic torque generated by the
2181 → constant far field magnetic field intensity on the iron core
2182 → normalized by the iron core area, if constraints are satisfied.
2183
2184 ## Example
2185 An example solution is shown below. You should not copy the example
2186 → solution, but you can refer to it to understand the task and
2187 → create better ones.
2188
2189 ---yaml
2190 geometry:
2191   pol1:
2192     type: Polygon
2193     table:
2194       - [0.01, -0.003]
2195       - [0.02, 0.001]
2196       - [0.01, 0.01]
2197       - [-0.01, 0.005]
2198       - [-0.02, -0.008]
2199       - [-0.02, -0.008]
2200       - [-0.003, -0.01]
2201
2202   selection:
2203     core:
2204       type: UnionSelection
2205       geometries:
2206         - pol1
2207
2208 ---
2209
2210
2211
2212
2213

```

#### Prompt for task Periodic Heat

```

2203 You are a helpful AI Assistant that provides well-reasoned and
2204 → detailed responses.
2205 You first think about the reasoning process as an internal monologue
2206 → and then provide the user with the answer.
2207 Respond in the following format: <think>
2208 ...
2209 </think>
2210 <answer>
2211 ...
2212 </answer>.
2213 ## Task Description
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2214
2215 You are a helpful AI Assistant and scientist with strong physical
2216 → background and wonderful geometric designing ideas.
2217 You are asked to generate the geometry design of a component using
2218 → yaml files under certain constraints. You will first create
2219 → geometries of your design, and then assign functions to the
2220 → geometries according to the specific requirements.
2221
2222 Your final answer should contain a yaml file enclosed in
2223 → ```yaml\n(your code)```. The yaml file should have a part named
2224 → geometry. The specific requirements are as follow:
2225
2226 1. geometry: A list of objects with type and type-specific parameters.
2227 → The types and parameters are as follows:
2228   Polygon: (2D) You can use it to create rectangles, triangles, etc.
2229     table: Ordered list of n vertices as [x, y] points. The
2230       → polygon is formed by **connecting consecutive points**
2231       → (p_i->p_{i+1}) and **automatically closing** the shape
2232       → (p_n->p_1).
2233     fillet: (Optional) A list of [i, r] tuples, where i is the
2234       → index (starting from 1) of a polygon vertex defined in the
2235       → above table, and r is the fillet radius for that
2236       → corresponding vertex.
2237   Ellipse: (2D) You can use it to create circles.
2238     semiaxes: [horizontal, vertical] axis lengths
2239     pos: [center_x, center_y] center position
2240     rot: (Optional) Rotation angle (degree) counterclockwise
2241     angle: (Optional) Angular span (degree) counterclockwise. e.g.
2242       → by setting angle=180 you can draw a upward semicircle.
2243   LineSegment: (1D)
2244     coord1: [start_x, start_y]
2245     coord2: [end_x, end_y]
2246   CircularArc: (1D)
2247     r: Radius
2248     angle1: Start angle (degree) counterclockwise, 0 degree
2249       → represent positive direction of X-axis.
2250     angle2: End angle (degree) counterclockwise
2251   CubicBezier: (1D)
2252     p: Control points as [[x0,x1,x2,x3], [y0,y1,y2,y3]]
2253     w: Weight values as [w0,w1,w2,w3]
2254   InterpolationCurve: (1D)
2255     table: Ordered list of [x,y] points to interpolate through.
2256       → The curve will pass every points smoothly (polynomial
2257       → interpolation for x and y).
2258   ParametricCurve: (1D)
2259     parname: Name of parameter
2260     parmin: Minimum value of parameter
2261     parmax: Maximum value of parameter
2262     coord: Expressions about the parameter like ["expression_x",
2263       → "expression_y"]. Trigonometric functions here use radians
2264   ConvertToSolid: (2D) Geometry formed by end-to-end connected 1D
2265   → curves.
2266   geometries: A dictionary of 1D geometries (using the same
2267     → structure as the top-level geometry section, recursive).
2268     → **They Must connect end-to-end and form a simply connected
2269     → space**.
2270   Union: (2D) Union of 2D geometries.
2271   geometries: A dictionary of geometries (recursive).
2272   Intersection: (2D) Intersection of 2D geometries.
2273     geometries: A dictionary of geometries (recursive).
2274   Difference: (2D) Difference of the 2D geometries.
2275     geometries_add: A dictionary of geometries to keep
2276       → (recursive).

```

```

2268
2269     geometries_subtract: A dictionary of geometries to subtract
2270     ↳ (recursive).
2271
2272 ## Structure Design of the 3D heat transfer unit cell
2273
2274 You are asked to design a unit cell structure in 3D. The objective is
2275 ↳ to maximize the effective thermal conductivity with limited
2276 ↳ material usage, which can be quantified as
2277 ↳  $\$trace(k_{\text{eff}})/\rho_{\text{eff}}$  where  $k_{\text{eff}}$  is the effective
2278 ↳ thermal conductivity matrix of shape  $3 \times 3$ , and  $\rho_{\text{eff}}$  is the
2279 ↳ effective density.
2280
2281 You should first define a 2D rectangular unit cell domain by giving
2282 ↳ the width and height of the domain.
2283
2284 You then design the hollow part of the 2D unit cell. You must create a
2285 ↳ geometry named 'hollow', represents the hollow part of the unit
2286 ↳ cell. Four copies of this geometry object will be created by
2287 ↳ translating the original object with the following vectors
2288 ↳  $[-\text{cell\_width}/2, -\text{cell\_height}/2], [-\text{cell\_width}/2, \text{cell\_height}/2],$ 
2289 ↳  $[\text{cell\_width}/2, -\text{cell\_height}/2], [\text{cell\_width}/2, \text{cell\_height}/2]$ .
2290
2291 The unit cell domain will be subtracted by the geometry object and its
2292 ↳ copies. The subtracted areas are filled with air (thermal
2293 ↳ conductivity  $\sim 0.026 \text{W}/(\text{m} \cdot \text{K})$ , density  $\sim 1.174 \text{kg}/\text{m}^3$ ) and the
2294 ↳ remaining areas are filled with aluminum (thermal conductivity
2295 ↳  $\sim 238 \text{W}/(\text{m} \cdot \text{K})$ , density  $2700 \text{ kg}/\text{m}^3$ ).
2296 The final 3D unit cell structure is generated by extruding the 2D unit
2297 ↳ cell.
2298
2299 The unit cell will subject to periodic boundary condition in  $x$ ,  $y$ , and
2300 ↳  $z$  directions. Your design should provide higher effective thermal
2301 ↳ conductivity using a reasonable amount of aluminum and carefully
2302 ↳ designed hollow shape.
2303
2304 The constraints are as follow:
2305 1. The original geometry object ('hollow') should not overlap with the
2306 ↳ boundary of the domain. But its copies may overlap with the
2307 ↳ boundary.
2308 2. The original and copied geometry objects should not overlap or
2309 ↳ adjacent with each other.
2310 3. The effective density should not exceed  $2000 \text{ kg}/\text{m}^3$ . You should
2311 ↳ control the usage of aluminum (by create larger hollow parts).
2312 4. The unit cell domain is centered strictly at the origin. The
2313 ↳ geometry you designed should be centered approximately at the
2314 ↳ origin, as it may not be symmetric.
2315
2316 The reward is calculated as follow:
2317 1. 0 if constraints are violated.
2318 2.  $\$trace(k_{\text{eff}})/\rho_{\text{eff}}$ , the effective thermal conductivity of the
2319 ↳ unit cell structure normalized by the effective density, if
2320 ↳ constraints are satisfied.
2321
2322 ## Example
2323 An example solution is shown below. You should not copy the example
2324 ↳ solution, but you can refer to it to understand the task and
2325 ↳ create better ones.
2326
2327 '''yaml
2328 cell:
2329     sizes: [5e-3, 3e-3]
2330 geometry:
2331     hollow:

```

```

2322
2323     type: Polygon
2324     table:
2325     - [1.61e-3, 0]
2326     - [8.04e-4, 1.45e-3]
2327     - [-8.04e-4, 1.45e-3]
2328     - [-1.61e-3, 0]
2329     - [-8.04e-4, -1.45e-3]
2330     - [8.04e-4, -1.45e-3]
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375

```

### Prompt for task Demultiplexer

You are a helpful AI Assistant that provides well-reasoned and  
 ↳ detailed responses.  
 You first think about the reasoning process as an internal monologue  
 ↳ and then provide the user with the answer.

Respond in the following format: <think>

...

</think>

<answer>

...

</answer>.

### ## Task Description

You are a helpful AI Assistant and scientist with strong physical  
 ↳ background and wonderful geometric designing ideas.

You are asked to generate the geometry design of a component using  
 ↳ yaml files under certain constraints. You will first create  
 ↳ geometries of your design, and then assign functions to the  
 ↳ geometries according to the specific requirements.

Your final answer should contain a yaml file enclosed in

↳ `` `yaml\n(your code)` `` . The yaml file should have at least two  
 ↳ parts: geometry and selection. The specific requirements are as  
 ↳ follow:

1. geometry: A list of objects with type and type-specific parameters.

↳ The types and parameters are as follows:

Polygon: (2D) You can use it to create rectangles, triangles, etc.

table: Ordered list of n vertices as [x, y] points. The  
 ↳ polygon is formed by **\*\*connecting consecutive points\*\***  
 ↳ (p\_i->p\_{i+1}) and **\*\*automatically closing\*\*** the shape  
 ↳ (p\_n->p\_1).

fillet: (Optional) A list of [i, r] tuples, where i is the  
 ↳ index (starting from 1) of a polygon vertex defined in the  
 ↳ above table, and r is the fillet radius for that  
 ↳ corresponding vertex.

Ellipse: (2D) You can use it to create circles.

semiaxes: [horizontal, vertical] axis lengths

pos: [center\_x, center\_y] center position

rot: (Optional) Rotation angle (degree) counterclockwise

angle: (Optional) Angular span (degree) counterclockwise. e.g.  
 ↳ by setting angle=180 you can draw a upward semicircle.

LineSegment: (1D)

coord1: [start\_x, start\_y]

coord2: [end\_x, end\_y]

CircularArc: (1D)

r: Radius

angle1: Start angle (degree) counterclockwise, 0 degree

↳ represent positive direction of X-axis.

angle2: End angle (degree) counterclockwise

CubicBezier: (1D)

```

2376
2377     p: Control points as [[x0,x1,x2,x3], [y0,y1,y2,y3]]
2378     w: Weight values as [w0,w1,w2,w3]
2379     InterpolationCurve: (1D)
2380         table: Ordered list of [x,y] points to interpolate through.
2381             ↳ The curve will pass every points smoothly (polynomial
2382                 ↳ interpolation for x and y).
2383     ParametricCurve: (1D)
2384         parname: Name of parameter
2385         parmin: Minimum value of parameter
2386         parmax: Maximum value of parameter
2387         coord: Expressions about the parameter like ["expression_x",
2388                 ↳ "expression_y"]. Trigonometric functions here use radians
2389     ConvertToSolid: (2D) Geometry formed by end-to-end connected 1D
2390         ↳ curves.
2391         geometries: A dictionary of 1D geometries (using the same
2392             ↳ structure as the top-level geometry section, recursive).
2393             ↳ **They Must connect end-to-end and form a simply connected
2394             ↳ space**.
2395     Union: (2D) Union of 2D geometries.
2396         geometries: A dictionary of geometries (recursive).
2397     Intersection: (2D) Intersection of 2D geometries.
2398         geometries: A dictionary of geometries (recursive).
2399     Difference: (2D) Difference of the 2D geometries.
2400         geometries_add: A dictionary of geometries to keep
2401             ↳ (recursive).
2402         geometries_subtract: A dictionary of geometries to subtract
2403             ↳ (recursive).

2404 After **geometry** was created, the shapes will be splitted into
2405     ↳ **non-overlapping connected regions**.
2406     - Overlapping 2D shapes create new regions (e.g., two intersecting
2407         ↳ circles → 3 regions)
2408     - Enclosed 2D shapes split regions (e.g., circle inside polygon →
2409         ↳ 2 regions: circle interior + polygon-ring)
2410     - 1D curves through 2D shapes create sub-regions (e.g., line
2411         ↳ segment through rectangle → alternating regions)
2412 The **regions** can be represented by the following ways:
2413     - point: You can select an interior point of the region to
2414         ↳ represent it. The point should never on boundaries/corners.
2415         ↳ One point per region suffices.
2416     - geometry: The 2d shapes you created might be splitted into
2417         ↳ several regions. You can select the geometry to represent all
2418         ↳ the regions in it.

