Sci-Mind: Cognitively-Inspired Adversarial Debate for Autonomous Mathematical Modeling

Ruiying Sun[1†], Wenjing Wang[2†], Qinhan Chen[3], Yanhui Song[3], Huangwei Chen[4], Haotong Luan[5] (✉), and Junhao Jia[3]

[1]Hangzhou Normal University [2]Xiamen University Malaysia
[3]Hangzhou Dianzi University [4]Zhejiang University
[5]Future Technology School Shenzhen Technology University
lht20060306@outlook.com, junhaojia530@gmail.com

Abstract. Real-world mathematical modeling is inherently an experiential and collaborative endeavor. Domain experts rarely solve complex problems from scratch; instead, they draw upon analogies from historical cases and subject their hypotheses to rigorous peer scrutiny. However, autonomous agents powered by Large Language Models predominantly rely on isolated reasoning paradigms, frequently generating plausible but fundamentally flawed models due to a lack of domain grounding and adversarial verification. To address these limitations, we propose Sci-Mind, a novel framework that mirrors the human scientific discovery process. Sci-Mind integrates Experiential Memory Recall to retrieve executable code snippets and modeling paradigm descriptors, grounding abstract reasoning in historical solutions. Subsequently, it employs an Adversarial Cognitive Dialectic where a Theorist optimizing mathematical coherence and a Pragmatist enforcing data feasibility debate through competing objectives to prune elegant but infeasible formulations. A Self-Validating Execution Strategy further ensures blueprint consistency through formal predicates before code generation, achieving fully autonomous execution. Extensive experiments on the MM-Bench and EngiBench benchmarks demonstrate that Sci-Mind significantly outperforms leading autonomous agents in both modeling rigorousness and code executability.

Keywords: Autonomous Agents · Scientific Reasoning · Multi-Agent Debate · Knowledge Retrieval.

1 Introduction

Mathematical modeling serves as the fundamental bridge translating complex physical phenomena into formal analytical systems [13]. Despite its wide applications in operations research [4], physical simulations [21], and biomedical network analysis [18,19], autonomous tracking of open-ended scientific tasks relying solely on isolated reasoning often fails under complex real-world constraints [36]. In contrast, human scientific methodology has the advantage of experiential and

† Equal contribution; ✉ corresponding author.

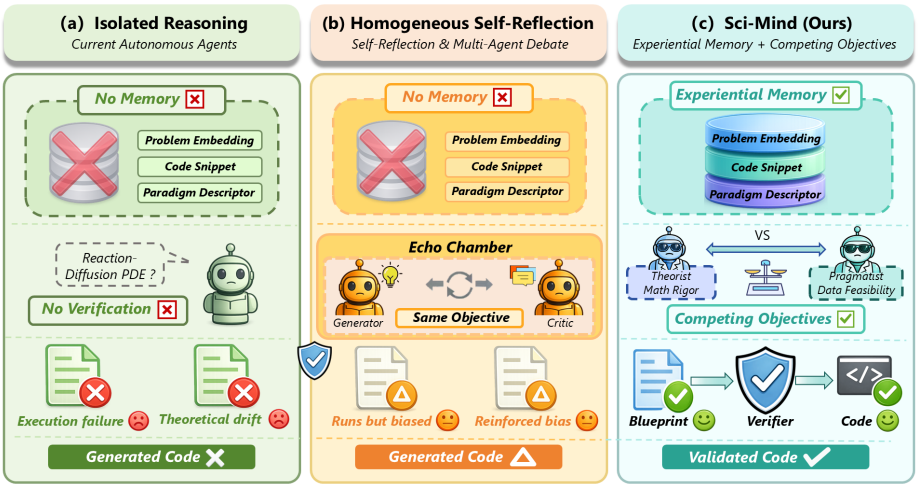

Fig. 1: Comparison of three paradigms for LLM-based mathematical modeling.

collaborative frameworks, providing rigorous verification and historical grounding [35]. To integrate these advantages, augmenting Large Language Models (LLMs) with scientific reasoning paradigms is crucial for robust performance.

However, current autonomous agents model the problem using zero-shot inference from the initial prompt, which often leads to severe theoretical drift or execution failures [20]. This limitation stems from two main factors. First, a single isolated reasoning process offers limited structural information about the target problem, failing to capture complex modeling paradigms across different scientific domains [31]. Second, the inherent ambiguity of real-world data often causes agents to overemphasize elegant but unvalidated theoretical features, resulting in practical infeasibility [45]. Moreover, directly translating abstract math into code without structural priors is highly brittle in practical scenarios [25], consequently hindering its widespread deployment. As illustrated in Fig. 1(a), these agents operate without experiential memory or adversarial verification, causing both execution failures and theoretical drift in the generated code.

On the other hand, recent advances in LLM reasoning have focused on integrating self-reflection mechanisms [34]. In previous works, actor and critic models are first prompted independently, then fused through iterative feedback loops operating at the semantic level [37]. However, a significant portion of this self-reflection is redundant, forming an echo chamber that amplifies generative biases without exposing logical blind spots [38]. As shown in Fig. 1(b), when the generator and critic share the same objective, the resulting code may execute but carries reinforced biases that remain undetected. Furthermore, heterogeneous domain gaps between abstract theoretical logic and concrete programming syntax hinder the establishment of effective code execution [46]. This raises a basic

question: *How can we achieve deep theoretical rigor while escaping echo-chamber redundancies and bridging the gap to executable code?*

To address these challenges, we introduce cognitive bionics as a high-level representation to enhance mathematical modeling in LLMs. Human-like scientific discovery provides a more grounded understanding of problems than isolated reasoning, addressing structural limitations in abstract generation [26]. Experiential memory representation can offer clear semantic priors, including logic paradigms and executable demonstrations, enabling effective theory-implementation separation [24]. Recent advances in Retrieval-Augmented Generation (RAG) have demonstrated the potential of external knowledge for robust reasoning [27]. However, these methods face the challenge of semantic misalignment, as they predominantly fetch unstructured text that fails to map onto formal executable structures [3]. To bridge this gap, we extend current retrieval methods by introducing executable code-level insights and a non-cooperative game mechanism.

Building upon the cognitive bionic paradigm, we propose a novel mathematical modeling framework. As depicted in Fig. 1(c), our framework consists of three key components: Experiential Memory Recall (EMR), Adversarial Cognitive Dialectic (ACD), and a Self-Validating Execution Strategy (SVE). The EMR retrieves executable code snippets and modeling paradigm descriptors from a structured knowledge base, grounding abstract reasoning in historical solutions. The ACD employs functionally asymmetric personas, a Theorist and a Pragmatist, who optimize explicitly competing objectives to dynamically verify both theoretical validity and data feasibility. The SVE implements an automated blueprint-verifier-code pipeline with structured JSON blueprints and formal consistency predicates to bridge validated hypotheses and code execution.

