# ReasFlow: Assisting Reasoning-Centric Scientific Discovery in Applied Mathematics via a Knowledge-Based Multi-Agent System

**Yutong He** [*1] **Daibo Li** [*1] **Guohong Li** [*1] **Jiahe Geng** [*1] **Zhengyang Huang** [*1] **Can Ren** [*1] **Kun Yuan** [1] **Zaiwen Wen** [1]

## Abstract

Existing autonomous research systems focus on empirically driven domains, while theory-driven discovery, which requires rigorous proofs and deep domain knowledge, remains underexplored. Key challenges include verifying theoretical reasoning at scale, insufficient LLM reasoning capabilities for frontier exploration, and the scarcity of procedural heuristics. We introduce **ReasFlow**, an end-to-end multi-agent system that integrates (i) an internal verification loop for auditing logical derivations, and (ii) an automated knowledge retrieval and self-improvement mechanism that proactively surfaces both declarative facts and overlooked heuristics, substantially reducing expert intervention. ReasFlow unifies literature survey, algorithm design, theorem proving, experimentation, and manuscript writing. Applied to five novel research tasks in distributed optimization with minimal human prompts, ReasFlow consistently outperforms state-of-the-art open-source baselines under rigorous LLM-based review.

## 1. Introduction

Recent advances in LLMs have enabled autonomous agents to tackle complex reasoning tasks, from international Olympiad competitions to automated scientific discovery (Trinh et al., 2024; Lu et al., 2024; Weng et al., 2025). However, these successes remain largely confined to **empirically driven** research, where progress is measured by benchmarks and experimental validation. In contrast, theory-driven research in **applied mathematics**, which demands rigorous theorem proving, convergence guarantees, and complexity characterizations, tightly integrated with application-driven modeling, remains underexplored. Existing automated systems, designed for competition-style problems or benchmark-driven workflows, lack specialized supports necessary for such reasoning-intensive discovery.

Two core challenges hinder automation in this domain. First, verifying theoretical reasoning is intrinsically difficult: proofs in natural language cannot be reliably checked at scale, and evaluating a theoretical contribution requires judging bound tightness, assumption parsimony, and conceptual significance. Second, current LLMs lack deep, subfield-specific knowledge, classical proof techniques, standard bounding arguments, and tacit heuristics, and therefore depend on continuous expert guidance to supply missing context and correct misaligned reasoning.

We address these challenges through two key mechanisms. (1) A robust **internal verification loop** enables the agent to audit its own derivations, checking logical coherence, algebraic consistency, and the validity of inequality relaxations, thereby filtering basic errors before human inspection. (2) A **domain-specific knowledge retrieval** system replaces live expert intervention: the agent actively retrieves relevant heuristics, proof strategies, and methodological conventions, shifting the workflow from expert-driven correction to agent-driven knowledge acquisition. To build this knowledge base without prohibitive manual curation, we devise a self-improving loop that optimizes extraction and retrieval using the agent's own reasoning, evaluated by end-to-end performance, thereby surfacing both declarative facts and tacit procedural heuristics.

Building on these ideas, we introduce **ReasFlow**, an end-to-end autonomous agent system for reasoning-centric discovery in applied mathematics. ReasFlow unifies literature survey, algorithm design, theorem proving, experimental evaluation, and manuscript writing within a human-in-the-loop workflow. We evaluate ReasFlow by generating five complete research papers, each presenting a novel algorithm with convergence guarantees and empirical validation, with a focus on distributed optimization. Compared against several state-of-the-art open-source research agents under identical prompts, ReasFlow achieves the highest scores on theory-oriented quality (Figure 1).

[1]Peking University, Beijing, China. Correspondence to: Zaiwen Wen <wenzw@pku.edu.cn>, Kun Yuan <kunyuan@pku.edu.cn>.

The $3^{rd}$ *AI for Math Workshop* at the $43^{rd}$ International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

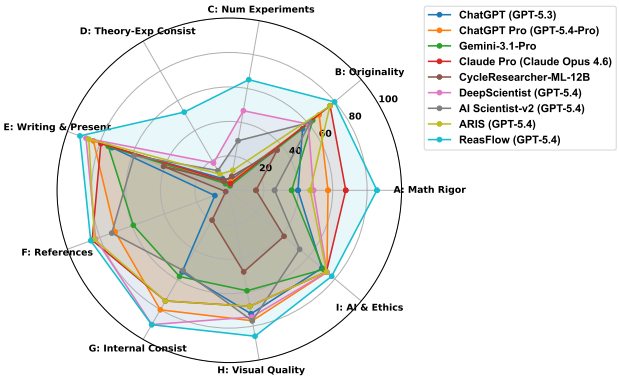


*Figure 1.* Radar chart comparing ReasFlow with baseline workflows across nine evaluation dimensions (A–I).