2419 2. selection: After regions are created, you will assign different
2420     ↳ functions to regions using selections.
2421     UnionSelection: Union of all the regions selected below.
2422         points: (Optional) List of [x,y] points representing distinct
2423             ↳ regions.
2424         geometries: (Optional) List of 2d geometry names you created
2425             ↳ above. By listing geometries here, you can select all the
2426             ↳ region this geometry contains.
2427         selections: (Optional) List of other selection names you
2428             ↳ created.
2429     IntersectionSelection: Intersection of all the regions selected
2430         ↳ below.
2431         same parameters as UnionSelection
2432     DifferenceSelection: Select the regions in Add but not in
2433         ↳ Subtract.
2434         add: same parameters as UnionSelection.
2435         subtract: same parameters as UnionSelection.

2436 Finally a yaml file will be like the following sample:
2437
2438
2439

```

```

2430
2431
2432 ```yaml
2433 geometry:
2434   uni1: # Name of this geometry
2435     type: Union
2436     geometries: # create geometries recursively below
2437       uni_ell1: # Name of the first ellipse to union
2438         type: Ellipse # Type of this geometry
2439         semiaxes: [2.0, 1.0] # Specific parameters
2440         pos: [1.0, 1.0]
2441       uni_poly1: # Name of the second polygon
2442         type: Polygon
2443         table:
2444           - [-1.0, -0.3]
2445           - [2.0, -1.0]
2446           - [1.0, 1.0]
2447     line1: # This line splits the ellipse into 2 regions.
2448       type: LineSegment
2449       coord1: [1.0, 2.0]
2450       coord2: [3.0, 1.0]
2451
2452
2453
2454
2455
2456
2457
2458
2459 selection:
2460   sel1: # Name of this selection
2461     type: DifferenceSelection
2462     add:
2463       geometries:
2464         - uni1 # Select all the regions in uni1
2465     subtract:
2466       points:
2467         - [2.5, 1.5] # Remove the region where (2.5, 1.5) in. This
2468           ↳ region is part of ellipse but splitted by the line
2469           ↳ segment.
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483 ## Geometric Design of a 2D sound wave demultiplexer
2484
2485 You are asked to design a 2D sound wave demultiplexer. The
2486   ↳ demultiplexer takes incident sound wave from port 1, and omits
2487   ↳ sound wave at port 2 and 3. The objective is to maximize the
2488   ↳ difference of sound pressure (on log scale) at two outlet ports,
2489   ↳ which is calculated as  $\log_{10}(\text{port2.P}_{\text{out}}) - \log_{10}(\text{port3.P}_{\text{out}})$ 
2490   ↳ with  $\text{P}_{\text{out}}$  being the sound pressure at the outlet ports.
2491
2492 The entire pressure acoustic region will be a circle of radius 0.1m
2493   ↳ centered at (0,0). The incident wave comes from the negative
2494   ↳ x-direction (9 o'clock). The sound waves are then omitted at 1
2495   ↳ o'clock (port 2) and 5 o'clock (port 3) of the acoustic region.
2496
2497 You should design void geometry (material to be removed) so that the
2498   ↳ sound wave will propagate through the remaining geometry and
2499   ↳ maximize the objective function. You should create a list of basic
2500   ↳ geometries and then select from them to form the void regions.
2501   ↳ **The Name of the selection must be `void`**. Keep in mind that
2502   ↳ the void geometry should stay inside the acoustic region and at
2503   ↳ least 0.15m away from the boundary of the acoustic region.
2504
2505 The constraints are as follow:
2506 1. After removing the void geometry, the remaining part should still
2507    ↳ be connected.
2508 2. The void geometry should stay inside the acoustic region, and at
2509    ↳ least 0.02m away from the boundary of the acoustic region.
2510
2511 The reward is calculated as follow:

```

```

2484
2485 1. 0 if constraints are violated.
2486 2.  $\log_{10}(\text{port2.P}_{\{\text{out}\}}) - \log_{10}(\text{port3.P}_{\{\text{out}\}})$ , the log scale pressure
2487 → difference between port 2 and port 3, if constraints are
2488 → satisfied.
2489
## Example
2490 An example solution is shown below. You should not copy the example
2491 → solution, but you can refer to it to understand the task and
2492 → create better ones. Feel free to add more basic geometries.
2493
yaml
2494 geometry:
2495   barrier:
2496     type: ConvertToSolid
2497     geometries:
2498       cir_inner:
2499         type: CircularArc
2500         r: 0.08
2501         angle1: -120
2502         angle2: 0
2503       cir_outer:
2504         type: CircularArc
2505         r: 0.06
2506         angle1: -120
2507         angle2: 0
2508       line1:
2509         type: LineSegment
2510         coord1: [0.06, 0.0]
2511         coord2: [0.08, 0.0]
2512       line2:
2513         type: LineSegment
2514         coord1: [0.06*cos(-120*pi/180), 0.06*sin(-120*pi/180)]
2515         coord2: [0.08*cos(-120*pi/180), 0.08*sin(-120*pi/180)]
2516
2517
2518
2519
2520
2521
2522 D.3 CIRCLE PACKING
2523
2524
Prompt for task Circle Packing
2525
2526 You are an expert software developer tasked with iteratively improving
2527 → a codebase.
2528 Your job is to analyze the current program and suggest improvements
2529 → based on feedback from previous attempts.
2530 Focus on making targeted changes that will increase the program's
2531 → performance metrics.
2532 Respond in the following format: <think>
2533 ...
2534 </think>
2535 <answer>
2536 ...
2537 </answer>.
# Problem Description

```

```

2538
2539 You are an expert mathematician specializing in circle packing
2540 → problems and computational geometry. Your task is to improve a
2541 → constructor function that directly produces a specific arrangement
2542 → of 26 circles in a unit square, maximizing the sum of their radii.
2543 → The AlphaEvolve paper achieved a sum of 2.635 for n=26.
2544
2545 Key geometric insights:
2546 - In dense regions, circles often follow hexagonal packing patterns,
2547 → with the theoretical maximum density for infinite packing being
2548 →  $\pi/(2\sqrt{3})=0.9069$ .
2549 - However, when confined to a finite square, edge effects disrupt
2550 → perfect symmetry and make pure hexagonal packing suboptimal.
2551 - Optimal arrangements often require variable-sized circles, as
2552 → this can improve space utilization compared to equal radii. Larger
2553 → circles can be placed toward the center, with smaller circles
2554 → strategically fitted near edges and corners.
2555 - Effective designs may use layered or shell-like patterns rather
2556 → than strict hexagonal grids. Hybrid approaches combining regular
2557 → arrangements in dense regions with adaptive adjustments near
2558 → boundaries are common in the densest known packings.
2559 - The optimization method plays a critical role: physics-inspired
2560 → simulations or algorithms with well-tuned parameters can yield
2561 → better configurations than purely geometric intuition.
2562 - Mathematical research indicates that for certain specific values of
2563 → n, special arrangements can achieve unusually high densities.
2564
2565 You may either design an explicit constructor of the result or
2566 → explore search-based, optimization, or even multi-stage
2567 → optimization methods, as long as they can finish running within 1
2568 → minutes.
2569
2570 ## Current Program
2571 Status: {current_status}
2572 ````python
2573 {current_program}
2574 ````

2575
2576 ## Task
2577 Suggest improvements to the program that will lead to better
2578 → performance on the specified metrics.
2579
2580 You MUST use the exact SEARCH/REPLACE diff format shown below to
2581 → indicate changes:
2582
2583 <<<<< SEARCH
2584 # Original code to find and replace (must match exactly)
2585 =====
2586 # New replacement code
2587 >>>>> REPLACE
2588
2589 You can suggest multiple changes. Each SEARCH section must exactly
2590 → match code in the current program.
2591 Be thoughtful about your changes and explain your reasoning
2592 → thoroughly.
2593
2594 Make sure your rewritten program still contains construct_packing()
2595 → function and maintains the same outputs. You can
2596 → add/delete/modify other functions arbitrarily.
2597
2598 If you want to use new packages, please import them. Usable packages:
2599 → scipy, sympy, shapely, pulp, cvxpy, nlopt, deap
2600
2601

```

```

2592
2593 If your code's execution time exceeds 1 minute, you will receive 0
2594 → reward. Pay attention to the runtime efficiency!
2595
2596 IMPORTANT: Do not rewrite the entire program - focus on targeted
2597 → improvements.
2598
2599
2600
2601 D.4 FUNCTION MINIMIZATION
2602
2603 Prompt for task Minimize Function
2604
2605 # Problem Description
2606
2607 You are an expert in optimization algorithms. Your task is to improve
2608 → a function minimization algorithm that minimizes a complex
2609 → non-convex function with multiple local minima. The function is
2610 → defined in {dimension}-dimensional space with the following
2611 → expression:
2612 ```python
2613 {formula}
2614 ```
2615
2616
2617 ## Current Program
2618 Status: {current_status}
2619 ```python
2620 {current_program}
2621 ```
2622
2623 ## Task
2624
2625 Suggest improvements to the program that will lead to better
2626 → performance on the specified metrics.
2627
2628 Your code's execution time should not exceed 10 seconds. Pay attention
2629 → to the runtime efficiency!
2630
2631 You MUST use the exact SEARCH/REPLACE diff format shown below to
2632 → indicate changes:
2633
2634 <<<<< SEARCH
2635 # Original code to find and replace (must match exactly)
2636 =====
2637 # New replacement code
2638 >>>>> REPLACE
2639
2640 Performance is evaluated using:
2641 1. value_score: Closeness to minimum function value: |global_min| /
2642 → (|global_min| + |found_value - global_min|)
2643 2. distance_score: Proximity to true solution point: 1/(1 +
2644 → distance_to_global_min)
2645 3. standard_deviation_score: Solution stability across runs:
2646 → (1/(1+std_x1) + 1/(1+std_x2) + ...)/dim
2647 4. speed_score: Execution efficiency: min(1/avg_runtime_in_seconds,
2648 → 10)/10
2649 5. reliability_score: successful_runs/total_runs. Successful run has
2650 → no tracebacks and timeouts.
2651 6. combined_score: **This is the final reward you received.** 100%
2652 → value_score.
2653
2654 If you want to use new packages, please import them.
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4098
4099
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4198
4199
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4298
4299
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4398
4399
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4498
4499
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4598
4599
4599
4600
4601
4602
4603
4604
4605
4606
4607
460
```

2646  
 2647 Make sure your rewritten program still contains `'run_search()'`  
 2648 → function and maintains the same outputs. You can add/delete/modify  
 2649 → other functions arbitrarily.

2650 IMPORTANT: Do not rewrite the entire program - focus on targeted  
 2651 → improvements.

2652

2653 **D.5 SYMBOLIC REGRESSION**

2654

**Prompt for Chemistry tasks**

2655 You are an expert software developer. Your job is to write a Python  
 2656 → function based on feedback from previous attempts.

2657 Write your code in exactly the following format:

2658 ````python`  
 2659 `# your code`  
 2660 `````

2661 Your code's execution time is limited, so pay attention to runtime  
 2662 → efficiency!

2663 If you use new packages, please import them.

2664 Ensure the program still contains the `func()` function and produces the  
 2665 → same outputs; other functions can be added, deleted, or modified  
 2666 → freely.

2667 IMPORTANT: The current task is a symbolic regression problem. Write a  
 2668 → Python expression in `func()` where parameter scales are as similar  
 2669 → as possible (use linear scaling or translation if needed). This  
 2670 → helps later optimization when all parameters are initialized  
 2671 → randomly in  $[0,1]$ .

2672 Respond in the following format: <think>

2673 `...`

2674 `</think>`

2675 `<answer>`

2676 `...`

2677 `</answer>.`

2678 Your task is to evolve a Python function `'func(x, params)'` that models  
 2679 → a scientific process, considering the physical meaning and  
 2680 → relationships of inputs, by predicting output variables based on  
 2681 → input variables.

2682 The function signature is:

2683 ````python`

2684 `def func(x: np.ndarray, params: np.ndarray) -> np.ndarray:`

2685 `````

2686 → `'x'` is a 2D NumPy array of shape `'(n_samples, 2)'`

2687 → `'params'` is a 1D NumPy array of up to 10 parameters

2688 → The function should return a 1D NumPy array of predictions with  
 2689 → shape `'(n_samples,)'`

2690 **\*\*Current Problem:\*\***

2691 Model the `dA_dt` (Rate of change of concentration in chemistry reaction  
 2692 → kinetics) using the input features: `t` (Time), `A` (Concentration at  
 2693 → time `t`)

2694 Thus, `'x'` contains 2 columns: `t` (Time), `A` (Concentration at time `t`).

2695

2696 The initial version of `'func'` is a simple linear model. Parameters in  
 2697 → `'params'` will be optimized externally using the BFGS algorithm  
 2698 → based on unseen training data.

2699 Your objective is to evolve `'func'` to improve predictive performance  
 2700 → on unseen data. Aim for a balance between:

```

2700
2701 - **Accuracy**: Lower mean squared error (MSE) on training data
2702 - **Simplicity**: Prefer concise, interpretable expressions
2703
2704 Model performance (score =  $-\log_{10}(\text{mse})$ ) will be evaluated on a
2705  $\hookrightarrow$  held-out dataset. Ensure the model is free of potential numerical
2706  $\hookrightarrow$  errors (e.g., log0, division by 0).
2707 ## Current Program
2708 Status: Initial Program
2709  $\sim\sim\text{python}$ 
2710 def func(x, params):
2711     """
2712         Calculates the model output using a linear combination of input
2713          $\hookrightarrow$  variables
2714         or a constant value if no input variables. Operates on a matrix of
2715          $\hookrightarrow$  samples.
2716
2717     Args:
2718         x (np.ndarray): A 2D numpy array of input variable values,
2719          $\hookrightarrow$  shape (n_samples, n_features).
2720         n_features is 2.
2721         If n_features is 0, x should be shape
2722          $\hookrightarrow$  (n_samples, 0).
2723         The order of columns in x must correspond to:
2724         (t, A).
2725         params (np.ndarray): A 1D numpy array of parameters.
2726          $\hookrightarrow$  Expected length: 10.
2727
2728     Returns:
2729         np.ndarray: A 1D numpy array of predicted output values, shape
2730          $\hookrightarrow$  (n_samples,).
2731         """
2732         result = x[:, 0] * params[0] + x[:, 1] * params[1]
2733         return result
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753

```

### Prompt for Biology tasks

You are an expert software developer. Your job is to write a Python  
 $\hookrightarrow$  function based on feedback from previous attempts.  
 Write your code in exactly the following format:  
 $\sim\sim\text{python}$   
 $\#$  your code  
 $\sim\sim$

Your code's execution time is limited, so pay attention to runtime  
 $\hookrightarrow$  efficiency!  
 If you use new packages, please import them.  
 Ensure the program still contains the `func()` function and produces the  
 $\hookrightarrow$  same outputs; other functions can be added, deleted, or modified  
 $\hookrightarrow$  freely.

**IMPORTANT:** The current task is a symbolic regression problem. Write a  
 $\hookrightarrow$  Python expression in `func()` where parameter scales are as similar  
 $\hookrightarrow$  as possible (use linear scaling or translation if needed). This  
 $\hookrightarrow$  helps later optimization when all parameters are initialized  
 $\hookrightarrow$  randomly in  $[0,1]$ .

Respond in the following format: <think>  
 $\dots$   
</think>  
<answer>  
 $\dots$   
</answer>.

```

2754
2755 Your task is to evolve a Python function func(x, params) that models
2756 → a scientific process, considering the physical meaning and
2757 → relationships of inputs, by predicting output variables based on
2758 → input variables.
2759
2760 The function signature is:
2761
2762 '''python
2763 def func(x: np.ndarray, params: np.ndarray) -> np.ndarray:
2764 '''
2765
2766 → 'x' is a 2D NumPy array of shape '(n_samples, 2)'
2767 → 'params' is a 1D NumPy array of up to 10 parameters
2768 → The function should return a 1D NumPy array of predictions with
2769 → shape '(n_samples,)'
2770
2771 **Current Problem:**
2772 Model the  $dP_dt$  (Population growth rate) using the input features:  $t$ 
2773 → (Time),  $P$  (Population at time  $t$ )
2774 Thus, 'x' contains 2 columns:  $t$  (Time),  $P$  (Population at time  $t$ ).
2775
2776 The initial version of func is a simple linear model. Parameters in
2777 → 'params' will be optimized externally using the BFGS algorithm
2778 → based on unseen training data.
2779
2780 Your objective is to evolve func to improve predictive performance
2781 → on unseen data. Aim for a balance between:
2782 → **Accuracy**: Lower mean squared error (MSE) on training data
2783 → **Simplicity**: Prefer concise, interpretable expressions
2784
2785 Model performance (score =  $-\log_{10}(\text{mse})$ ) will be evaluated on a
2786 → held-out dataset. Ensure the model is free of potential numerical
2787 → errors (e.g.,  $\log 0$ , division by 0).
2788 ## Current Program
2789 Status: Initial Program
2790
2791 '''python
2792 def func(x, params):
2793     """
2794         Calculates the model output using a linear combination of input
2795         → variables
2796         or a constant value if no input variables. Operates on a matrix of
2797         → samples.
2798
2799 Args:
2800
2801     x (np.ndarray): A 2D numpy array of input variable values,
2802     → shape (n_samples, n_features).
2803         n_features is 2.
2804         If n_features is 0, x should be shape
2805         → (n_samples, 0).
2806         The order of columns in x must correspond to:
2807         (t, P).
2808     params (np.ndarray): A 1D numpy array of parameters.
2809         Expected length: 10.
2810
2811 Returns:
2812
2813     np.ndarray: A 1D numpy array of predicted output values, shape
2814     → (n_samples,).
2815     """
2816
2817     result = x[:, 0] * params[0] + x[:, 1] * params[1]
2818     return result
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4038
4039
4040
4041
4042
4043
4044
4045
4046
4046
4047
4048
4049
4049
4050
4051
4052
4053
4054
4055
4055
4056
4057
4058
4058
4059
4059
4060
4061
4062
4062
4063
4063
4064
4064
4065
4065
4066
4066
4067
4067
4068
4068
4069
4069
4070
4070
4071
4071
4072
4072
4073
4073
4074
4074
4075
4075
4076
4076
4077
4077
4078
4078
4079
4079
4080
4080
4081
4081
4082
4082
4083
4083
4084
4084
4085
4085
4086
4086
4087
4087
4088
4088
4089
4089
4090
4090
4091
4091
4092
4092
4093
4093
4094
4094
4095
4095
4096
4096
4097
4097
4098
4098
4099
4099
4100
4100
4101
4101
4102
4102
4103
4103
4104
4104
4105
4105
4106
4106
4107
4107
4108
4108
4109
4109
4110
4110
4111
4111
4112
4112
4113
4113
4114
4114
4115
4115
4116
4116
4117
4117
4118
4118
4119
4119
4120
4120
4121
4121
4122
4122
4123
4123
4124
4124
4125
4125
4126
4126
4127
4127
4128
4128
4129
4129
4130
4130
4131
4131
4132
4132
4133
4133
4134
4134
4135
4135
4136
4136
4137
4137
4138
4138
4139
4139
4140
4140
4141
4141
4142
4142
4143
4143
4144
4144
4145
4145
4146
4146
4147
4147
4148
4148
4149
4149
4150
4150
4151
4151
4152
4152
4153
4153
4154
4154
4155
4155
4156
4156
4157
4157
4158
4158
4159
4159
4160
4160
4161
4161
4162
4162
4163
4163
4164
4164
4165
4165
4166
4166
4167
4167
4168
4168
4169
4169
4170
4170
4171
4171
4172
4172
4173
4173
4174
4174
4175
4175
4176
4176
4177
4177
4178
4178
4179
4179
4180
4180
4181
4181
4182
4182
4183
4183
4184
4184
4185
4185
4186
4186
4187
4187
4188
4188
4189
4189
4190
4190
4191
4191
4192
4192
4193
4193
4194
4194
4195
4195
4196
4196
4197
4197
4198
4198
4199
4199
4200
4200
4201
4201
4202
4202
4203
4203
4204
4204
4205
4205
4206
4206
4207
4207
4208
4208
4209
4209
4210
4210
4211
4211
4212
4212
4213
4213
4214
4214
4215
4215
4216
4216
4217
4217
4218
4218
4219
4219
4220
4220
4221
4221
4222
4222
4223
4223
4224
4224
4225
4225
4226
4226
4227
4227
4228
4228
4229
4229
4230
4230
4231
4231
4232
4232
4233
4233
4234
4234
4235
4235
4236
4236
4237
4237
4238
4238
4239
4239
4240
4240
4241
4241
4242
4242
4243
4243
4244
4244
4245
4245
4246
4246
4247
4247
4248
4248
4249
4249
4250
4250
4251
4251
4252
4252
4253
4253
4254
4254
4255
4255
4256
4256
4257
4257
4258
4258
4259
4259
4260
4260
4261
4261
4262
4262
4263
4263
4264
4264
4265
4265
4266
4266
4267
4267
4268
4268
4269
4269
4270
4270
4271
4271
4272
4272
4273
4273
4274
4274
4275
4275
4276
4276
4277
4277
4278
4278
4279
4279
4280
4280
4281
4281
4282
4282
4283
4283
4284
4284
4285
4285
4286
4286
4287
4287
4288
4288
4289
4289
4290
4290
4291
4291
4292
4292
4293
4293
4294
4294
4295
4295
4296
4296
4297
4297
4298
4298
4299
4299
4300
4300
4301
4301
4302
4302
4303
4303
4304
4304
4305
4305
4306
4306
4307
4307
4308
4308
4309
4309
4310
4310
4311
4311
4312
4312
4313
4313
4314
4314
4315
4315
4316
4316
4317
4317
4318
4318
4319
4319
4320
4320
4321
4321
4322
4322
4323
4323
4324
4324
4325
4325
4326
4326
4327
4327
4328
4328
4329
4329
4330
4330
4331
4331
4332
4332
4333
4333
4334
4334
4335
4335
4336
4336
4337
4337
4338
4338
4339
4339
4340
4340
4341
4341
4342
4342
4343
4343
4344
4344
4345
4345
4346
4346
4347
4347
4348
4348
4349
4349
4350
4350
4351
4351
4352
4352
4353
4353
4354
4354
4355
4355
4356
4356
4357
4357
4358
4358
4359
4359
4360
4360
4361
4361
4362
4362
4363
4363
4364
4364
4365
4365
4366
4366
4367
4367
4368
4368
4369
4369
4370
4370
4371
4371
4372
4372
4373
4373
4374
4374
4375
4375
4376
4376
4377
4377
4378
4378
4379
4379
4380
4380
4381
4381
4382
4382
4383
4383
4384
4384
4385
4385
4386
4386
4387
4387
4388
4388
4389
4389
4390
4390
4391
4391
4392
4392
4393
4393
4394
4394
4395
4395
4396
4396
4397
4397
4398
4398
4399
4399
4400
4400
4401
4401
4402
4402
4403
4403
4404
4404
4405
4405
4406
4406
4407
4407
4408
4408
4409
4409
4410
4410
4411
4411
4412
4412
4413
4413
4414
4414
4415
4415
4416
4416
4417
4417
4418
4418
4419
4419
4420
4420
4421
4421
4422
4422
4423
4423
4424
4424
4425
4425
4426
4426
4427
4427
4428
4428
4429
4429
4430
4430
4431
4431
4432
4432
4433
4433
4434
4434
4435
4435
4436
4436
4437
4437
4438
4438
4439
4439
4440
4440
4441
4441
4442
4442
4443
4443
4444
4444
4445
4445
4446
4446
4447
4447
4448
4448
4449
4449
4450
4450
4451
4451
4452
4452
4453
4453
4454
4454
4455
4455
4456
4456
4457
4457
4458
4458
4459
4459
4460
4460
4461
4461
4
```

## Prompt for Physics tasks

You are an expert software developer. Your job is to write a Python function based on feedback from previous attempts.

Write your code in exactly the following format:

```
```python
# your code
```
```

Your code's execution time is limited, so pay attention to runtime efficiency!

If you use new packages, please import them.

Ensure the program still contains the `func()` function and produces the same outputs; other functions can be added, deleted, or modified freely.

**IMPORTANT:** The current task is a symbolic regression problem. Write a Python expression in `func()` where parameter scales are as similar as possible (use linear scaling or translation if needed). This helps later optimization when all parameters are initialized randomly in  $[0,1]$ .

Respond in the following format: <think>

...