With the above components, our Sci-Mind enables comprehensive scientific reasoning, leveraging both historical evidence and adversarial context to maintain theoretical rigor under complex scenarios. Extensive experiments on the MM-Bench [30] and EngiBench [48] datasets demonstrate the effectiveness of our method. In summary, our main contributions are as follows:

(1) We propose Sci-Mind, a cognitively-inspired framework that retrieves executable structural priors from a multi-tier knowledge base via Experiential Memory Recall, effectively bridging the abstraction-implementation gap in autonomous mathematical modeling.

(2) We introduce the Adversarial Cognitive Dialectic, in which a Theorist and a Pragmatist debate through explicitly competing objectives, and demonstrate that this objective asymmetry is the critical factor for escaping theoretical local optima and echo-chamber redundancies.

(3) Extensive experiments across two benchmarks across multiple LLM backbones confirm that Sci-Mind consistently outperforms existing autonomous agents in both modeling rigorousness and code executability, with gains that are orthogonal to backbone capabilities.

2 Related Work

2.1 Autonomous Agents for Scientific Reasoning

Autonomous agents powered by LLMs have emerged as a promising paradigm for decomposing and solving complex analytical tasks [40,44]. Early frameworks leverage Zero-shot CoT prompting [41,23] to elicit step-by-step reasoning, while DS-Agent [12] and MM-Agent [30] advance workflow standardization for data science and mathematical modeling respectively. In parallel, Coscientist [5] and ChemCrow [32] demonstrate the potential of LLM agents in autonomous experimental design. Recently, RAG [27] has been integrated into scientific agents to mitigate hallucinations [10], but existing approaches predominantly retrieve unstructured textual definitions that fail to bridge the gap between abstract formulations and executable code. To address this limitation, Sci-Mind introduces Experiential Memory Recall to retrieve executable code snippets alongside modeling paradigm descriptors, providing both procedural and conceptual priors for grounded reasoning.

2.2 Self-Reflection and Multi-Agent Debate

Self-reflection mechanisms enable LLMs to iteratively critique and refine their outputs. Reflexion [37] maintains a verbal memory of past failures to guide future attempts, while Self-Refine [33] prompts a single model to generate, critique, and revise in a closed loop. Multi-agent frameworks extend this idea by simulating collaborative environments: CAMEL [28] introduces inception prompting for role-playing communication, MetaGPT [15] assigns specialized software engineering roles to distinct agents, FetalAgents [16] demonstrates multi-agent collaboration for medical image analysis, and AutoGen [42] adapts the Actor-Critic paradigm for LLM-based hypothesis generation and evaluation. Multi-agent debate methods [9,29,6] further enhance robustness through interactive multi-round discussions. However, when all agents share the same implicit objective, the interaction frequently degenerates into an echo chamber that reinforces biases without exposing fundamental flaws [17]. This is particularly problematic in open-ended scientific modeling, where theoretical validity and data feasibility constitute structurally different evaluation dimensions that a single shared objective cannot simultaneously capture. In contrast, Sci-Mind introduces the Adversarial Cognitive Dialectic, in which a Theorist optimizing mathematical coherence and a Pragmatist enforcing data feasibility engage in structured debate with explicitly competing objectives.

2.3 Code Generation and Autonomous Execution

Code generation agents translate natural language into executable scripts, with HumanEval [7] establishing the evaluation paradigm, LDB [47] advancing step-by-step runtime verification, Voyager [39] introducing self-improving code libraries, and Data Interpreter [14] decomposing data science problems via hierarchical graph modeling. Recent agentic systems [30,48] further adopt sandboxed

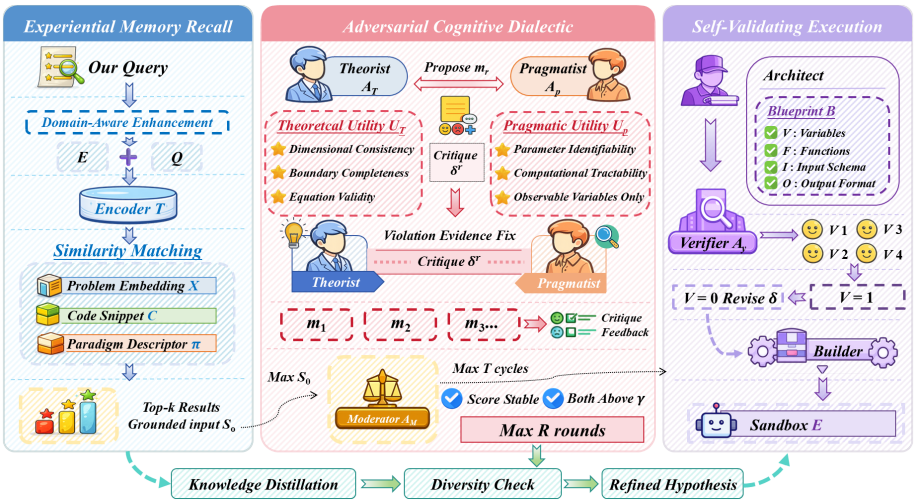

Fig. 2: The Overall Architecture of the Sci-Mind Framework.

environments with automated error tracing. Despite these advances, direct logic-to-code translation remains brittle because modeling decisions and implementation decisions are entangled in a single generation pass. Our approach decouples these concerns through a Self-Validating Execution Strategy that interposes a structured JSON blueprint and an automated Verifier agent checking formal consistency predicates before code generation proceeds.

3 Methodology

In this paper, we propose Sci-Mind, a cognitively-inspired framework for autonomous mathematical modeling. As shown in Fig 2, Sci-Mind consists of three tightly integrated components: Experiential Memory Recall (EMR) for grounding abstract reasoning in executable structural priors, Adversarial Cognitive Dialectic (ACD) for jointly optimizing theoretical rigor and data feasibility through structured adversarial interaction, and Self-Validating Execution Strategy (SVE) for autonomous blueprint verification and robust code synthesis. We further introduce an Epistemic Self-Evolution mechanism to enable continual knowledge accumulation. Details are described as follows.

3.1 Overall Framework

At reasoning step t, the system receives a user query $\mathbf{Q} \in \mathbb{R}^{N \times d}$, where N is the token length and d is the hidden dimension. The overall pipeline can be formulated as a sequential composition of three mappings:

$$\mathbf{C}^* = \mathcal{S} \circ \mathcal{A} \circ \mathcal{R}(\mathbf{Q}, \mathcal{K}) \tag{1}$$

where $\mathcal{R}$ denotes the EMR retrieval operator, $\mathcal{A}$ represents the ACD refinement process, and $\mathcal{S}$ is the SVE code synthesis module. The knowledge base $\mathcal{K}$ is maintained as a dynamic repository of structured triplets. The output $\mathbf{C}^*$ is an executable code artifact that is validated within a sandboxed environment $\mathcal{E}$.