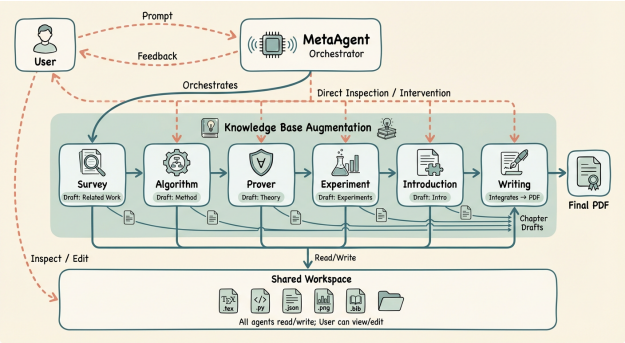


*Figure 2.* Illustration of the ReasFlow system.

Our contributions are: (1) a principled framework combining self-verification and automated knowledge infusion to strengthen theoretical reasoning in AI research agents; (2) the ReasFlow system, which unifies the full applied-mathematics research pipeline; and (3) a comprehensive evaluation demonstrating superior generation quality and theoretical depth over existing agents.

## 2. Related Work

**AI-Assisted Scientific Research.** Recent years have seen rapid progress in end-to-end frameworks that automate the full research lifecycle. AI Scientist (Lu et al., 2024) pioneered closed-loop experimentation and peer review; CycleResearcher (Weng et al., 2024) introduced iterative preference learning for quality refinement; and DeepScientist (Weng et al., 2025) focused on the autonomous discovery of state-of-the-art algorithms. Specialized tools such as PaperQA2 (Skarlinski et al., 2024) and OpenScholar (Asai et al., 2024) further support high-fidelity literature synthesis. Despite these advances, existing systems predominantly target empirically driven research and offer little support for rigorous theorem proving or theoretical derivation. ReasFlow is designed to fill this gap, extending autonomous scientific discovery to theory-intensive domains.

**AI-Assisted Mathematical Reasoning.** LLM-based mathematical reasoning has advanced rapidly from competition problems to research-level challenges. Milestone systems like AlphaGeometry and AlphaProof (Trinh et al., 2024; Hubert et al., 2025) achieved medal-level IMO performance by coupling deep search with formal verification, and a growing line of work integrates LLMs with proof assistants such as Lean to improve reliability. For research-level tasks, Aletheia (Feng et al., 2026) demonstrated fully autonomous paper generation in arithmetic geometry via natural-language iteration, while Matlas (Ju et al., 2026b;a) tightly integrated informal reasoning with Lean to resolve a decade-long open problem. In mathematical discovery, FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Novikov et al., 2025) showed that LLM-guided search can produce new verifiable results. A shared characteristic of these systems is their focus on well-formulated problems with predefined specifications. In contrast, ReasFlow targets open-ended theoretical research, where the appropriate hypotheses and even the form of a meaningful contribution are not known in advance, requiring both rigorous reasoning and scientific judgment.

**LLM-Based Automatic Evaluation and Review.** The strong reasoning and instruction-following capabilities of LLMs have made them increasingly viable as automated reviewers. Foundational work (Zheng et al., 2023) established that LLMs can serve as effective judges, and subsequent research has developed structured multi-model review systems that deliver diverse, professionally rigorous assessments (Chu et al., 2024; Gao et al., 2025; Zhu et al., 2025). These findings motivate the internal verification loops in ReasFlow, where LLM critics iteratively audit logical coherence and algebraic consistency. Ensuring fairness and robustness remains critical; studies (Ye et al., 2024; Lou et al., 2024) show that task-specific distortions can be mitigated through careful rubric design. Building on these insights, ReasFlow adopts a multi-criteria evaluation protocol grounded in explicit scientific rubrics, aiming for impartial and expert-aligned automated review.

## 3. ReasFlow System

We introduce **ReasFlow**, a multi-agent system for reasoning-centric scientific research in applied mathematics, supporting algorithmic design, convergence analysis, and numerical experimentation. ReasFlow encompasses five core stages: (i) literature survey, (ii) algorithm design, (iii) convergence proof, (iv) numerical experiments, and (v) manuscript writing. Each stage is handled by a specialized agent with domain-specific knowledge; users can initiate a project with a simple topic description and optional references, and interactively refine outputs as needed.