```
</think>
<answer>
...
</answer>.
```

Your task is to evolve a Python function `func(x, params)` that models a scientific process, considering the physical meaning and relationships of inputs, by predicting output variables based on input variables.

The function signature is:

```
```python
def func(x: np.ndarray, params: np.ndarray) -> np.ndarray
```
```

- `x` is a 2D NumPy array of shape `(n_samples, 3)`  
- `params` is a 1D NumPy array of up to 10 parameters  
- The function should return a 1D NumPy array of predictions with shape `(n_samples,)`

**\*\*Current Problem:\*\***

Model the `dv_dt` (Acceleration in Nonlinear Harmonic Oscillator) using the input features: `x` (Position at time `t`), `t` (Time), `v` (Velocity at time `t`)

Thus, `x` contains 3 columns: `x` (Position at time `t`), `t` (Time), `v` (Velocity at time `t`).

The initial version of `func` is a simple linear model. Parameters in `params` will be optimized externally using the BFGS algorithm based on unseen training data.

Your objective is to evolve `func` to improve predictive performance on unseen data. Aim for a balance between:

- \*\*Accuracy\*\*:** Lower mean squared error (MSE) on training data
- \*\*Simplicity\*\*:** Prefer concise, interpretable expressions

Model performance (`score = -log_10(mse)`) will be evaluated on a held-out dataset. Ensure the model is free of potential numerical errors (e.g., `log0`, division by 0).

**## Current Program**

Status: Initial Program

```
```python
def func(x, params):
```

```

2862
2863     """
2864     Calculates the model output using a linear combination of input
2865     ↪ variables
2866     or a constant value if no input variables. Operates on a matrix of
2867     ↪ samples.
2868
2869     Args:
2870         x (np.ndarray): A 2D numpy array of input variable values,
2871             ↪ shape (n_samples, n_features).
2872             n_features is 3.
2873             If n_features is 0, x should be shape
2874             ↪ (n_samples, 0).
2875             The order of columns in x must correspond to:
2876             (x, t, v).
2877             params (np.ndarray): A 1D numpy array of parameters.
2878                 Expected length: 10.
2879
2880             Returns:
2881                 np.ndarray: A 1D numpy array of predicted output values, shape
2882                 ↪ (n_samples,).
2883
2884             """
2885
2886             result = x[:, 0] * params[0] + x[:, 1] * params[1] + x[:, 2] *
2887             ↪ params[2]
2888             return result
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915

```

### Prompt for Material Science tasks

You are an expert software developer. Your job is to write a Python function based on feedback from previous attempts.

Write your code in exactly the following format:

```

```python
# your code
```

```

Your code's execution time is limited, so pay attention to runtime efficiency!

If you use new packages, please import them.

Ensure the program still contains the `func()` function and produces the same outputs; other functions can be added, deleted, or modified freely.

IMPORTANT: The current task is a symbolic regression problem. Write a Python expression in `func()` where parameter scales are as similar as possible (use linear scaling or translation if needed). This helps later optimization when all parameters are initialized randomly in  $[0,1]$ .

Respond in the following format: <think>

```

...
</think>
<answer>
...
</answer>.

```

Your task is to evolve a Python function `func(x, params)` that models a scientific process, considering the physical meaning and relationships of inputs, by predicting output variables based on input variables.

The function signature is:

```

```python
def func(x: np.ndarray, params: np.ndarray) -> np.ndarray:
```

```

```
2916
2917 - `x` is a 2D NumPy array of shape `(n_samples, 2)`
2918 - `params` is a 1D NumPy array of up to 10 parameters
2919 - The function should return a 1D NumPy array of predictions with
2920   ↳ shape `(n_samples,)`  

2921 **Current Problem:**
2922 Model the sigma (Stress) using the input features: epsilon (Strain), T
2923   ↳ (Temperature)
2924 Thus, `x` contains 2 columns: epsilon (Strain), T (Temperature).  

2925 The initial version of `func` is a simple linear model. Parameters in
2926   ↳ `params` will be optimized externally using the BFGS algorithm
2927   ↳ based on unseen training data.  

2928 Your objective is to evolve `func` to improve predictive performance
2929   ↳ on unseen data. Aim for a balance between:
2930 - **Accuracy**: Lower mean squared error (MSE) on training data
2931 - **Simplicity**: Prefer concise, interpretable expressions  

2932 Model performance (score = -log_10(mse)) will be evaluated on a
2933   ↳ held-out dataset. Ensure the model is free of potential numerical
2934   ↳ errors (e.g., log0, division by 0).  

2935 ## Current Program
2936 Status: Initial Program
2937 ```python
2938 def func(x, params):
2939   """
2940     Calculates the model output using a linear combination of input
2941     ↳ variables
2942     or a constant value if no input variables. Operates on a matrix of
2943     ↳ samples.
2944
2945     Args:
2946       x (np.ndarray): A 2D numpy array of input variable values,
2947         ↳ shape (n_samples, n_features).
2948           n_features is 2.
2949           If n_features is 0, x should be shape
2950             ↳ (n_samples, 0).
2951           The order of columns in x must correspond to:
2952             (epsilon, T).
2953       params (np.ndarray): A 1D numpy array of parameters.
2954           Expected length: 10.
2955
2956     Returns:
2957       np.ndarray: A 1D numpy array of predicted output values, shape
2958         ↳ (n_samples,).
2959   """
2960
2961   result = x[:, 0] * params[0] + x[:, 1] * params[1]
2962   return result
2963 ```
```

## E METHODOLOGICAL CHALLENGES AND COMPARATIVE ANALYSIS OF RL-EA INTEGRATION

This appendix details the specific technical challenges associated with integrating Reinforcement Learning (RL) and Evolutionary Algorithms (EAs). We further analyze why a naive sequential combination (e.g., AlphaEvolve) fails to scale effectively compared to the proposed HELIX framework, supported by empirical evidence from the Circle Packing problem.

2970 E.1 TECHNICAL CHALLENGES AND SOLUTIONS  
29712972 The integration of RL and EAs presents several non-trivial challenges, primarily stemming from the  
2973 conflicting objectives and operational domains of the two paradigms. HELIX addresses these as  
2974 follows:2975 **Goal Mismatch and Unification.** A fundamental disconnect exists between RL, which learns a  
2976 policy mapping states to actions, and EAs, which act as population-based optimization methods  
2977 relying on recombination and mutation. Integrating these requires a principled bridge rather than a  
2978 naive combination.2979

- **In-Context Learning as a Bridge:** HELIX adopts an in-context learning paradigm where  
2980 previously discovered high-quality solutions are injected into the prompt as explicit mem-  
2981 ory. This transforms the Large Language Model (LLM) into a *parameterized mutation*  
2982 *operator*, conditioned on historical trajectories.
- **Unified Optimization:** We employ Group Relative Policy Optimization (GRPO) to train  
2983 this mutation operator. GRPO naturally generates diverse rollouts that serve as the popu-  
2984 lation for evolutionary selection. Consequently, policy optimization (RL) and evolution-  
2985 ary search (EA) are coupled within a closed loop: test-time scaling via evolution provides high-  
2986 quality data for RL, while RL improves the mutation operator for subsequent evolution-  
2987 ary steps.

2989

**Diversity Estimation in Giant Code Spaces.** Traditional evolutionary metrics are ill-suited for  
2990 code optimization, where the action space is discrete, high-dimensional, and highly structured.  
2991 Measuring individual diversity in this domain is challenging. HELIX resolves this by utilizing an  
2992 embedding-based approach to quantify semantic distances between code individuals. We compute  
2993 population diversity via k-nearest neighbors (kNN) in this embedding space, providing a scalable  
2994 and semantically meaningful metric to guide selection.

## 2996 E.2 LIMITATIONS OF NAIVE INTEGRATION: A CASE STUDY

2997 To demonstrate why HELIX offers a necessary advancement over "naive" integration, we compare  
2998 it against the AlphaEvolve paradigm. AlphaEvolve represents a sequential approach: post-training  
2999 an LLM on general domains followed by applying evolutionary algorithms to downstream tasks  
3000 without further policy updates.3001 We conducted a comparative experiment on the *Circle Packing* problem (maximizing the sum of  
3002 radii for 26 non-overlapping circles in a unit square). We evaluated Direct Prompting (BO64),  
3003 OpenEvolve (an open-source reproduction of AlphaEvolve), and HELIX using Qwen 14B and 32B  
3004 base models.

3005 Table 2: Performance Comparison on Circle Packing Task

3006

| 3007 <b>Method</b>            | 3008 <b>Score (Sum of Radii)</b> |
|-------------------------------|----------------------------------|
| 3009 Direct Prompt (Qwen 14B) | 3010 1.673                       |
| 3011 OpenEvolve (Qwen 14B)    | 3012 1.586                       |
| 3013 OpenEvolve (Qwen 32B)    | 3014 1.956                       |
| <b>3015 HELIX (Qwen 14B)</b>  | <b>3016 2.636</b>                |

3017 As shown in Table 2, OpenEvolve with Qwen 14B performed worse than the Direct Prompt baseline,  
3018 despite utilizing significantly more computational resources. Our analysis identifies two critical  
3019 failure modes in naive integration of OpenEvolve:3020 **1. Constraints of Initialization Bias.** Naive approaches are heavily constrained by their initial  
3021 seed solutions. OpenEvolve generates a small set of seed trials (e.g., 5) and iterates upon them.  
3022 If these initial trials lack diversity or occupy a low-performance region, the evolutionary process  
3023 stagnates in local optima. In contrast, Direct Prompting (BO64) benefits from 64 i.i.d. evaluations,  
offering a broader initial coverage that the naive evolutionary process failed to surpass.

3024  
 3025 **2. Rejection of Novelty and Destructive Changes.** A more subtle failure mode is the re-  
 3026 jection of potentially high-reward strategies that require initial “destructive” changes. In our  
 3027 Qwen 32B OpenEvolve experiment (4,147 trials), the model predominantly attempted to ad-  
 3028 just coordinates directly. Only 12 trials (0.3%) attempted a radically different approach using  
 3029 `scipy.optimize.minimize`.

3030 

- 3031 • **The Failure:** All 12 trials initially yielded a reward of 0.0 due to minor compilation errors  
 3032 or timeouts. Traditional evolutionary selection, driven by immediate reward or superficial  
 3033 code features (e.g., length), discarded these candidates.
- 3034 • **The Consequence:** The system failed to explore the `scipy` approach, which—once de-  
 3035 bugged—is capable of yielding scores  $> 2.0$ .

3036 **E.3 THE HELIX ADVANTAGE**

3037 HELIX overcomes the aforementioned limitations through two specific mechanisms:

3038 

- 3039 1. **Explicit Diversity Accounting:** By using an embedding model to distinguish methods  
 3040 semantically, HELIX assigns a high *Diversity Score* to the rare `scipy`-based solutions,  
 3041 even when their immediate reward is low. This ensures they are retained in the population  
 3042 for further mutation/debugging.
- 3043 2. **Parameter Learning via RL:** Once a diversity-preserved rollout successfully fixes the  
 3044 implementation bug (generating a high-reward solution  $s_{t+1}^*$ ), HELIX utilizes this trajec-  
 3045 tory for RL updates. This update increases the probability of the policy generating similar  
 3046 sophisticated methods in future steps.

3047 This establishes a positive feedback loop: diversity metrics preserve potential innovation, and RL  
 3048 consolidates successful realizations of that innovation into the model parameters, allowing HELIX  
 3049 to break out of local optima where naive methods stagnate.

3050 **F THEORETICAL ANALYSIS OF THE FRAMEWORK**

3051 In this section, we employ a simplified mathematical model to provide theoretical insights into the  
 3052 advantages of our algorithm in solving complex scientific problems. We demonstrate the efficiency  
 3053 of HELIX in discovering optimal solutions compared to baseline methods.

3054 **F.1 MATHEMATICAL SETUP AND PRELIMINARIES**

3055 First, we establish the geometric and probabilistic foundations of the problem. For a given problem  
 3056  $q$ , we assume the existence of the following structures:

3057 

- 3058 • **Solution Space:** A set of solutions  $\mathcal{S}$ , which can be viewed as a simply connected open  
 3059 manifold in a complex space.
- 3060 • **Reward Function:** A continuous function  $R : \mathcal{S} \rightarrow \mathbb{R}^+$ .
- 3061 • **Embedding:** A mapping  $\Phi : \mathcal{S} \rightarrow \mathbb{R}^n$  that maps the solution space to an Embedding Space  
 3062  $\mathbb{R}^n$ , satisfying:
  - 3063 1. **Continuity:** For any  $s_1, s_2 \in \mathcal{S}$  derived via similar methods, their embeddings  $v_1 = \Phi(s_1)$  and  $v_2 = \Phi(s_2)$  are adjacent in  $\mathbb{R}^n$ .
  - 3064 2. **Injectivity:** Distinct solutions have distinct embeddings.
  - 3065 3. **Open Map:**  $\Phi$  maps open sets in  $\mathcal{S}$  to open sets in  $\mathbb{R}^n$ .