3.2 Experiential Memory Recall (EMR)

A critical bottleneck in autonomous modeling is the abstraction-implementation gap: an agent may possess sufficient declarative knowledge to formulate a model but lack the procedural knowledge to implement it as executable code. Existing retrieval-augmented methods [22] predominantly fetch unstructured textual definitions, which provide semantic context but fail to supply the syntactic scaffolding necessary for code generation. To bridge this gap, EMR retrieves structured historical solutions as executable priors.

Multi-Tier Knowledge Representation. We organize the knowledge base $\mathcal{K}$ as a set of structured triplets $\{k_i\}_{i=1}^{|\mathcal{K}|}$, where each entry $k_i = (\mathbf{x}_i, \mathbf{c}_i, \pi_i)$ consists of three components: the problem semantic embedding $\mathbf{x}_i \in \mathbb{R}^d$, a verified executable code snippet $\mathbf{c}_i$, and a modeling paradigm descriptor π_i that encodes the abstract solution pattern. This tri-level representation ensures that retrieved knowledge simultaneously provides semantic context, implementation templates, and high-level structural guidance.

Domain-Aware Retrieval. To mitigate semantic misalignment between natural language problem descriptions and formal mathematical structures, we construct a domain-aware enhanced query. Specifically, we prepend a retrieval instruction prefix $\mathbf{E}$ to the input query, forming the augmented representation:

$$\mathbf{H} = [\mathbf{E}\,;\,\mathbf{Q}] \tag{2}$$

where $[\,;\,]$ denotes sequence concatenation and $\mathbf{E}$ is instantiated as "*Retrieve the formal mathematical structure for* [`domain`]", with [`domain`] dynamically populated from the query context.

The augmented query is encoded by a dense encoder $\mathcal{T}$ to produce a query vector $\hat{\mathbf{h}} \in \mathbb{R}^d$:

$$\hat{\mathbf{h}} = \text{MeanPool}(\mathcal{T}(\mathbf{H})) \tag{3}$$

We then compute the cross-modal relevance between $\hat{\mathbf{h}}$ and each knowledge entry. The relevance score for entry k_j is defined as:

$$s(\mathbf{Q}, k_j) = \frac{\hat{\mathbf{h}}^\top \phi(\mathbf{x}_j)}{\|\hat{\mathbf{h}}\|_2 \cdot \|\phi(\mathbf{x}_j)\|_2} \tag{4}$$

where $\phi(\cdot)$ denotes the knowledge encoder, which shares parameters with $\mathcal{T}$ but operates on the stored problem embeddings.

The top-k entries with the highest relevance scores are selected to form the structural prior set:

$$\mathcal{C}_{top\text{-}k} = \underset{\mathcal{C}\subset\mathcal{K},\,|\mathcal{C}|=k}{\arg\max} \sum_{k_j\in\mathcal{C}} s(\mathbf{Q}, k_j) \tag{5}$$

The retrieved code snippets and paradigm descriptors are concatenated with the original query to form the structurally-grounded input $\mathbf{S}^0$:

$$\mathbf{S}^0 = [\mathbf{Q}\,;\, \bigoplus_{k_j\in\mathcal{C}_{top\text{-}k}} (\mathbf{c}_j \oplus \pi_j)] \tag{6}$$

where $\oplus$ denotes token-level concatenation. This $\mathbf{S}^0$ serves as the grounded initialization for the subsequent adversarial refinement stage.

3.3 Adversarial Cognitive Dialectic (ACD)

Standard self-reflection methods prompt a single model to iteratively critique its own output. Recent studies [17] have shown that such homogeneous reflection frequently degenerates into an echo chamber that reinforces generative biases without exposing fundamental flaws. We argue that the root cause is objective homogeneity: both the generator and the critic optimize toward the same implicit objective. To address this, ACD decomposes the verification task into two explicitly competing objectives evaluated by functionally asymmetric agents, drawing on the principle that structured disagreement produces more robust outcomes than unchallenged consensus.

Dual-Objective Formulation. We define two evaluation functions that operate on different modalities of the modeling hypothesis. Given a candidate model hypothesis $\mathbf{m}$ and the observable dataset constraints $\mathcal{D}_{obs}$, we define:

Theoretical Utility $U_T(\mathbf{m})$ evaluates the mathematical coherence of the hypothesis, quantified as a weighted combination of interpretable criteria scored by the Theorist agent $\mathcal{A}_T$:

$$U_T(\mathbf{m}) = \sum_{l=1}^{L} w_l^{(T)} \cdot \sigma_l^{(T)}(\mathbf{m}) \tag{7}$$

where $\sigma_l^{(T)}(\mathbf{m}) \in [0,1]$ denotes the l-th evaluation criterion, $w_l^{(T)}$ is the corresponding importance weight with $\sum_l w_l^{(T)} = 1$, and L is the number of criteria. Each $\sigma_l^{(T)}$ is implemented as a structured rubric prompt that returns a normalized score.

Pragmatic Utility $U_P(\mathbf{m} \mid \mathcal{D}_{obs})$ evaluates the feasibility of the hypothesis against concrete data constraints, scored by the Pragmatist agent $\mathcal{A}_P$:

$$U_P(\mathbf{m} \mid \mathcal{D}_{obs}) = \sum_{l=1}^{L'} w_l^{(P)} \cdot \sigma_l^{(P)}(\mathbf{m}, \mathcal{D}_{obs}) \tag{8}$$

where $\sigma_l^{(P)}(\mathbf{m}, \mathcal{D}_{obs}) \in [0,1]$ denotes pragmatic criteria, $w_l^{(P)}$ is the corresponding weight with $\sum_l w_l^{(P)} = 1$, and L' is the number of pragmatic criteria.

The key design principle is that U_T and U_P are structurally non-aligned: maximizing U_T alone tends to produce mathematically elegant but over-parameterized models, while maximizing U_P alone leads to oversimplified models that lack generalization. The productive tension between these objectives drives the system toward balanced solutions.

Structured Adversarial Interaction. At debate round r, the interaction proceeds as a two-player sequential game:

Step 1: Proposal. The Theorist proposes or refines a model hypothesis conditioned on the structural prior and the accumulated critique history:

$$\mathbf{m}^r = \mathcal{A}_T\left(\left[\mathbf{S}^0\,;\,\boldsymbol{\delta}^{r-1}\right]\right) \tag{9}$$

where $\boldsymbol{\delta}^0 = \varnothing$ for the initial round.

Step 2: Critique. The Pragmatist evaluates $\mathbf{m}^r$ against $\mathcal{D}_{obs}$ and produces a structured critique identifying specific feasibility violations:

$$\boldsymbol{\delta}^r = \mathcal{A}_P\left([\mathbf{m}^r\,;\,\mathcal{D}_{obs}]\right) \tag{10}$$

Each critique $\boldsymbol{\delta}^r$ is constrained to a structured format that specifies: (i) the violated pragmatic criterion, (ii) the conflicting evidence from $\mathcal{D}_{obs}$, and (iii) a suggested remediation direction. This structured format prevents vague or redundant feedback.