### 3.1. System Overview

ReasFlow is orchestrated by a central `MetaAgent`, which interacts with the user to define the research topic, generates a structured plan, and sequentially invokes specialized sub-agents accordingly, including `SurveyAgent`, `AlgorithmAgent`, `ProverAgent`, `ExperimentAgent`, `IntroductionAgent`, and `WritingAgent` (Fig. 2).

Each agent is powered by a large language model and operates in a shared workspace with basic utilities (`ReadFile`, `WriteFile`, `EditFile`, `RunTerminalCmd`). Agents are further equipped with two auxiliary mechanisms: **skills**, which return predefined prompts for in-context capabilities, and **sub-agents**, which enable task delegation with isolated memory. For example, the `ProverAgent` is equipped with a tool for downloading arXiv papers, a skill for search-query generation, and sub-agents for proof generation and verification under clean contexts.

A key component is a built-in repository of **knowledge cards** for applied mathematics, providing domain-specific heuristics beyond the agents' base training. The system can also incorporate user-provided knowledge (see Sec. 3.2).

The default `MetaAgent` workflow proceeds as follows: upon user approval, the `SurveyAgent` conducts a literature survey and drafts the *Related Work* section; the `AlgorithmAgent` generates candidate algorithms with pseudocode and Python implementation; the `ProverAgent` establishes convergence theorems with full proofs; the `ExperimentAgent` designs and executes numerical experiments; finally, the `IntroductionAgent` drafts the introduction, and the `WritingAgent` assembles the complete manuscript. ReasFlow also supports an alternative scheme where users directly interact with sub-agents for flexible control, and a human-in-the-loop mode where domain experts can intervene at any stage.

### 3.2. Knowledge Augmentation

A central limitation of general-purpose LLMs in reasoning-centric tasks is their lack of structured, domain-specific strategies. While large language models like `GPT-5.1` or `Gemini-3-Pro` can attempt convergence proofs, they frequently fail to identify correct analytical pathways without expert guidance.

**Motivating Example.** We observe that LLMs often mishandle stochastic independence: they may apply lemmas requiring independence when the condition is not satisfied, or overlook independence where it holds, yielding unnecessarily weak bounds. For independent stochastic vectors $g_1, \ldots, g_n$ with expectation $g_\star$ and $\mathbb{E}[\|g_i - g_\star\|^2] \leq \sigma^2$, a sharp bound uses independence to obtain $\mathbb{E}[\|\bar{g} - g_\star\|^2] \leq \sigma^2/n$. In contrast, LLMs frequently default to Cauchy's inequality, producing the looser bound $\mathbb{E}[\|\bar{g} - g_\star\|^2] \leq \sigma^2$. While correct proofs can be elicited under direct human supervision, such guidance is impractical at scale.

**Example Knowledge Card**

**Id:** TECH-DISTRIBUTED-VARIANCE-REDUCTION
**Type:** proof technique **Domain:** optimization
**Title:** Distributed Variance Reduction (Linear Speedup)
**Description:** Leverages independence of noise across $N$ workers to reduce aggregated gradient variance by $1/N$, enabling linear speedup.
**Problem Pattern:** Analyzing distributed SGD; need to show $1/N$ variance reduction; Jensen's inequality alone is too loose.
**Structural Prerequisites:** Zero-mean noise; independence across workers; simple-average aggregation.
**Execution Logic:** (1) Decompose into mean and noise; (2) Use independence to eliminate cross-terms $\mathbb{E}[\langle \xi_i, \xi_j \rangle] = 0$; (3) Apply $1/N^2$ scaling to obtain $\sigma^2/N$; (4) Handle bias terms with standard inequalities.
**Expected Outcome:** Variance term scales as $O(\sigma^2/N)$.

*Figure 3.* Example knowledge card extracted by refined skill prompts (abridged).

**Algorithm 1** Auto-Update for `CardGeneratorSkill`

**Input:** Initial prompt $s_0$, reference papers $R$, proof task $T$, $k \leftarrow 0$
Compute baseline proof $p_\star = \Phi_{\text{base}}(R, T)$
**while** true **do**
  Extract knowledge cards $C_k = \Psi(s_k, R)$
  Generate current proof $p_k = \Phi_{\text{cards}}(C_k, T)$
  **if** $\Omega(p_k, p_\star) =$ false **then**
    Update prompt $s_{k+1} = \Upsilon(s_k, R, C_k, p_k, p_\star)$
    $k \leftarrow k + 1$
  **else**
    **break**
  **end if**
**end while**

**Knowledge Cards.** To address this, ReasFlow introduces knowledge cards, *i.e.*, concise text files encoding targeted reasoning patterns and heuristics. For the `ProverAgent`, cards emphasize proof strategies and application conditions; for the `SurveyAgent`, they prioritize factual connections among literature. We posit that a knowledge-augmented agent can match the performance of a non-augmented agent operating under real-time expert guidance.