3066 We define the reward function in the embedding space  $\mathbb{R}^n$  as follows. For any  $v \in \mathbb{R}^n$ :

$$3067 r(v) = \begin{cases} R(\Phi^{-1}(v)) & v \in \Phi(\mathcal{S}) \\ 3068 0 & v \notin \Phi(\mathcal{S}) \end{cases}, \quad (29)$$

3069 where  $s = \Phi^{-1}(v)$  is the solution corresponding to  $v$ . Restricted to the image set,  $\Phi : \mathcal{S} \rightarrow \Phi(\mathcal{S})$  is  
 3070 a bijection, making its inverse well-defined. Given the continuity of  $\Phi$  and  $R$ , and the open mapping  
 3071 property of  $\Phi$ ,  $r(v)$  is continuous.

3078  
 3079  
 3080  
**Definition F.1** (LLM Transition Process). For a solution  $s$ , an LLM parameterized by  $\theta$  transforms  
 the solution by outputting an action  $a \sim \pi_\theta(\cdot|s)$ , resulting in  $s' = T(s, a)$ . Based on this, we define  
 the LLM Transition Function on  $\mathbb{R}^n$  as a stochastic process:

3081  
 3082  $T_\theta^* : \mathbb{R}^n \rightarrow (\Omega \rightarrow \mathbb{R}^n), \quad T_\theta^*(v) = \Phi(T(\Phi^{-1}(v), a)), \quad \text{where } a \sim \pi_\theta(\cdot|\Phi^{-1}(v)). \quad (30)$

3083 For tractability, we approximate  $T_\theta^*$  as an independent Normal distribution:  
 3084

3085  $T_\theta^*(v) \sim \mathcal{N}(v + \delta_\theta(v), \sigma \mathbf{I}). \quad (31)$   
 3086

3087 This implies each transition follows  $v \rightarrow v + \delta_\theta(v) + \xi$ , where  $\xi \sim \mathcal{N}(0, \sigma \mathbf{I})$ . This Gaussian  
 3088 approximation is justified as LLMs typically generate modest modifications to the current solution,  
 3089 making local approximations valid in the embedding space.

3090  
 3091 **F.2 THEORETICAL ANALYSIS OF GRPO**

3092  
 3093 **F.2.1 SETUP AND ASSUMPTIONS**

3094 In the GRPO method, since the prompt is fixed, the model evolves solely from an initial solution  
 3095  $v_0$ . The transition samples from  $\mathcal{N}(v_0 + \delta_\theta(v_0), \sigma \mathbf{I})$ . GRPO estimates the gradient of the reward  
 3096 function near  $v = v_0 + \delta_\theta(v_0)$  and updates the model parameters. The effective update dynamics in  
 3097 the embedding space can be described as:

3098  
 3099  $\delta_\theta(v_0) \leftarrow \delta_\theta(v_0) + \eta \nabla_v r(v_0 + \delta_\theta(v_0)), \quad (32)$

3100 which simplifies to the gradient ascent process  $v \leftarrow v + \eta \nabla_v r(v)$ , where  $\eta$  is the learning rate.

3102 To analyze convergence, we introduce the following assumption regarding the reward landscape.

3103 **Assumption F.2** (Reward Landscape). We assume the reward function  $r(v)$  consists of two Gaus-  
 3104 sian peaks, representing a local optimum ( $v_{loc}$ ) and a global optimum ( $v_{opt}$ ):

3105  
 3106  $r(v) = A_{loc} \exp\left(-\frac{\|v - v_{loc}\|^2}{2w^2}\right) + A_{opt} \exp\left(-\frac{\|v - v_{opt}\|^2}{2w^2}\right). \quad (33)$   
 3107

3108 Let  $L = \|v_{opt} - v_{loc}\|$  be the distance between the optima.

3109 **Theorem F.3** (Convergence to Local Optimum of GRPO). *Let  $v_0$  be the initial solution for GRPO.  
 3110 GRPO will converge to the local optimum near  $v_{loc}$  if the following conditions are met:*

3112 1. **Separation:**  $L > 2w$ . There is sufficient separation between the global and local optima.  
 3113  
 3114 2. **Amplitude:**  $A_{loc} > A_{opt} \cdot \frac{L}{w} \cdot \exp\left(-\frac{L^2}{2w^2}\right)$ . The local optimum is not significantly weaker  
 3115 than the global optimum locally.  
 3116  
 3117 3. **Initialization:** Decomposing the initial solution as  $v_0 = v_{loc} + v_\perp + \gamma_0(v_{opt} - v_{loc})$ , where  
 3118  $v_\perp \perp (v_{opt} - v_{loc})$ , we require  $\gamma_0 < \gamma_{barrier} \approx \frac{1}{2} - \frac{w^2}{L^2} \ln \frac{A_{opt}}{A_{loc}}$ . This implies the initial  
 3119 solution is geometrically closer to the local optimum's basin of attraction.  
 3120

3121 *Comment.* These assumptions hold in many scientific problems where distinct methods (local vs.  
 3122 global) have a large semantic gap ( $L > 2w$ ), and initial "naive" solutions naturally fall closer to  
 3123 simpler local optima. This illustrates that GRPO, lacking memory or population mechanisms, is  
 3124 prone to trapping in local optima.

3125  
 3126 **F.2.2 PROOF OF THEOREM F.3**

3127 We decompose the gradient of  $r(v)$ . Let  $v$  be parameterized as  $v = v_{loc} + v_\perp + \gamma(v_{opt} - v_{loc})$ . The  
 3128 gradient  $\nabla r(v)$  satisfies:  
 3129

3130  $\nabla r(v) = \frac{A_{loc}}{w^2} (v_{loc} - v) \exp\left(-\frac{\|v - v_{loc}\|^2}{2w^2}\right) + \frac{A_{opt}}{w^2} (v_{opt} - v) \exp\left(-\frac{\|v - v_{opt}\|^2}{2w^2}\right). \quad (34)$   
 3131

3132 Projecting onto the line connecting the optima ( $v_{opt} - v_{loc}$ ):  
 3133

$$\begin{aligned} 3134 \quad \langle \nabla r(v), v_{opt} - v_{loc} \rangle &= -\frac{A_{loc}L^2}{w^2}\gamma \exp\left(-\frac{\gamma^2L^2 + \|v_{\perp}\|^2}{2w^2}\right) \\ 3135 \quad &+ \frac{A_{opt}L^2}{w^2}(1-\gamma) \exp\left(-\frac{(1-\gamma)^2L^2 + \|v_{\perp}\|^2}{2w^2}\right). \end{aligned} \quad (35)$$

3139 Projecting onto the perpendicular component  $v_{\perp}$ :  
 3140

$$\langle \nabla r(v), v_{\perp} \rangle = -\frac{\|v_{\perp}\|^2}{w^2} \left( A_{loc} \left( -\frac{\|v - v_{loc}\|^2}{2w^2} \right) + A_{opt} \left( -\frac{\|v - v_{opt}\|^2}{2w^2} \right) \right). \quad (36)$$

3143 First, analyzing the dynamics of  $v_{\perp}$ :  
 3144

$$\frac{d}{dt} \|v_{\perp}\|^2 = 2\langle \dot{v}_{\perp}, v_{\perp} \rangle \propto -C(v) \|v_{\perp}\|^2 < 0. \quad (37)$$

3147 Regardless of initialization,  $v_{\perp}$  decays exponentially to 0. The system converges to the linear manifold connecting  $v_{loc}$  and  $v_{opt}$ . Assuming  $v_{\perp} = 0$ , the dynamics of  $\gamma$  are governed by:  
 3148

$$\frac{d\gamma}{dt} \propto -\gamma A_{loc} \exp\left(-\frac{\gamma^2L^2}{2w^2}\right) + (1-\gamma) A_{opt} \exp\left(-\frac{(1-\gamma)^2L^2}{2w^2}\right). \quad (38)$$

3152 Solving for equilibrium points ( $\frac{d\gamma}{dt} = 0$ ) yields a stable local equilibrium near  $\gamma \approx 0$ , an unstable saddle point  $\gamma_{barrier} \approx \frac{1}{2} - \frac{w^2}{L^2} \ln \frac{A_{opt}}{A_{loc}}$ , and a stable global equilibrium near  $\gamma \approx 1$ . If  $\gamma_0 < \gamma_{barrier}$ ,  
 3153 the system flows to the local optimum.  $\square$   
 3154

### 3156 F.3 THEORETICAL ANALYSIS OF EVOLVE AND HELIX

#### 3158 F.3.1 SETUP: UNIFIED DRIFT-DIFFUSION AND SELECTION FRAMEWORK

3160 We analyze the iterative processes of Evolve and HELIX by modeling them as continuous-time  
 3161 stochastic processes. Both algorithms maintain a population  $\mathcal{P}$  and update it via  $v'$ .  
 3162

- 3163 • **Evolve (Selection-Diffusion):** In the Evolve algorithm, the model parameters cannot be  
 3164 adjusted. Thus, we assume the model has no inherent directional bias towards different  
 3165 methods of this specific problem ( $\delta_{\theta}(v) \equiv 0$ ). The iteration simplifies to a random walk  
 3166  $v' = v + \sigma\xi$ . At each step, a solution  $v$  is drawn from  $\mathcal{P}$ , and  $v' = v + \sigma\xi$  is generated.  
 3167 Critically, solutions with higher Reward are sampled with a higher probability. We can  
 3168 model this selection by a weight function  $w(v) = \exp(\alpha r(v))$ , where  $\alpha$  represents the  
 3169 selection pressure. The new solution is added to the population:  $\mathcal{P} \leftarrow \mathcal{P} \cup \{v'\}$ .  
 3170
- 3171 • **HELIX (Drift-Diffusion):** HELIX maintains a population  $\mathcal{P}$  and dynamically adjusts the  
 3172 directional bias  $\delta_{\theta}(v)$ . Through the GRPO mechanism, this direction will gradually approx-  
 3173 imate the gradient  $\nabla r(v)$ . Upon sufficient convergence, the HELIX iteration approximates  
 3174 a guided random walk:  $v' = v + \eta \nabla r(v) + \sigma\xi$ . In HELIX, we also sample high-Reward  
 3175 solutions with higher probability, but for mathematical tractability, we assume the selection  
 3176 weight parameter  $\alpha = 0$ , meaning the sampling weight is uniform ( $w(v) \equiv 1$ ).  
 3177

3176 The comparison is summarized in Table 3.  
 3177

3178 Table 3: Comparison of Algorithm Dynamics  
 3179

| 3180 <b>Algorithm</b> | 3181 <b>Dynamics Equation</b>           | 3182 <b>Drift <math>D(v)</math></b> | 3183 <b>Selection <math>w(v)</math></b> |
|-----------------------|-----------------------------------------|-------------------------------------|-----------------------------------------|
| 3182 Evolve           | $v' = v + \sigma\xi$                    | $0$                                 | $\exp(\alpha r(v))$                     |
| 3183 HELIX            | $v' = v + \eta \nabla r(v) + \sigma\xi$ | $\eta \nabla r(v)$                  | $1 (\alpha = 0)$                        |

3184 **Theorem F.4** (Stationary Distribution). *Assuming the solution space is bounded, as  $t \rightarrow \infty$ , the  
 3185 population distribution  $p^*(\mathbf{v})$  converges to:*

3186 1. **Evolve**: Converges to the principal eigenfunction of the associated Schrödinger operator.  
 3187 Under the WKB approximation ( $\sigma \rightarrow 0$ ):

3188

$$3189 p_{Evo}^*(\mathbf{v}) \asymp \exp \left( -\frac{\sqrt{2\alpha}}{\sigma} \int_{v_{opt}}^v \sqrt{r(v_{opt}) - r(u)} du \right). \quad (39)$$

3190

3191 2. **HELIX**: Converges to a Boltzmann-Gibbs Measure:

3192

$$3193 p_{Helix}^*(\mathbf{v}) \propto \exp \left( \frac{2\eta}{\sigma^2} r(\mathbf{v}) \right). \quad (40)$$

3194

3195 **Comment on theorem F.4.**

3196 1. **Concentration and  $\sigma$  Scaling.** The concentration power of the stationary distributions—defined  
 3197 as the inverse of their variance—exhibits distinct scaling behaviors with respect to the noise pa-  
 3198 rameter  $\sigma$ . Specifically, the concentration scales as  $\mathcal{O}(1/\sigma)$  for Evolve and  $\mathcal{O}(1/\sigma^2)$  for HELIX.  
 3199 Given that  $\sigma \ll 1$  in high-precision search contexts, it follows that  $1/\sigma^2 \gg 1/\sigma$ . This inequality  
 3200 indicates that the sampling distribution of HELIX is exponentially more concentrated around the  
 3201 optimum than that of Evolve. Under identical environmental conditions, HELIX achieves a signifi-  
 3202 cantly more exhaustive exploration of the highly rewarded vicinity of the optimal solution.