Step 3: Moderation. A Moderator agent $\mathcal{A}_M$ evaluates both utilities at each round and computes a joint consensus score:

$$\mathit{\Gamma}^r = \lambda \cdot U_T(\mathbf{m}^r) + (1-\lambda)\cdot U_P(\mathbf{m}^r \mid \mathcal{D}_{obs}) \tag{11}$$

where $\lambda \in (0,1)$ is a balancing coefficient that controls the trade-off between theoretical elegance and practical feasibility. The debate terminates when the following convergence condition is satisfied:

$$|\mathit{\Gamma}^r - \mathit{\Gamma}^{r-1}| < \epsilon \quad \text{and} \quad \min\{U_T(\mathbf{m}^r),\, U_P(\mathbf{m}^r \mid \mathcal{D}_{obs})\} > \gamma \tag{12}$$

where ϵ is the convergence tolerance and γ is the minimum acceptable quality threshold. This dual condition ensures that convergence requires not only stability but also that neither objective is sacrificed below an acceptable level.

The first condition enforces Pareto stationarity: the solution has reached a point where further debate rounds yield diminishing marginal improvements to the joint objective. The second condition acts as a safety constraint, preventing degenerate equilibria where one utility collapses. Together, they define a bounded rational equilibrium, a stable state where neither agent can unilaterally improve their objective without degrading the other's, subject to the quality floor γ.

To prevent indefinite oscillation, we impose a maximum round budget R_{max}. If the convergence condition is not met within R_{max} rounds, the Moderator selects the hypothesis with the highest $\mathit{\Gamma}^r$ across all rounds:

$$\mathbf{m}^* = \underset{r\in\{1,\ldots,R_{max}\}}{\arg\max}\ \mathit{\Gamma}^r \tag{13}$$

3.4 Self-Validating Execution Strategy (SVE)

To bridge the gap between refined model hypotheses and reliable code execution, we introduce SVE, a fully autonomous pipeline that structures code generation.

Blueprint Construction. The Architect module translates the equilibrium hypothesis $\mathbf{m}^*$ into a structured JSON blueprint $\mathcal{B}$, which serves as an intermediate representation between abstract modeling logic and concrete code. This design draws on the principle that separating specification from implementation reduces cascading errors in automated code synthesis [8]. Formally, $\mathcal{B}$ is defined as a tuple:

$$\mathcal{B} = \langle \mathcal{V}, \mathcal{F}, \mathcal{I}, \mathcal{O} \rangle \tag{14}$$

where $\mathcal{V} = \{(v_i, \tau_i, d_i)\}$ specifies variable declarations with types τ_i and tensor dimensions d_i; $\mathcal{F} = \{(f_j, \text{sig}_j, \text{dep}_j)\}$ defines function signatures with dependency relations; $\mathcal{I}$ specifies the data ingestion schema; and $\mathcal{O}$ defines the expected output format. This explicit intermediate representation decouples modeling decisions from syntactic implementation, reducing cascading errors during code generation.

Automated Consistency Verification. Rather than relying on costly human approval, we introduce a Verifier agent $\mathcal{A}_V$ that performs automated multi-level consistency checks between the blueprint $\mathcal{B}$ and the equilibrium hypothesis $\mathbf{m}^*$. The Verifier evaluates a set of formal consistency predicates and produces a composite verification score:

$$V(\mathcal{B}, \mathbf{m}^*) = \prod_{p=1}^{P} v_p(\mathcal{B}, \mathbf{m}^*) \tag{15}$$

where each $v_p \in \{0, 1\}$ is a binary consistency predicate.

The verification outcome determines the downstream action:

$$\mathbf{C}^* = \begin{cases} \text{Builder}(\mathcal{B}), & \text{if } V(\mathcal{B}, \mathbf{m}^*) = 1 \\ \text{Builder}(\text{Revise}(\mathcal{B},\, \boldsymbol{\xi}))\,, & \text{if } V(\mathcal{B}, \mathbf{m}^*) = 0 \end{cases} \tag{16}$$

where $\boldsymbol{\xi} = \{(p, \text{desc}_p) \mid v_p = 0\}$ is the set of violated predicates with diagnostic descriptions generated by $\mathcal{A}_V$. The Revise operator updates $\mathcal{B}$ by addressing each violation while preserving the components that passed verification. This process iterates for at most T_{max} cycles:

$$\mathcal{B}^{t+1} = \text{Revise}(\mathcal{B}^t, \boldsymbol{\xi}^t), \quad t = 0, 1, \ldots, T_{max} - 1 \tag{17}$$

terminating early when $V(\mathcal{B}^t, \mathbf{m}^*) = 1$. Compared to human gating, this automated mechanism achieves higher coverage on formal consistency dimensions that human reviewers frequently overlook, while eliminating latency and maintaining full autonomy.

Self-Correcting Execution. The synthesized code $\mathbf{C}^*$ executes within a sandboxed environment $\mathcal{E}$. In the event of a runtime failure, the error trace $\mathbf{e}^j$ is extracted and combined with the original code context to produce a corrected version:

$$\mathbf{C}^{j+1} = \mathcal{F}_{refine}([\mathbf{C}^j \,;\, \mathbf{e}^j \,;\, \mathcal{B}]) \tag{18}$$

where $\mathcal{F}_{refine}$ denotes the LLM-based refinement operator and j indexes the correction iteration. Crucially, we include the blueprint $\mathcal{B}$ in the refinement context to prevent the correction process from drifting away from the validated modeling intent. The refinement process terminates upon successful execution or after exhausting a maximum retry budget J_{max}.

We define the execution success indicator as:

$$\mathbb{1}_{exec} = \begin{cases} 1, & \text{if } \mathcal{E}(\mathbf{C}^j) \text{ returns valid output} \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

The overall Code Executability (CE) metric reported in our experiments is then computed as $\text{CE} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbb{1}_{exec}^{(p)}$, where $\mathcal{P}$ is the problem set.

3.5 Epistemic Self-Evolution

To enable lifelong learning beyond a static knowledge base, we design a continuous self-evolution mechanism [39] that consolidates successful reasoning experiences into reusable knowledge.

Knowledge Distillation. Upon successfully solving a novel problem $\mathbf{q}_{new}$ with validated solution $\mathbf{s}_{new}^+$, a memory consolidation function $\Psi(\cdot)$ abstracts the solution into a generalized knowledge entry:

$$k_{new} = \Psi(\mathbf{s}_{new}^+) = (\mathbf{x}_{new},\, \mathbf{c}_{new},\, \pi_{new}) \tag{20}$$

where $\mathbf{x}_{new}$ is the embedding of the problem, $\mathbf{c}_{new}$ is the validated code, and π_{new} is the abstracted paradigm descriptor.