**Automatic Prompt Refinement.** Constructing effective knowledge cards and retrieving them reliably present two challenges: (1) proof techniques are often implicit, making it difficult to describe their scope correctly; (2) LLMs tend to be overconfident, performing insufficient searches and declining to read retrieved cards. To overcome these, we design automatic refine-

**Algorithm 2** Auto-Update for `CardRetrieverSkill`

**Input:** Initial prompt $s_0$, proof task $T$, ground-truth cards $C_\star$, $k \leftarrow 0$
Compute baseline proof $p_\star = \Phi_{\text{cards}}(C_\star, T)$
**while** true **do**
  Generate query $q_k = \Theta(T, s_k)$
  Retrieve knowledge cards $C_k = \Lambda(q_k)$
  Generate current proof $p_k = \Phi_{\text{cards}}(C_k, T)$
  **if** $\Omega(p_k, p_\star) =$ false **then**
    Update prompt $s_{k+1} = \tilde{\Upsilon}(s_k, C_k, C_\star, p_k, p_\star)$
    $k \leftarrow k + 1$
  **else**
    **break**
  **end if**
**end while**

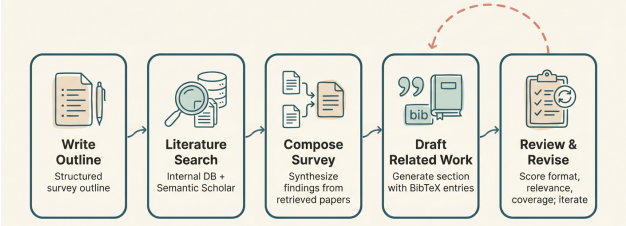


*Figure 4.* Workflow illustration of the `SurveyAgent`.

ment procedures for the `CardGeneratorSkill` and `CardRetrieverSkill`.

Algorithm 1 formalizes the refinement of card generation: starting from an initial prompt $s_0$ and reference papers $R$, we iteratively extract cards $C_k$, generate a proof $p_k$, compare it against a baseline proof $p_\star$ obtained with full access to $R$, and update the prompt $s_k$ until $p_k$ matches or exceeds $p_\star$. Algorithm 2 similarly refines the retrieval prompt by comparing cards retrieved via learned queries against ground-truth card sets.

**Remark.** Our experiments confirm that after prompt refinement, the `ProverAgent` with knowledge cards achieves proof quality comparable to direct access to original references. This validates the knowledge augmentation method in two ways: it surfaces implicit high-level insights through LLM-driven comprehension rather than surface extraction, and the autonomous verification loop safeguards against errors in published literature. More broadly, this demonstrates the potential to bridge informal scholarly discourse and rigorous, verifiable inference in specialized domains.

### 3.3. Agent Specifications

ReasFlow comprises six specialized sub-agents, each responsible for a distinct stage of the research pipeline. Below we summarize their core responsibilities and high-level workflows; detailed tool specifications and workflow descriptions are provided in Appendix A.

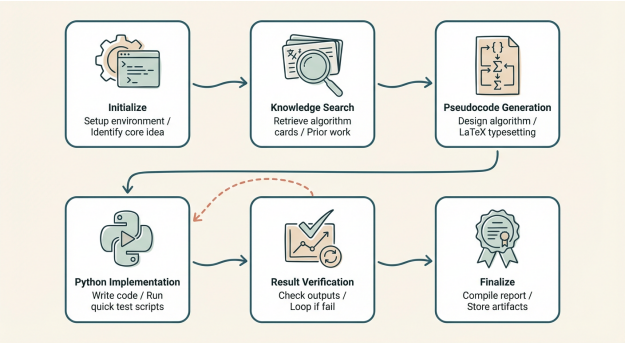


*Figure 5.* Workflow illustration of the `AlgorithmAgent`.

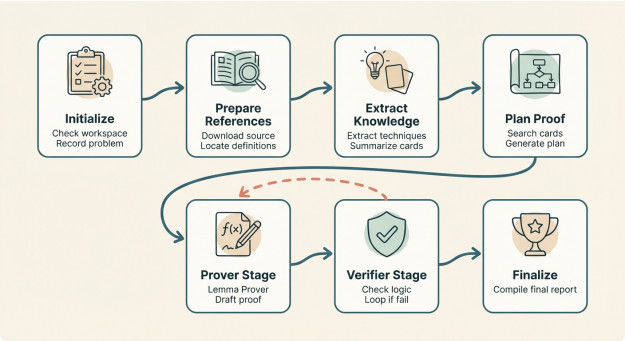


*Figure 6.* Workflow illustration of the `ProverAgent`.