3203 2. **Intuitive Comparison (Quadratic Reward).** To provide a concrete comparison, we analyze  
 3204 the behavior under a local quadratic approximation of the reward function,  $r(v) = r_{opt} - \frac{k}{2}\|v\|^2$   
 3205 (centered at  $v_{opt} = 0$ ). Deriving the exact Gaussian forms of the stationary distributions allows for  
 3206 a direct comparison of their variances, as summarized in Table 4.

3207 Table 4: Comparison of Stationary Distributions under Quadratic Reward

3208

| Algorithm     | Gaussian Form $p^*(\mathbf{v})$                                        | Variance $\Sigma^2$                               | Scaling vs. $\sigma$         |
|---------------|------------------------------------------------------------------------|---------------------------------------------------|------------------------------|
| <b>HELIX</b>  | $\propto \exp \left( -\frac{\eta k}{\sigma^2} \ v\ ^2 \right)$         | $\Sigma_{Helix}^2 = \frac{\sigma^2}{2\eta k}$     | $\propto \sigma^2$ (Sharper) |
| <b>Evolve</b> | $\propto \exp \left( -\frac{\sqrt{\alpha k}}{2\sigma} \ v\ ^2 \right)$ | $\Sigma_{Evo}^2 = \frac{\sigma}{\sqrt{\alpha k}}$ | $\propto \sigma$ (Broader)   |

3209 The ratio of their variances is given by:

3210

$$\frac{\Sigma_{Helix}^2}{\Sigma_{Evo}^2} = \frac{\sigma^2/2\eta k}{\sigma/\sqrt{\alpha k}} = \frac{\sqrt{\alpha}}{2\eta\sqrt{k}} \cdot \sigma \propto \sigma. \quad (41)$$

3211

3212 As  $\sigma \rightarrow 0$ , this ratio tends to zero. This rigorously confirms that HELIX’s mechanism—utilizing the  
 3213 gradient for directional movement—provides a superior capacity for stabilizing and concentrating  
 3214 the population compared to Evolve’s reliance on scalar selection alone.

3215 3. **Potential for Further Reinforcement.** It is worth noting that the current analysis assumes a  
 3216 uniform selection weight for HELIX ( $\alpha = 0$ ). If we were to incorporate a non-trivial selection  
 3217 weight  $w(v) = \exp(\alpha r(v))$  into the HELIX framework, the final stationary distribution would  
 3218 theoretically become even more concentrated. Although a quantitative closed-form solution for  
 3219 this combined Drift-Diffusion-Selection process is mathematically intractable, qualitative analysis  
 3220 suggests that this would further reinforce HELIX’s focus and exploitation capabilities within high-  
 3221 reward regions.

3222 F.3.2 PROOF OF THEOREM F.4

3223 **Part I: HELIX (Drift-Diffusion).** The dynamics follow the Langevin Equation  $d\mathbf{v}_t = \eta \nabla r(\mathbf{v}_t) dt +$   
 3224  $\sigma d\mathbf{W}_t$ . The probability density  $p(\mathbf{v}, t)$  evolves via the Fokker-Planck equation:

3225

$$\frac{\partial p}{\partial t} = -\nabla \cdot (p \cdot \eta \nabla r(\mathbf{v})) + \frac{\sigma^2}{2} \nabla^2 p. \quad (42)$$

3226

3240 At steady state ( $\partial p / \partial t = 0$ ), the probability flux  $\mathbf{J}$  vanishes:  
 3241

$$3242 \quad \mathbf{J} = \eta p^* \nabla r(\mathbf{v}) - \frac{\sigma^2}{2} \nabla p^* = \mathbf{0} \implies \frac{\nabla p^*}{p^*} = \frac{2\eta}{\sigma^2} \nabla r(\mathbf{v}). \quad (43)$$

3244 Integrating both sides yields  $\ln p^*(\mathbf{v}) = \frac{2\eta}{\sigma^2} r(\mathbf{v}) + C$ , confirming the Boltzmann distribution.  
 3245

3246 **Part II: Evolve (Selection-Diffusion).** The discrete selection-mutation process converges to the  
 3247 Replicator-Mutator Equation in continuous time:

$$3248 \quad \frac{\partial p}{\partial t} = \frac{\sigma^2}{2} \nabla^2 p + \alpha (r(v) - \bar{r}) p. \quad (44)$$

3251 The stationary distribution  $p^*$  satisfies the Schrödinger-like equation (where  $E = \alpha \bar{r}$ ):  
 3252

$$3253 \quad \frac{\sigma^2}{2} \nabla^2 p^* + \alpha r(v) p^* = E p^*. \quad (45)$$

3254 Using the WKB Ansatz  $p^*(v) = C(v) \exp(-S(v)/\sigma)$ , and substituting into the equation, the lead-  
 3255 ing order terms ( $\sigma \rightarrow 0$ ) yield the Hamilton-Jacobi equation:  
 3256

$$3257 \quad \frac{1}{2} \|\nabla S\|^2 + \alpha r(v) \approx E. \quad (46)$$

3259 Setting the ground state condition at  $v_{opt}$  gives  $E = \alpha r(v_{opt})$ . Solving for  $\nabla S$ :  
 3260

$$3261 \quad \|\nabla S(v)\| = \sqrt{2\alpha(r(v_{opt}) - r(v))}. \quad (47)$$

3263 Integrating along the path from  $v_{opt}$  gives the action  $S(v)$ , yielding the final asymptotic form for  
 3264  $p_{Evo}^*$ .  $\square$   
 3265

## 3266 G FORMALIZED ALGORITHM

3268 In this appendix, we provide the detailed procedural description of the HELIX framework. Al-  
 3269 gorithm 1 summarizes the full workflow, including sampling, prompt construction, model rollout,  
 3270 reinforcement learning updates, diversity estimation, and evolutionary population selection. These  
 3271 details complement the main text and offer a complete specification of the method.  
 3272

## 3273 H EXAMPLE OF MODEL OUTPUT

3275 We present examples of the best solutions found by our framework across different task categories.  
 3276 These visualizations highlight how HELIX generates high-quality and interpretable outputs in di-  
 3277 verse scientific domains.  
 3278

### 3279 H.1 PHYSICS SIMULATION TASKS

3281 **Acoustic demultiplexer.** Figure 14 displays the acoustic pressure field of our best-performing  
 3282 demultiplexer, which achieves a reward of 14.260.  
 3283

3284 **Iron core torque optimization.** The best iron core design is shown in Figure 15, where the mag-  
 3285 netic flux density norm reaches a reward of 11.045.  
 3286

3287 **Beam design.** Figure 16 illustrates the von Mises stress pattern of the best beam structure discov-  
 3288 ered, which achieves a reward of 17.298.  
 3289

3290 **Meta-material optimization.** The temperature distributions of the optimized meta-material under  
 3291 two loading conditions are presented in Figure 17, yielding a reward of 1.278.  
 3292

3293 **Inductor design.** The optimized inductor is visualized in Figure 18, with a magnetic flux density  
 3294 norm field corresponding to a reward of 9.609.

---

**Algorithm 1** HELIX Framework

---

**Require:** Problem description  $p$ ; initial solution(s)  $s_0$ ; batch size  $B$ ; GRPO group size  $G$ ; number of samples in prompt  $n$ ; transition function  $T$ ; reward function  $R$ ; feedback function  $F$ ; embedding model  $E$ .

1: Initialize dataset  $\mathcal{D}_0 = \{s_0\}$ .  
 2: Initialize population  $\mathcal{P}_0 = \mathcal{D}_0$ .  
 3: Initialize policy model parameters  $\theta$ .  
 4: **for** iteration  $t = 0, 1, 2, \dots$  **do**

5:   Sample  $B$  solutions from  $\mathcal{P}_t$ , obtaining  $\{s_{t,i}\}_{i=1}^B$ . ▷ Prompt Construction  
 6:   **for**  $i = 1$  to  $B$  **do**  
 7:     Retrieve  $n$  ancestral states of  $s_{t,i}$ :  $\{f^{(k)}(s_{t,i})\}_{k=1}^{n-1}$ .  
 8:     Construct prompt:  

$$q_i = \text{ConstructPrompt}(\{p\} \cup \{s_{t,i}, R(s_{t,i}), F(s_{t,i})\} \cup \{f^{(k)}(s_{t,i}), R(f^{(k)}(s_{t,i})), F(f^{(k)}(s_{t,i}))\}_{k=1}^{n-1}).$$

9:   **end for**  
 10:   **for**  $i = 1$  to  $B$  **do** ▷ Model Rollout and Evaluation  
 11:     **for**  $j = 1$  to  $G$  **do**  
 12:       Generate action  $a_{i,j} \sim \pi_\theta(\cdot | q_i)$ .  
 13:       Obtain new solution  $s_{t+1,i,j} = T(s_{t,i}, a_{i,j})$ .  
 14:       Evaluate reward  $r_{t+1,i,j} = R(s_{t+1,i,j})$ .  
 15:       Record feedback  $f_{t+1,i,j} = F(s_{t+1,i,j})$ .  
 16:     **end for**  
 17:   **end for**  
 18:   **for**  $i = 1$  to  $B$  **do** ▷ Reinforcement Learning Update  
 19:     **for**  $j = 1$  to  $G$  **do**  
 20:       Compute normalized advantage:  

$$\tilde{r}_{t+1,i,j} = \frac{r_{t+1,i,j} - \text{mean}_j\{r_{t+1,i,j}\}}{\text{std}_j\{r_{t+1,i,j}\}}.$$

21:     **end for**  
 22:   **end for**  
 23:   Update policy:  $\theta \leftarrow \theta - \gamma \cdot \nabla_\theta \mathcal{L}_{\text{GRPO}}$ .

24:   **for**  $i = 1$  to  $B$  **do** ▷ Diversity Estimation  
 25:     **for**  $j = 1$  to  $G$  **do**  
 26:       Compute embedding  $h_{t+1,i,j} = E(s_{t+1,i,j})$ .  
 27:       Compute diversity score  $\text{Div}(s_{t+1,i,j})$  (as in Eq. (6)).  
 28:     **end for**  
 29:   **end for**  
 30:   Update dataset:  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{s_{t+1,i,j}\}$ . ▷ Population Update  
 31:   Use NSGA-II to select next population:  

$$\mathcal{P}_{t+1} = \text{SelectTop}_{\text{NSGA-II}} \left( \bigcup_{0 \leq s \leq t+1} \mathcal{D}_s \right).$$

32: **end for**

## H.2 CIRCLE PACKING TASKS

**Packing in square.** As shown in Figure 19, our framework successfully packs 26 circles inside a square, achieving a sum of radii of 2.6359830849 and surpassing the previous world record.