Diversity-Aware Admission. To prevent knowledge base bloating with redundant entries, we impose a diversity-aware admission criterion based on the maximum similarity with existing entries:

$$\Delta(\mathbf{x}_{new}, \mathcal{K}) = \max_{\mathbf{x}_i \in \mathcal{K}} \frac{\mathbf{x}_{new}^\top \mathbf{x}_i}{\|\mathbf{x}_{new}\|_2 \cdot \|\mathbf{x}_i\|_2} \tag{21}$$

The new entry is admitted if and only if it provides sufficient novelty:

$$\mathcal{K}_{t+1} = \begin{cases} \mathcal{K}_t \cup \{k_{new}\}, & \text{if } \Delta(\mathbf{x}_{new}, \mathcal{K}_t) < \tau \\ \mathcal{K}_t, & \text{otherwise} \end{cases} \tag{22}$$

where $\tau \in (0, 1)$ is the novelty threshold. This mechanism ensures that the knowledge base grows in informational diversity rather than volume, maintaining retrieval efficiency while expanding the coverage of solvable problem types.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on two complementary benchmarks. MM-Bench [30] contains 111 real-world mathematical modeling problems from MCM/ICM competitions spanning 2000 to 2025, covering 10 domains and 8 task types with end-to-end problem solving requirements. EngiBench Level 3 [48] provides 43 open-ended engineering modeling tasks from MCM/ICM, CUMCM, and APMCM spanning 2010 to 2024, covering Systems & Control, Physical & Structural, and Chemical & Biological subfields with official rubrics.

Baselines. We compare against seven baselines spanning direct prompting, specialized agents, and multi-agent frameworks. Direct Prompting applies Zero-shot CoT [41] to the backbone LLM. ReAct [44] interleaves reasoning with acting. DS-Agent [12] leverages case-based reasoning from Kaggle. CAMEL [28] enables role-playing multi-agent communication. AutoGen [42] supports flexible multi-agent conversation with code execution. MM-Agent [30] is the current state-of-the-art framework for mathematical modeling. Human Expert solutions are those awarded Honorable Mention or higher in actual MCM/ICM competitions.

Metrics. Following the MM-Bench evaluation protocol [30], we report four qualitative dimensions scored on a 1 to 10 scale: Analysis Evaluation (AE), Modeling Rigorousness (MR), Practicality (PS), and Result Analysis (RBA). The Avg. score is their arithmetic mean. We additionally report Code Executability (CE), defined as the percentage of generated solutions that execute without runtime errors. Each problem is evaluated 3 times and we report the mean with standard deviation. For EngiBench, we follow its official protocol [48].

4.2 Implementation Details

Backbones. To disentangle method-level contributions from backbone capabilities, we evaluate Sci-Mind and key baselines across four representative LLMs: GPT-4o [1], Claude-3.5-Sonnet [2], DeepSeek-R1 [11], and Qwen-2.5-72B [43]. These four backbones span proprietary and open-source models, general-purpose and reasoning-specialized architectures, and a range of parameter scales. All experiments use default temperature settings provided by the respective API providers. Multi-agent baselines are given equivalent computational budgets in terms of total LLM calls per problem to ensure fair comparison.

Hyperparameters. The EMR knowledge base $\mathcal{K}$ is constructed from 847 historical entries sourced from Kaggle competition solutions and open-access scientific code repositories, each processed into the tri-level format of problem embedding, verified code snippet, and paradigm descriptor. We use the top-$k = 3$ retrieved entries per query. The dense encoder $\mathcal{T}$ in EMR uses a pre-trained Sentence-BERT model. For ACD, the balancing coefficient is set to $\lambda = 0.5$, the quality threshold to $\gamma = 0.75$, the convergence tolerance to $\epsilon = 0.05$, and the maximum debate rounds to $R_{max} = 6$. For SVE, the blueprint revision budget is $T_{max} = 3$ and the code retry budget is $J_{max} = 5$. For self-evolution, the novelty threshold is $\tau = 0.85$.

Table 1: Main results on MM-Bench across four backbones. All scores are reported as mean ± std over three evaluation runs.

Method	Backbone	AE	MR	PS	RBA	CE (%)	Avg.
Direct Prompting	GPT-4o	7.62±.18	3.86±.24	5.17±.21	6.28±.15	42.5	5.73
	Claude-3.5	7.78±.16	4.15±.22	5.42±.19	6.50±.17	45.8	5.96
	DeepSeek-R1	8.05±.14	5.32±.28	5.85±.20	6.92±.16	48.2	6.54
	Qwen-2.5-72B	7.45±.20	3.58±.26	4.90±.23	6.05±.18	38.7	5.50
ReAct	GPT-4o	7.89±.15	5.24±.31	6.02±.19	6.71±.22	51.8	6.47
DS-Agent	GPT-4o	8.18±.14	7.08±.26	7.47±.18	7.86±.20	68.2	7.65
CAMEL	GPT-4o	8.35±.16	6.72±.29	7.18±.23	7.54±.18	63.7	7.45
AutoGen	GPT-4o	8.42±.13	6.95±.25	7.62±.17	7.93±.19	71.4	7.73
MM-Agent	GPT-4o	9.15±.11	7.28±.22	8.44±.14	8.85±.12	82.0	8.43
	Claude-3.5	9.22±.10	7.45±.20	8.52±.13	8.90±.11	84.5	8.52
	DeepSeek-R1	9.30±.09	7.82±.18	8.68±.12	9.02±.10	85.2	8.71
	Qwen-2.5-72B	8.95±.13	6.90±.24	8.15±.16	8.58±.14	76.8	8.15
Human Expert	–	9.04±.35	7.91±.42	7.42±.38	8.92±.30	–	8.32
Sci-Mind (Ours)	GPT-4o	9.45±.09	8.92±.14	9.15±.11	9.05±.10	96.4	9.14
	Claude-3.5	9.50±.08	9.05±.12	9.22±.10	9.12±.09	97.1	9.22
	DeepSeek-R1	**9.58**±.07	**9.28**±.10	**9.35**±.09	**9.25**±.08	**97.8**	**9.37**
	Qwen-2.5-72B	9.28±.11	8.52±.16	8.82±.13	8.78±.12	93.5	8.85

4.3 Main Results

Table 1 presents the main results on MM-Bench. Three key findings emerge.

Consistent SOTA across backbones. Sci-Mind outperforms all baselines on every backbone. With GPT-4o it surpasses MM-Agent by 0.71 in Avg. score. With DeepSeek-R1 the margin remains at 0.66. Sci-Mind with Qwen-2.5-72B achieves 8.85, still exceeding MM-Agent with DeepSeek-R1 at 8.71, demonstrating that the gains are architecture-driven rather than backbone-dependent.