**`SurveyAgent`.** The `SurveyAgent` conducts comprehensive literature reviews, retrieving papers from both a built-in FAISS vector database and online sources via the Semantic Scholar API. Its workflow proceeds through four stages: (i) outline writing, (ii) survey composition with literature search, (iii) *Related Work* drafting with automatic bibtex formatting, and (iv) automated quality review along four dimensions (format integrity, citation relevance, reference coverage, and instruction following), with iterative revision until the review score meets an acceptance threshold (Fig. 4). Knowledge cards for each paper, indexed with semantic embeddings, encapsulate both metadata and analytical insights such as strengths and weaknesses, enhancing retrieval precision and description accuracy.

**`AlgorithmAgent`.** The `AlgorithmAgent` designs, implements, validates, and reports a new algorithm end-to-end. Its workflow proceeds through initialization, knowledge search, pseudocode generation (via `LatexWriterAgent`), Python implementation with quick experiments and visualization, result verification, and final report compilation (Fig. 5). Knowledge cards capture key algorithmic design elements-pseudocode, advantages, and convergence properties-to accelerate principled design choices and guide automated verification.

**`ProverAgent`.** The `ProverAgent` establishes convergence theorems with mathematically rigorous proofs via natural language reasoning. Its workflow includes initialization, reference preparation (downloading papers and locating

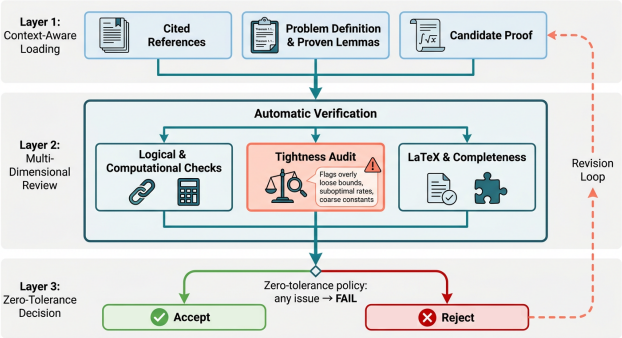


*Figure 7.* Illustration of the automatic verification procedure.

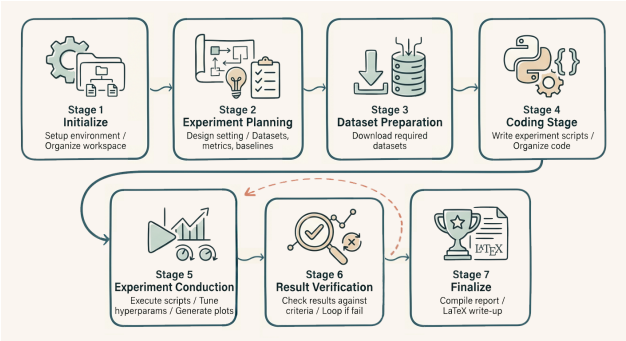


*Figure 8.* Workflow illustration of the `ExperimentAgent`.

relevant content), optional knowledge extraction (creating reusable knowledge cards via `CardGeneratorSkill`), proof planning with knowledge card retrieval, lemma-level proof generation using `LemmaProverAgent`, and verification by `LemmaVerifierAgent` (Fig. 6). The verifier performs context-aware checking against problem descriptions and prior lemmas, explicitly audits bound tightness, and enforces a zero-tolerance policy where any flaw triggers a revision loop (Fig. 7). Proofs are processed one statement at a time to prevent error propagation. Knowledge augmentation for the `ProverAgent` addresses the gap between plausible-looking arguments and truly rigorous proofs, as detailed in Sec. 3.2.

**`ExperimentAgent`.** The `ExperimentAgent` designs, implements, conducts, and verifies computational experiments, producing reproducible code, publication-ready figures, and a draft *Experiment* section. Its workflow spans initialization, experiment planning (dataset/metric/baseline selection via knowledge cards), dataset download, coding, experiment conduction with automatic hyperparameter tuning, result verification, and final report compilation (Fig. 8). Knowledge cards provide conventional experimental design choices and baseline algorithm pseudocode, enabling standards-aligned planning and efficient implementation without consulting full original papers.