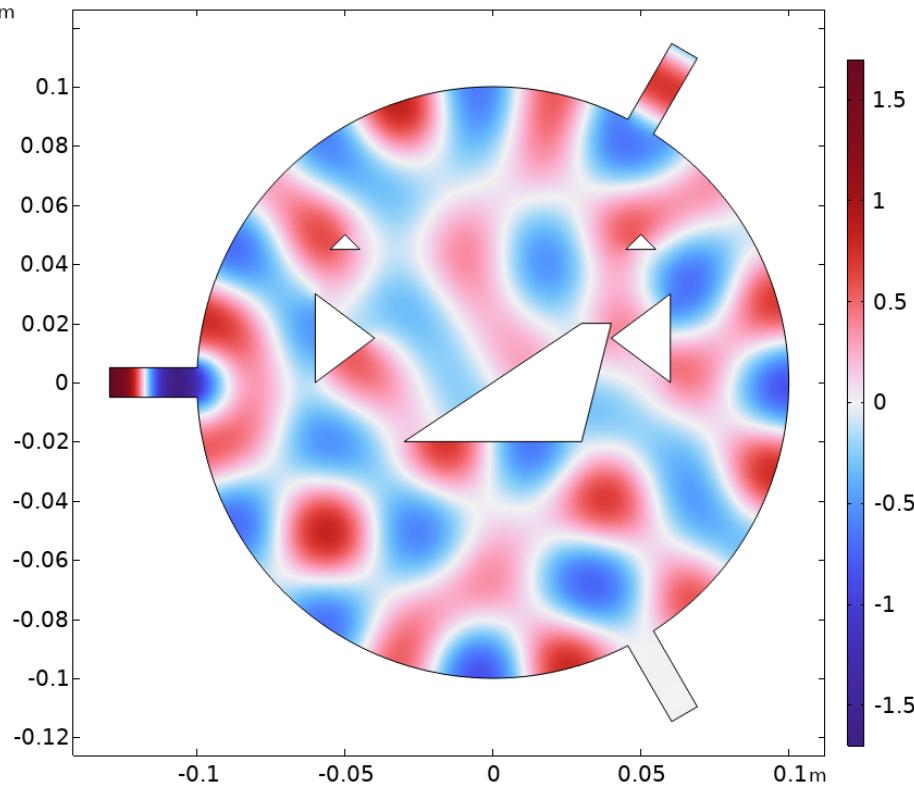


Figure 14: Acoustic pressure distribution in the optimal acoustic demultiplexer design obtained by our framework, with reward 14.260.

**Packing in disk.** Figure 20 demonstrates the packing of 26 circles inside a disk, reaching a total radius sum of 4.664465.

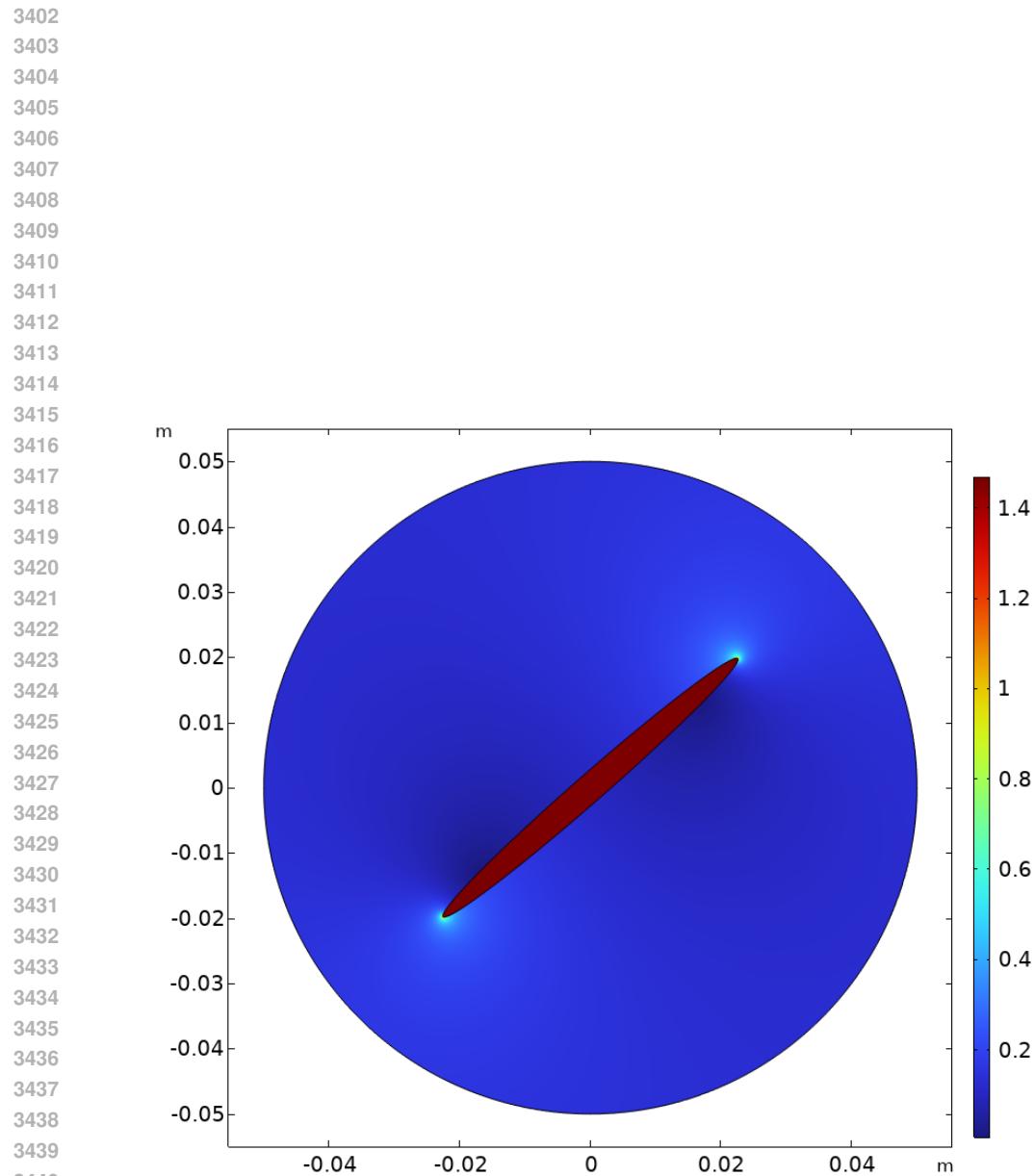
### H.3 MACHINE LEARNING TASKS

We further demonstrate how our framework can be applied to classical machine learning problems, using both classification and regression benchmarks. The first example focuses on the Adult dataset, where we design a rich set of engineered features that combine polynomial transformations, ratios, interaction terms, and domain-specific indicators. This structured feature space, coupled with a LightGBM classifier and hyperparameter tuning, enables our model to achieve a strong performance of 82.07 in macro F1 score (Figure 21).

For regression, we turn to the Boston Housing dataset. Here, we integrate robust preprocessing with advanced feature transformations. Missing values in numeric features are imputed with KNN and scaled robustly, while categorical variables undergo smoothed target encoding. Additional interaction and polynomial features are then injected through a transformer pipeline. With these enhancements, our model coupled with an XGBoost regressor attains a reward of 1.742, corresponding to an RMSE of 1.813 (Figure 22).

### I LLM USAGE STATEMENT

Large Language Models (LLMs) were used solely to aid writing and polishing the manuscript. All research ideas, experiments, and analyses were conceived and conducted by the authors, who take full responsibility for the content.



3441 Figure 15: Magnetic flux density norm field for the optimized iron core configuration identified by  
3442 our framework, achieving reward 11.045.

3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455

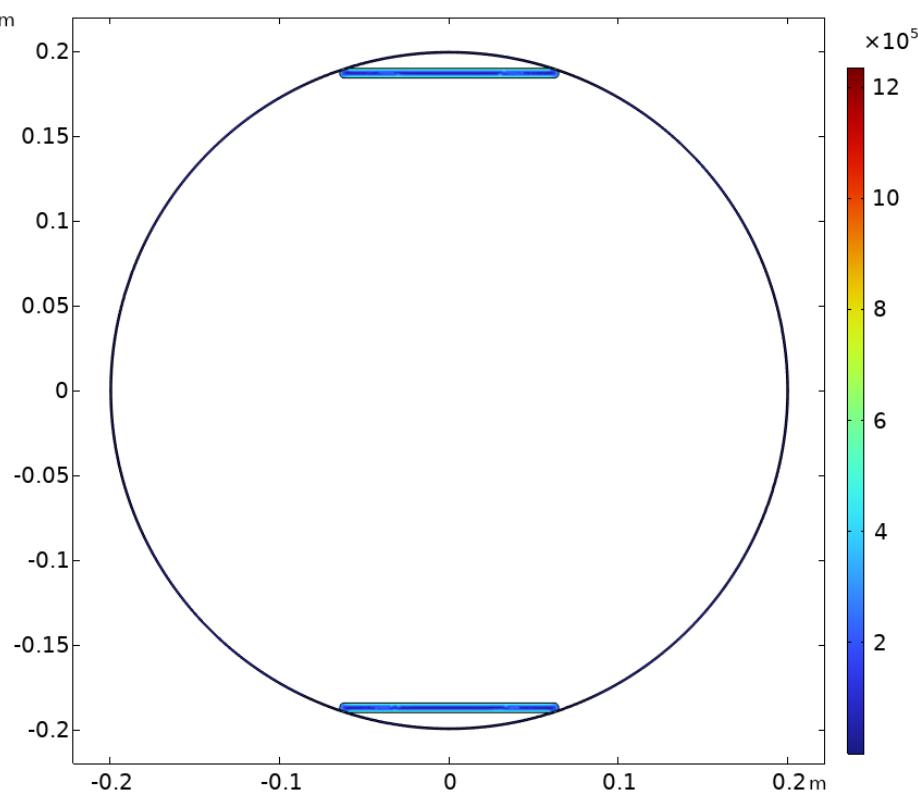


Figure 16: Von Mises stress distribution of the optimized beam design obtained by our framework, with reward 17.298.

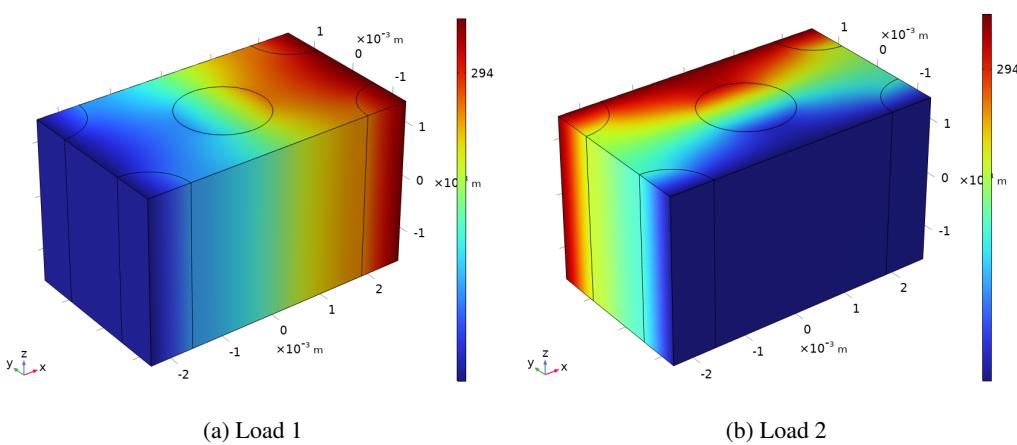


Figure 17: Optimized temperature fields of the meta-material designed by our framework under two different load conditions, achieving reward 1.278.

3510

3511

3512

3513

3514

3515

3516

3517

3518

3519

3520

3521

3522

3523

3524

3525

3526

3527

3528

3529

3530

3531

3532

3533

3534

3535

3536

3537

3538

3539

3540

3541

3542

3543

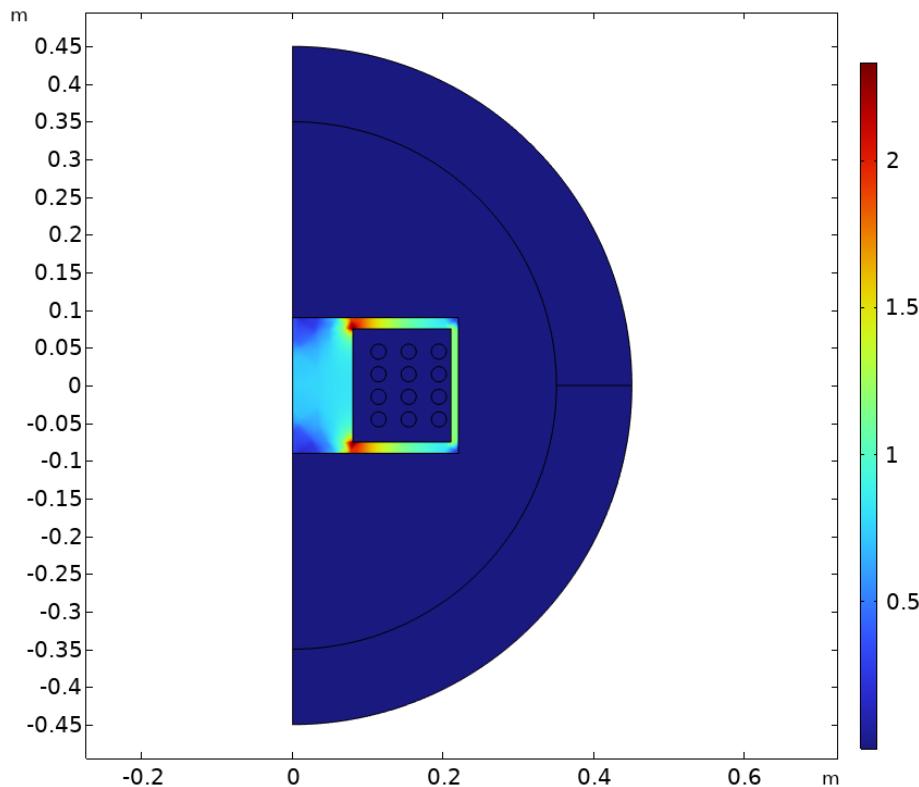
3544

3545

3546

3547

3548



3549 Figure 18: Magnetic flux density norm field of the best inductor configuration identified by our  
3550 framework, achieving reward 9.609.