MR and CE as primary improvement channels. Across all backbones, the largest gains appear in MR with an average improvement of 1.52 and CE with an average improvement of 13.8, confirming that ACD and EMR address the two core bottlenecks, theoretical drift and the abstraction-implementation gap, that backbone scaling alone cannot resolve. DeepSeek-R1 amplifies these advantages further, reaching MR of 9.28 and CE of 97.8.

Generic multi-agent frameworks are insufficient. CAMEL and AutoGen outperform single-agent baselines but fall short of the specialized MM-Agent, indicating that general-purpose multi-agent dialogue lacks the targeted verification needed for scientific modeling.

Regarding Human Experts, Sci-Mind's advantage is most pronounced in PS, scoring 9.15 compared to 7.42, likely because the system validates implementation details against data constraints more exhaustively. We note that all scores are produced by automated evaluators and may not fully capture the nuanced creativity of human solutions.

Table 2: Evaluation on EngiBench Level 3. SC: Systems & Control, PS: Physical & Structural, CB: Chemical & Biological. RS denotes rubric score (%), CE denotes code executability (%). RS is reported as mean ± std over three runs.

Method	**Backbone**	**SC**		**PS**		**CB**		**Avg.**
		RS	**CE**	**RS**	**CE**	**RS**	**CE**	
Direct Prompting	GPT-4o	28.4±1.8	35.0	24.1±2.1	30.0	31.7±1.6	40.0	28.1
	DeepSeek-R1	34.2±1.5	42.5	30.8±1.9	37.5	38.5±1.4	47.5	34.5
MM-Agent	GPT-4o	51.3±1.2	72.5	46.8±1.5	65.0	54.0±1.1	75.0	50.7
	DeepSeek-R1	56.8±1.0	77.5	52.4±1.3	72.5	59.2±0.9	80.0	56.1
Sci-Mind	GPT-4o	63.8±0.8	90.0	58.5±1.0	85.0	66.2±0.7	92.5	62.8
	DeepSeek-R1	**68.5**±0.6	**92.5**	**64.2**±0.8	**90.0**	**71.0**±0.5	**95.0**	**67.9**

4.4 Cross-Domain Generalization on EngiBench

Table 2 confirms that Sci-Mind's advantages generalize to engineering domains beyond competition-style problems. With GPT-4o, Sci-Mind achieves an average RS improvement of 12.1 over MM-Agent across three subfields. With DeepSeek-R1, the gap remains at 11.8. The largest gains appear in PS, where complex dynamical modeling and spatially heterogeneous data handling are required, precisely the scenarios where ACD most effectively identifies model-data mismatches. The smaller gap in CB reflects the heavier reliance on well-established reaction kinetics paradigms that baseline agents already handle reasonably well. CE improvements average 17.5 over MM-Agent across all configurations, indicating that EMR's structural priors transfer robustly across domain boundaries. Notably, Sci-Mind with GPT-4o already surpasses MM-Agent with DeepSeek-R1 on both RS and CE in every subfield, further confirming that the framework's design contributes more than backbone strength alone.

4.5 Ablation Study

Table 3 presents the ablation results across three tiers.

RQ1: Does EMR bridge the abstraction-implementation gap? Removing EMR causes CE to drop from 96.4% to 55.2% while MR decreases only by 0.42. This asymmetry confirms that ACD preserves modeling knowledge but EMR alone supplies the procedural scaffolding for translating equations into executable code. The retrieval modality ablation further disentangles each knowledge tier: Text-Only retrieval maintains MR at 8.55 but CE falls to 65.8% because the agent lacks implementation templates, while Code-Only retrieval achieves CE of 91.5% but MR drops to 7.90 because conceptual guidance for model selection is absent. These complementary degradation patterns validate that the tri-level representation is essential and each tier addresses a distinct failure mode.

Table 3: Comprehensive ablation on MM-Bench with GPT-4o backbone. All scores are reported as mean ± std over three evaluation runs.

Variant	AE	MR	PS	RBA	CE (%)	ΔCE	Overall
Module-Level Ablation							
w/o EMR	9.20±.15	8.50±.18	7.80±.22	8.10±.19	55.2	−41.2	8.40
w/o ACD	9.30±.12	7.45±.24	8.60±.16	8.50±.15	94.0	−2.4	8.46
w/o SVE	9.38±.10	8.78±.15	8.45±.17	8.62±.14	79.5	−16.9	8.81
ACD Design Alternatives							
Self-Reflection	9.32±.11	7.68±.21	8.72±.14	8.55±.13	93.5	−2.9	8.57
Homogeneous	9.28±.12	7.92±.19	8.65±.15	8.60±.14	94.2	−2.2	8.61
w/o Moderator	9.35±.10	8.35±.16	8.82±.13	8.72±.12	95.0	−1.4	8.81
EMR Retrieval Modality							
Text-Only	9.25±.13	8.55±.17	8.12±.19	8.35±.16	65.8	−30.6	8.57
Code-Only	9.18±.14	7.90±.20	8.85±.12	8.65±.13	91.5	−4.9	8.65
Sci-Mind (Ours)	**9.45**±.09	**8.92**±.14	**9.15**±.11	**9.05**±.10	**96.4**	—	**9.14**

RQ2: Does ACD escape theoretical local optima? Removing ACD yields the sharpest MR decline of 1.47 while CE stays at 94.0%, confirming that without adversarial verification the agent produces executable but theoretically flawed models. The design alternatives form a monotonic progression: Self-Reflection reaches MR 7.68, Homogeneous Debate 7.92, ACD without Moderator 8.35, and full ACD 8.92. The gap of 1.00 between Homogeneous Debate and full ACD confirms that objective asymmetry is the critical factor, and the Moderator contributes an additional 0.57 by preventing premature termination on suboptimal hypotheses.

RQ3: Does SVE enhance execution robustness? Removing SVE drops CE by 16.9% with only 0.14 MR change, confirming its dedicated role as a verification layer downstream of modeling decisions. This pattern complements the ACD ablation: ACD ensures theoretical soundness while SVE ensures faithful translation into executable code through formal consistency checking.