**`IntroductionAgent`.** The `IntroductionAgent` synthesizes outputs from prior agents into a comprehensive *Introduction* section. It first uses dedicated extraction tools

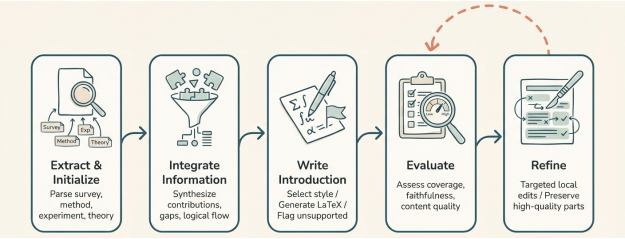


*Figure 9.* Workflow illustration of the `IntroductionAgent`.

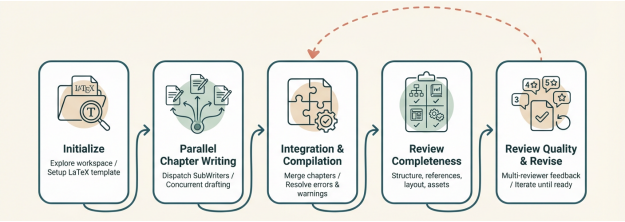


*Figure 10.* Workflow illustration of the `WritingAgent`.

to parse survey reports, method descriptions, experimental results, and theoretical contributions into structured summaries. The workflow proceeds through extraction, information integration, style-adaptive writing (supporting ML, Math, and Default templates), and an automated evaluation–refinement loop that assesses coverage, faithfulness, and content quality, followed by targeted local edits (Fig. 9). A knowledge card database, similar to the `SurveyAgent`'s, enables accurate and fair description of cited work.

**`WritingAgent`.** The `WritingAgent` assembles and polishes all draft materials into a publication-ready manuscript using a two-tier architecture: a `Coordinator` plans the paper structure, dispatches chapter-level tasks, and integrates the final manuscript, while multiple `SubWriter` instances work on individual chapters in parallel. The workflow proceeds through initialization (workspace exploration and LaTeX template setup), parallel chapter writing with incremental cross-chapter coherence editing, integration and compilation with iterative error/warning resolution, and a review-and-revision loop using completeness and writing-quality reviewers followed by page-level visual inspection (Fig. 10). Knowledge augmentation provides modular skill documents covering asset-handling, venue-specific formatting rules, and chapter-compilation procedures.

## 4. Experiment

We evaluate ReasFlow's ability to assist reasoning-centric research in applied mathematics, focusing on the domain of distributed optimization. We first generate five complete papers with varying human-interaction modes, then compare ReasFlow against several publicly available AI research systems on full-paper generation, and finally conduct separate evaluations of individual sub-agents.

*Table 1.* Full-paper generation scores. **Bold**: best; underlined: second best.

| AI Agent \Reviewer Backend | GPT-5.1 | GPT-5.4 | Gemini-3-Pro | Average |
|---|---|---|---|---|
| ChatGPT (GPT-5.3) | 51.7 | 32.2 | 29.8 | 37.9 |
| ChatGPT Pro (GPT-5.4-Pro) | 62.4 | 44.5 | 41.9 | 49.6 |
| Gemini (Gemini-3.1-Pro) | 51.5 | 31.4 | 35.1 | 39.3 |
| Claude Pro (Claude Opus 4.6) | 64.4 | 32.9 | 61.4 | 52.9 |
| CycleResearcher-ML-12B | 34.6 | 19.1 | 10.1 | 21.2 |
| DeepScientist (GPT-5.4) | 71.9 | 56.6 | 43.5 | 57.3 |
| AI Scientist-v2 (GPT-5.4) | 60.6 | 34.7 | 29.6 | 41.6 |
| ARIS (GPT-5.4) | 59.3 | 52.9 | 38.6 | 50.3 |
| ReasFlow (GPT-5.4) | **80.5** | **65.6** | **87.6** | **77.9** |

*Table 2.* SurveyAgent vs. base LLM (averaged across three evaluator models).

| Task | Backbone | Base | ReasFlow | Adv. |
|---|---|---|---|---|
| He et al. (2026) | DeepSeek-v3.2 | 12.65 | 17.93 | +5.27 |
| | GPT-5.1 | 19.20 | 17.78 | −1.41 |
| | GPT-5.4 | 9.08 | 16.75 | +7.67 |
| | Claude Sonnet 4.6 | 11.07 | 17.24 | +6.17 |
| | Average | 13.00 | 17.42 | +4.42 |
| Zhu et al. (2026) | DeepSeek-v3.2 | 13.28 | 14.03 | +0.75 |
| | GPT-5.1 | 14.52 | 16.63 | +2.11 |
| | GPT-5.4 | 13.23 | 16.15 | +2.92 |
| | Claude Sonnet 4.6 | 14.03 | 14.90 | +0.88 |
| | Average | 13.77 | 15.43 | +1.66 |
| Zhang et al. (2026) | DeepSeek-v3.2 | 8.14 | 18.36 | +10.23 |
| | GPT-5.1 | 7.00 | 16.03 | +9.03 |
| | GPT-5.4 | 12.50 | 13.67 | +1.17 |
| | Claude Sonnet 4.6 | 17.32 | 11.62 | −5.70 |
| | Average | 11.24 | 14.92 | +3.68 |
| Overall Average | | 12.67 | 15.92 | **+3.26** |