3551

3552

3553

3554

3555

3556

3557

3558

3559

3560

3561

3562

3563

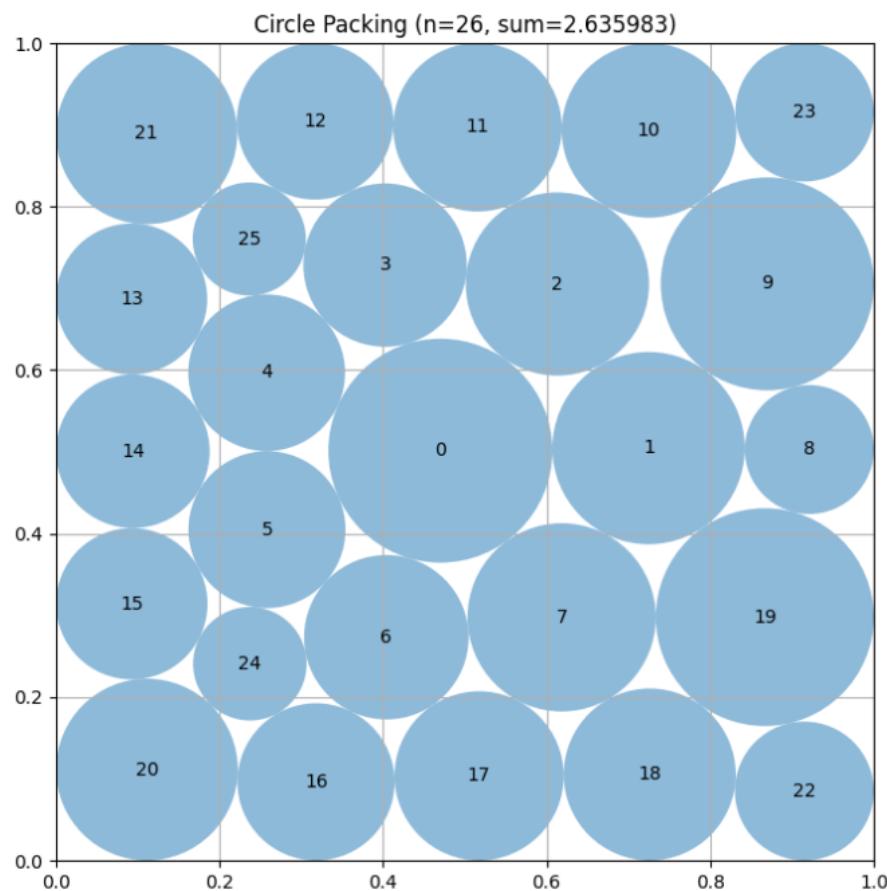
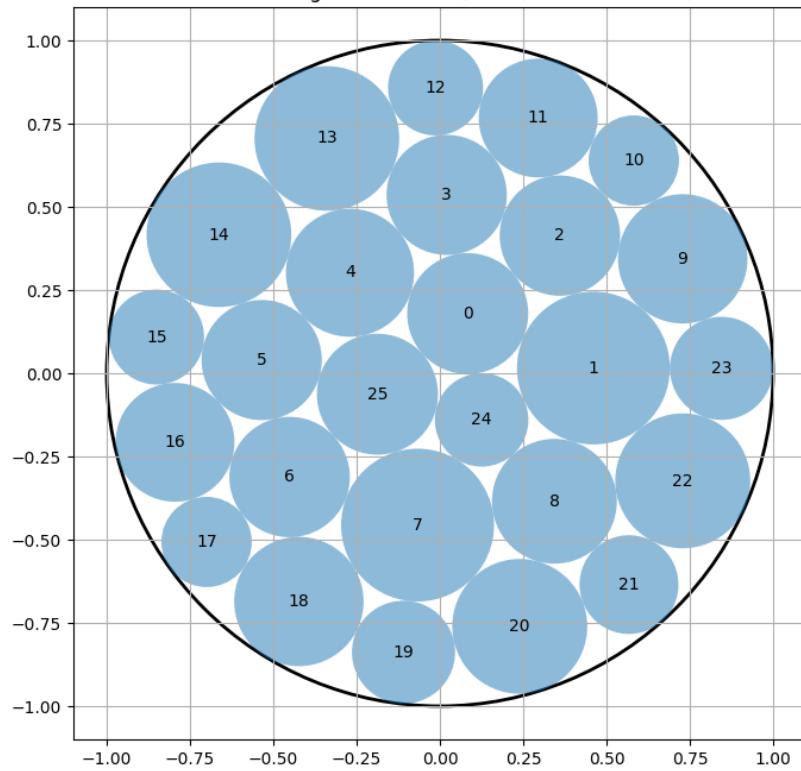


Figure 19: Arrangement of 26 circles within a square obtained by our framework, achieving a record-breaking sum of radii of 2.635983.

3618  
3619  
3620  
3621  
3622  
3623  
3624  
3625  
3626  
3627  
3628  
3629  
3630  
3631  
3632 Circle Packing in Unit Circle (n=26, sum=4.664465)  
3633  
3634  
3635  
3636  
3637  
3638  
3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646  
3647  
3648  
3649  
3650  
3651  
3652  
3653  
3654  
3655  
3656  
3657  
3658  
3659  
3660 Figure 20: Optimized circle packing of 26 disks within a unit disk by our model, yielding a sum of  
3661 radii of 4.664465.  
3662  
3663  
3664  
3665  
3666  
3667  
3668  
3669  
3670  
3671



```

3672
3673
3674
3675
3676 1 def engineer_features(X):
3677 2     features = X.copy()
3678 3     num_cols = [col for col in X.columns if X[col].dtype != 'object' and
3679 4     col not in ['fnlwgt', 'education-num']]
3680
3681 5     # Core interaction features
3682 6     features['age_hpw_product'] = features['age'] * features['hours-per-
3683 7     week']
3684 8     features['capital_total'] = features['capital-gain'] + features['
3685 9     capital-loss']
3686 10    features['log_fnlwgt'] = np.log(features['fnlwgt'] + 1)
3687 11    # Enhanced polynomial features
3688 12    for col in ['age', 'hours-per-week', 'capital-gain', 'capital-loss']:
3689 13        features[f'{col}_sq'] = features[col] ** 2
3690 14        features[f'{col}_cb'] = features[col] ** 3
3691 15    # Age-based features with log transformation
3692 16    features['log_age'] = np.log(features['age'] + 1)
3693 17    # Capital features with log transformations
3694 18    features['capital_gain'] = np.log(features['capital-gain'] + 1)
3695 19    features['capital_loss'] = np.log(features['capital-loss'] + 1)
3696 20    # Binned features for age and hours per week
3697 21    features['age_group'] = pd.cut(features['age'], bins=5, labels=False)
3698 22    features['hour_group'] = pd.cut(features['hours-per-week'], bins=5,
3699 23    labels=False)
3700 24    # Economic status features combining multiple variables
3701 25    features['economic_status'] = (features['age'] / features['hours-per-
3702 26    week']) * (features['capital_total'])
3703 27    # Additional indicators for capital gains and losses
3704 28    features['has_cagain'] = (features['capital-gain'] > 0).astype(int)
3705 29    features['has_caploss'] = (features['capital-loss'] > 0).astype(int)
3706 30    # Professional education indicator
3707 31    features['isProfessional'] = ((features['education'] == 'Prof-
3708 32    specialty') | (features['education'] == 'Exec-managerial') | (
3709 33    features['education'] == 'Assoc-acdm')).astype(int)
3710 34    # Managerial education indicator
3711 35    features['isManagerial'] = ((features['education'] == 'Exec-
3712 36    managerial') | (features['education'] == 'Assoc-voc')).astype(int)
3713 37    # Interaction between numerical features
3714 38    interaction_cols = ['age', 'hours-per-week', 'capital_gain', ' '
3715 39    'capital_loss']
3716 40    for i in range(len(interaction_cols)):
3717 41        for j in range(i+1, len(interaction_cols)):
3718 42            col1 = interaction_cols[i]
3719 43            col2 = interaction_cols[j]
3720 44            features[f'{col1}_x_{col2}'] = features[col1] * features[col2]
3721 45    # Ratio and difference features
3722 46    features['capital_gain_ratio'] = features['capital_gain'] / features[
3723 47    'capital_loss'].replace(0, 1)
3724 48    features['capital_diff'] = features['capital_gain'] - features['
3725 49    capital_loss']

50    return features

```

Figure 21: Python code of feature engineering for solving classification task on Adult dataset. Together with a LGBMClassifier and parameter search, our model achieved 82.07 marco f1 score.

```

3726
3727
3728
3729
3730 1 # Engineer more comprehensive interaction and polynomial features
3731 2 def create_engineered_features(df):
3732 3     # Interaction features
3733 4     df['RM_LSTAT'] = df['RM'] * df['LSTAT']
3734 5     df['NOX_DIS'] = df['NOX'] * df['DIS']
3735 6     df['CRIM_DIS'] = df['CRIM'] * df['DIS']
3736 7     df['INDUS_NOX'] = df['INDUS'] * df['NOX']
3737 8     df['CHAS_RM'] = df['CHAS'] * df['RM']
3738 9     df['AGE_DIS'] = df['AGE'] * df['DIS']
3739 10    df['RAD_NOX'] = df['RAD'] * df['NOX']
3740 11    df['PTRATIO_RM'] = df['PTRATIO'] * df['RM']
3741 12    df['INDUS_CHAS'] = df['INDUS'] * df['CHAS']
3742 13    df['RAD_DIS'] = df['RAD'] * df['DIS']
3743 14    df['RAD_CHAS'] = df['RAD'] * df['CHAS'] # New interaction
3744 15    df['CRIM_CHAS'] = df['CRIM'] * df['CHAS'] # Enhanced interaction
3745 16    # Polynomial features
3746 17    df['NOX_SQ'] = df['NOX'] ** 2
3747 18    df['RM_SQ'] = df['RM'] ** 2
3748 19    df['LSTAT_SQ'] = df['LSTAT'] ** 2
3749 20    df['NOX_CUBED'] = df['NOX'] ** 3
3750 21    df['RM_CUBED'] = df['RM'] ** 3
3751 22    df['LSTAT_CUBED'] = df['LSTAT'] ** 3
3752 23    df['NOX_FOUR'] = df['NOX'] ** 4 # Higher degree polynomial
3753 24    return df
3754 25
3755 26 engineered_features_transformer = Pipeline([
3756 27     ('engineer', FunctionTransformer(create_engineered_features))
3757 28 ])
3758 29
3759 30 # Preprocess numeric features
3760 31 numeric_transformer = Pipeline([
3761 32     ('imputer', KNNImputer(n_neighbors=3, weights='uniform')),
3762 33     ('scaler', RobustScaler())
3763 34 ])
3764 35
3765 36 # Preprocess categorical features
3766 37 categorical_transformer = Pipeline([
3767 38     ('imputer', SimpleImputer(strategy='mode')),
3768 39     ('target_encode', FunctionTransformer(lambda df: df.astype(object).
3769 40         where(df.notna(), df.mode().iloc[0])))
3770 41 ])
3771 42
3772 43 # Combine transformations
3773 44 preprocessor = ColumnTransformer(
3774 45     transformers=[
3775 46         ('eng', engineered_features_transformer, numeric_features),
3776 47         ('num', numeric_transformer, numeric_features),
3777 48         ('cat', categorical_transformer, categorical_features)
3778 49     ],
3779 50     remainder='drop'
3780 51 )

```

Figure 22: Key pre-processing steps the model implemented on Boston Housing dataset. Together with a XGBRegressor, the model achieved reward of 1.758, which means 1.747 in RMSE.