4.6 Convergence Behavior of ACD

Figure 3 presents the ACD convergence dynamics across all 111 MM-Bench problems. All four backbones exhibit the same qualitative trajectory in panel (a): Γ rises steeply from Round 1 to Round 3 then plateaus into the convergence zone, with DeepSeek-R1 converging fastest at 3.6 rounds on average with 94% convergence rate and Qwen-72B requiring 4.8 rounds at 82%. This ordering aligns with backbone reasoning strength, yet even DeepSeek-R1 starts with U_P well below the quality threshold γ, confirming that adversarial verification remains essential regardless of backbone capability. Panel (b) reveals the underlying mechanism under GPT-4o: at Round 1, U_T reaches 0.82 while U_P is only 0.41, creating a

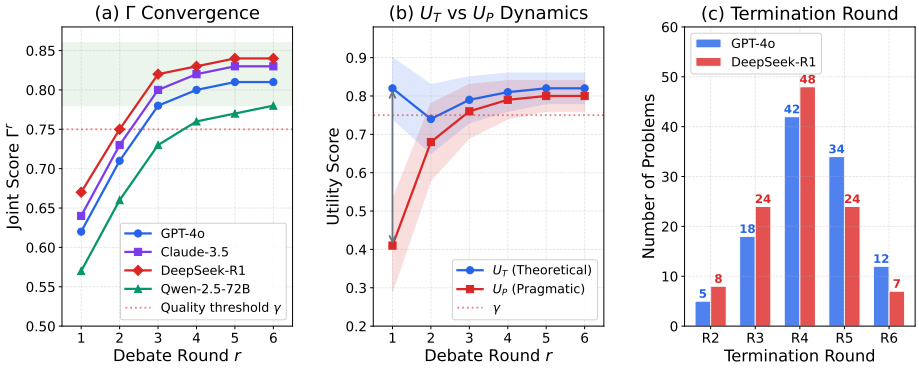

Fig. 3: ACD convergence analysis across 111 MM-Bench problems.

large theory-data gap where the Theorist generates mathematically elegant but practically infeasible models. The Pragmatist's critique rapidly closes this gap in Round 2 by forcing U_P up to 0.68 at the cost of a temporary U_T dip to 0.74 as the Theorist simplifies its formulation to accommodate data constraints. From Round 3 onward both utilities co-ascend with shrinking variance, indicating that later rounds primarily resolve edge cases rather than systematic modeling errors. Panel (c) confirms that the modal termination round is R4 for both GPT-4o and DeepSeek-R1, with only 12 and 7 problems respectively requiring the full $R_{max} = 6$ budget, suggesting that the computational overhead of multi-round debate is bounded and predictable in practice.

4.7 Qualitative Case Study

Figure 4 traces how Sci-Mind resolves a representative modeling challenge, where the task is to predict invasive species spread across 48 counties under three climate scenarios. The dataset provides county-level population counts, environmental covariates, and a binary adjacency matrix, but no continuous spatial coordinates. EMR retrieves historically similar cases that employed a Fisher-KPP PDE, which would be the natural choice for spatial diffusion modeling. However, the system simultaneously recognizes a fundamental incompatibility: the dataset lacks the high-resolution spatial grid required by the Laplacian operator, the adjacency matrix encodes only binary contiguity rather than metric distances, and discrete county-level observations cannot support continuous spatial derivatives. Rather than propagating this infeasible formulation into code, the Adversarial Cognitive Dialectic drives the Theorist to reformulate the dynamics as a discrete population growth model on the county adjacency graph: $\mathbf{n}(t{+}1) = \mathbf{L}(\mathbf{I}{+}\alpha\mathbf{A})\mathbf{n}(t)$, where the adjacency matrix serves directly as the dispersal operator and environmental covariates modulate local logistic growth rates. The Self-Validating Execution module then confirms data availability, dimensional consistency, covariate dependence, and adjacency matrix integrity before

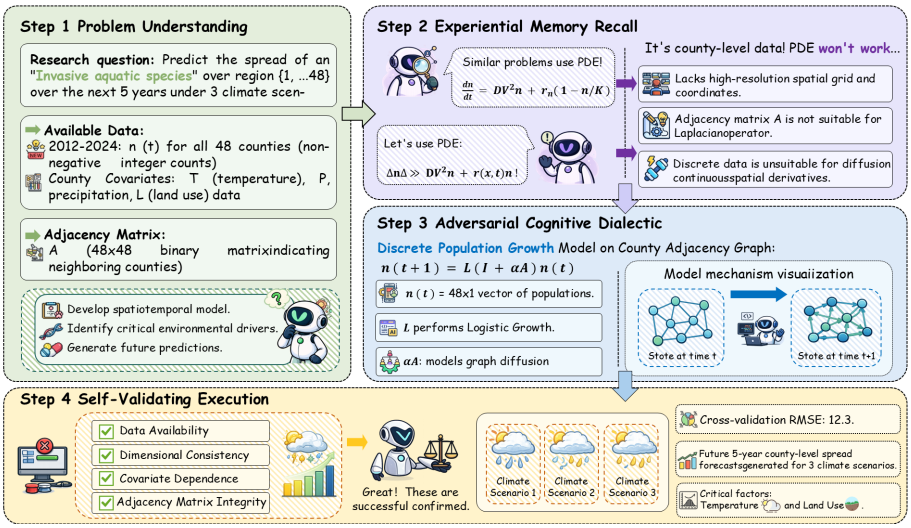

Fig. 4: Case study reasoning trace of Sci-Mind.

code generation. The model executes successfully, generating 5-year county-level spread forecasts for three climate scenarios with cross-validation RMSE of 12.3.

5 Discussion and Conclusion

We presented Sci-Mind, a cognitively-inspired framework that integrates Experiential Memory Recall, Adversarial Cognitive Dialectic, and Self-Validating Execution Strategy for autonomous mathematical modeling. Experiments on MM-Bench and EngiBench across four backbones confirm state-of-the-art performance in both modeling rigorousness and code executability, with gains orthogonal to backbone capabilities. Our analysis reveals two broader insights: objective asymmetry matters more than agent count for effective verification, as the gap between Homogeneous Debate and full ACD far exceeds the gap between Self-Reflection and Homogeneous Debate; and the abstraction-implementation gap is a distinct failure mode where agents possess declarative knowledge to formulate correct models but lack procedural knowledge to implement them, which retrieval of executable structural priors can directly address.