### 4.1. Case Studies: End-to-End Discovery

We conducted five case studies spanning distributed and federated optimization. In Cases 1-3, experts interacted directly with specialized sub-agents (direct-prompting); in Cases 4-5, experts interacted only with the MetaAgent (orchestrated workflow). All five papers were produced with minimal human refinement (polishing only). Below we highlight the main challenges and outcomes.

- **Case 1:** The first paper (He et al., 2026) proposes and studies a novel algorithm that incorporate low-rank gradient projection to federated learning to reduce client-side memory for optimizer states, which simultaneously saves communication between clients and the server. Building upon (Yang et al., 2021) and (He et al., 2025), the AlgorithmAgent designed the algorithm, and the ProverAgent established convergence at a rate of $\mathcal{O}(1/\sqrt{NT})$, explicitly proving linear speedup in the number of clients. The ExperimentAgent evaluated the method on MNIST with a three-layer MLP. At a compression ratio of $1/7$, the proposed algorithm achieved a final test accuracy of $0.9511$, comparable to uncompressed FedAvg-M ($0.9551$) and substantially outperforming baselines including FedLoRA-M, FedMef, NeuLite, and FederatedSelect. Ablation studies further demonstrated robustness under high data heterogeneity.
- **Case 2:** The second paper (Zhu et al., 2026) demands compositional reasoning: merging the heterogeneity correction mechanism of SCAFFOLD with subspace projection methods that simultaneously improve communication, computation, and memory efficiency. The AlgorithmAgent formulated a backfill-style update that preserves residual control information when the active subspace rotates. The ProverAgent derived a non-

*Table 3.* Algorithm script first-run success rate. AI Scientist-v2 is default with `Sonnet 4.6` and `GPT-4o-20241120`. S.T. represents successful tasks, S.R. represents first-run success rate. **Bold**: best; underlined: second best.

| System | S.T. | S.R. |
|---|---|---|
| ReasFlow (`GPT-5.4`) | 35 | **100%** |
| ReasFlow (`GPT-5.1`) | 35 | 94% |
| Codex-CLI (`GPT-5.4`) | 35 | 97% |
| Claude Code (`Sonnet 4.6`) | 35 | 71% |
| AI Scientist-v2 (Default) | 35 | 60% |
| ARIS (`Sonnet 4.6`) | 35 | 44% |

asymptotic convergence rate of $\tilde{\mathcal{O}}(1/T + 1/\sqrt{NKT})$, establishing linear speedup in both client participation and iterations. The `ExperimentAgent` validated the algorithm on two fronts: a controlled matrix-regression toy problem and a ResNet-110 benchmark on CIFAR-100. Over 100 training epochs, the proposed method achieved a final test accuracy of 45.43%, surpassing FedAvg (34.35%) and FedSub (23.17%), while narrowing the gap to full-dimensional SCAFFOLD (52.92%). Ablation studies confirmed stability across varying subspace dimensions and data heterogeneity.

- **Case 3:** The third paper (Zhang et al., 2026) presents the most intricate theoretical challenge: extending the Muon optimizer, which relies on the non-linear matrix sign operator $\mathrm{msgn}$, to sparse peer-to-peer networks where $\mathrm{msgn}$ does not commute with gossip averaging. The `AlgorithmAgent` proposed a unified primal-dual framework parameterized by $(A, B, C)$ that recovers EXTRA and Gradient Tracking as special cases while ensuring polarization occurs after tracking. The `ProverAgent` delivered the central theoretical contribution: a topology-separated convergence rate of $\mathcal{O}((1 + \sigma/\sqrt{N})K^{-1/4})$ in nuclear-norm geometry. Crucially, the leading term is independent of the network graph, with the spectral gap affecting only fast-decaying transients. The `ExperimentAgent` found regime-dependent graph effects: SUDA variants match DeMuon on ring benchmark and near-IID GPT-2/Wikitext-2, while on 20-node, 100-epoch non-IID CIFAR-100 both outperform DeMuon.
- **Case 4:** The fourth paper (Sun & Yuan, 2026) proposes a new accelerated algorithm for decentralized stochastic optimization by unifying multi-round communication and optimization acceleration. Under orchestration by the `MetaAgent`, the study builds on prior work in accelerated distributed methods and fast gossip (Jakovetić et al., 2014; Scaman et al., 2018; Berthier et al., 2020) to design *Multi-Gossip Accelerated DSGD*, which couples a Nesterov-type primal-dual extrapolation with repeated fast gossip averaging per outer iteration. Convergence is established under smoothness, strong convexity, and bounded-variance assumptions, yielding the non-asymptotic rate