One limitation is that our experiments focus on competition-style modeling tasks, and generalizability to industrial or laboratory scientific workflows with proprietary data and domain-specific constraints remains to be validated. Looking forward, extending the fixed verification predicates to a learnable generation mechanism, and accumulating domain-specific critique templates through finer-grained self-evolution represent promising directions. We hope Sci-Mind provides a useful step toward autonomous agents that reason about scientific problems with the rigor and groundedness of human domain experts.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Anthropic, A.: The claude 3 model family: Opus, sonnet, haiku. Claude-3 Model Card **1**(1), 4 (2024)
3. Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., Abdelrazek, M.: Seven failure points when engineering a retrieval augmented generation system. In: Proceedings of the IEEE/ACM International Conference on AI Engineering-Software Engineering for AI. pp. 194–199 (2024)
4. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research **290**(2), 405–421 (2021)
5. Boiko, D.A., MacKnight, R., Kline, B., Gomes, G.: Autonomous chemical research with large language models. Nature **624**(7992), 570–578 (2023)
6. Chan, C.M., Chen, W., Su, Y., Yu, J., Xue, W., Zhang, S., Fu, J., Liu, Z.: ChatEval: Towards better LLM-based evaluators through multi-agent debate. In: The International Conference on Learning Representations (2024)
7. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.D.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
8. Chen, X., Lin, M., Schärli, N., Zhou, D.: Teaching large language models to self-debug. In: The International Conference on Learning Representations (2024)
9. Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I.: Improving factuality and reasoning in language models through multiagent debate. In: International Conference on Machine Learning. pp. 11733–11763. PMLR (2024)
10. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2023)
11. Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al.: DeepSeek-R1 incentivizes reasoning in llms through reinforcement learning. Nature **645**(8081), 633–638 (2025)
12. Guo, S., Deng, C., Wen, Y., Chen, H., Chang, Y., Wang, J.: DS-Agent: Automated data science by empowering large language models with case-based reasoning. In: International Conference on Machine Learning. pp. 16813–16848. PMLR (2024)
13. Hao, Z., Yao, J., Su, C., Su, H., Wang, Z., Lu, F., Xia, Z., Zhang, Y., Liu, S., Lu, L., et al.: Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. Advances in Neural Information Processing Systems **37**, 76721–76774 (2024)
14. Hong, S., Lin, Y., Liu, B., Liu, B., Wu, B., Zhang, C., Li, D., Chen, J., Zhang, J., Wang, J., et al.: Data interpreter: An LLM agent for data science. In: Findings of the Association for Computational Linguistics. pp. 19796–19821 (2025)
15. Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S.K.S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., Schmidhuber, J.: MetaGPT: Meta programming for a multi-agent collaborative framework. In: The International Conference on Learning Representations (2024)
16. Hu, X., Huang, J., Liu, M., Anmahapong, K., Chen, Y., Luo, Y., Huang, Y., Bai, X., Li, Z., Liao, Y., et al.: Fetalagents: A multi-agent system for fetal ultrasound image and video analysis. arXiv preprint arXiv:2603.09733 (2026)

17. Huang, J., Chen, X., Mishra, S., Zheng, H.S., Yu, A.W., Song, X., Zhou, D.: Large language models cannot self-correct reasoning yet. In: The International Conference on Learning Representations (2024)
18. Jia, J., Liu, Y., Yang, C., Sun, Y., Qin, F., Wang, C., Peng, Y.: Brain-HGCN: A hyperbolic graph convolutional network for brain functional network analysis. arXiv preprint arXiv:2509.14965 (2025)
19. Jia, J., Sun, Y., Liu, Y., Yang, C., Wang, C., Qin, F., Peng, Y., Min, W.: Rt-gmff: Enhanced fmri-based brain disorder diagnosis via roi-driven text generation and multimodal feature fusion. In: International Conference on Bioinformatics and Biomedicine. pp. 2301–2308. IEEE (2025)
20. Jimenez, C.E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., Narasimhan, K.R.: SWE-bench: Can language models resolve real-world github issues? In: International Conference on Learning Representations (2024)
21. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nature Reviews Physics **3**(6), 422–440 (2021)
22. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 6769–6781 (2020)
23. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in Neural Information Processing Systems **35**, 22199–22213 (2022)
24. Kolodner, J.L.: An introduction to case-based reasoning. Artificial Intelligence Review **6**(1), 3–34 (1992)
25. Lai, Y., Li, C., Wang, Y., Zhang, T., Zhong, R., Zettlemoyer, L., Yih, W.t., Fried, D., Wang, S., Yu, T.: DS-1000: A natural and reliable benchmark for data science code generation. In: International Conference on Machine Learning. pp. 18319–18345. PMLR (2023)
26. Langley, P.: Scientific discovery: Computational explorations of the creative processes. MIT press (1987)
27. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
28. Li, G., Hammoud, H., Itani, H., Khizbullin, D., Ghanem, B.: CAMEL: Communicative agents for" mind" exploration of large language model society. Advances in Neural Information Processing Systems **36**, 51991–52008 (2023)
29. Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Shi, S., Tu, Z.: Encouraging divergent thinking in large language models through multi-agent debate. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 17889–17904 (2024)
30. Liu, F., Yang, Z.R., Liu, C., SONG, T., Gao, X., Liu, H.: MM-agent: LLM as agents for real-world mathematical modeling problem. In: The Annual Conference on Neural Information Processing Systems (2025)
31. Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.W., Galley, M., Gao, J.: MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In: International Conference on Learning Representations (2024)
32. M. Bran, A., Cox, S., Schilter, O., Baldassari, C., White, A.D., Schwaller, P.: Augmenting large language models with chemistry tools. Nature Machine Intelligence **6**(5), 525–535 (2024)

33. Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al.: Self-refine: Iterative refinement with self-feedback. Advances in Neural Information Processing Systems **36**, 46534–46594 (2023)
34. Pan, L., Saxon, M., Xu, W., Nathani, D., Wang, X., Wang, W.Y.: Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. Transactions of the Association for Computational Linguistics **12**, 484–506 (2024)
35. Popper, K.: The logic of scientific discovery. Routledge (2005)
36. Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M.P., Dupont, E., Ruiz, F.J., Ellenberg, J.S., Wang, P., Fawzi, O., et al.: Mathematical discoveries from program search with large language models. Nature **625**(7995), 468–475 (2024)
37. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems **36**, 8634–8652 (2023)
38. Valmeekam, K., Marquez, M., Sreedharan, S., Kambhampati, S.: On the planning abilities of large language models-a critical investigation. Advances in Neural Information Processing Systems **36**, 75993–76005 (2023)
39. Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. Transactions on Machine Learning Research (2024)
40. Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al.: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (2024)
41. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
42. Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A.H., White, R.W., Burger, D., Wang, C.: AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In: ICLR Workshop on Large Language Model (LLM) Agents (2024)
43. Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., et al.: Qwen2.5 technical report. arXiv preprint arXiv:2412.15115 (2024)
44. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: ReAct: Synergizing reasoning and acting in language models. In: The International Conference on Learning Representations (2023)
45. Yin, S., You, W., Ji, Z., Zhong, G., Bai, J.: MuMath-Code: combining tool-use large language models with multi-perspective data augmentation for mathematical reasoning. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 4770–4785 (2024)
46. Zan, D., Chen, B., Zhang, F., Lu, D., Wu, B., Guan, B., Yongji, W., Lou, J.G.: Large language models meet NL2Code: A survey. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 7443–7464 (2023)
47. Zhong, L., Wang, Z., Shang, J.: Debug like a human: A large language model debugger via verifying runtime execution step by step. In: Findings of the Association for Computational Linguistics. pp. 851–870 (2024)
48. Zhou, X., Wang, X., He, Y., Wu, Y., Zou, R., Cheng, Y., Xie, Y., Liu, W., Zhao, H., Xu, Y., et al.: EngiBench: A benchmark for evaluating large language models on engineering problem solving. arXiv preprint arXiv:2509.17677 (2025)