$$\widetilde{\mathcal{O}}\left(\frac{\sigma^2}{\mu n \epsilon} + \sqrt{\frac{L}{\mu}}\frac{1}{\sqrt{1-\beta}}\right),$$

which attains the best known communication-complexity scaling up to logarithmic factors. The analysis constructs a coupled Lyapunov function capturing both optimization progress and network disagreement, combining the acceleration from primal-dual iteration with the contraction of multi-round gossip to produce the final guarantee and the optimal communication depth guideline $R = \widetilde{\mathcal{O}}(1/\sqrt{1-\beta})$.
- **Case 5:** The fifth paper (Li & Hu, 2026) combines a retraction-free method with an EXTRA-type primal-dual recursion to improve both computational and communication efficiency. Within the `MetaAgent`-coordinated workflow, the resulting RF-EXTRA algorithm embeds an approximate gradient mapping for orthogonality-constrained optimization into the EXTRA architecture, preserving simple decentralized communication while controlling feasibility in ambient space. The theoretical analysis establishes an exact $\mathcal{O}(1/K)$ convergence rate under a constant stepsize, using a joint-error contraction inequality and a careful study of primal descent on the original objective function. Experimental evaluation on decentralized PCA and low-rank matrix completion demonstrates that RF-EXTRA compares favorably with reported decentralized baselines and exhibits strong communication efficiency on the tested Stiefel manifold tasks.

These cases collectively demonstrate ReasFlow's capacity to produce rigorous theoretical results and empirical validations across a spectrum of difficulty, from variance-bound tightening to non-commutativity management and manifold-constrained optimization.

### 4.2. Evaluation on Automatic Full-Paper Generation

We conduct a controlled experiment for end-to-end manuscript generation of He et al. (2026). All systems receive an identical initial prompt (core idea and seminal references, see Appendix B) and are allowed a single autonomous trial. ReasFlow's `MetaAgent` (powered by `GPT-5.4`) orchestrates the pipeline.

Baselines include proprietary web-based systems (ChatGPT, ChatGPT Pro, Gemini, Claude Pro), *DeepScientist* (Weng et al., 2025), *CycleResearcher* (Weng et al., 2024), *AI-Scientist-v2* (Yamada et al., 2025), and *ARIS* (Yang et al., 2026). All manuscripts are evaluated by a `PaperReviewAgent` across nine dimensions (mathematical rigor, originality, experiments, theory-experiment consistency, writing, citations, logical coherence, formatting, and ethics compliance). Detailed rubric in Appendix B.

As shown in Table 1, ReasFlow achieves an average score of **77.9**, outperforming the strongest baseline (DeepScien-

*Table 4.* ProverAgent evaluation scores. **Bold**: best; underlined: second best.

| AI Agent | Reasoning | GPT-5.1 | GPT-5.4 | Gemini-3-Pro | Average |
|---|---|---|---|---|---|
| ChatGPT (GPT-5.3) | Default | 29 | 15 | 7 | 17 |
| Gemini (Gemini-3.1-Pro) | Default | 81 | 24 | 48 | 51 |
| ReasLingo (Gemini-3-Pro) | High | 91 | 48 | 40 | 59.7 |
| ReasLingo (GPT-5.1) | High | 37 | 35 | 40 | 37.3 |
| ReasLingo (GPT-5.4) | High | 92 | 57 | 60 | 69.7 |
| ProofGrader (GPT-5.1) | High | 92 | 34 | 70 | 65.3 |
| ProofGrader (GPT-5.4) | High | 93 | 39 | 75 | 69 |
| QED (GPT-5.4) | High | **98** | 51 | 70 | 73 |
| ReasFlow (Gemini-3-Pro) | Medium | 91 | 62 | **100** | 84.3 |
| ReasFlow (GPT-5.1) | Medium | 93 | 45 | **100** | 79.3 |
| ReasFlow (GPT-5.4) | Medium | 94 | **78** | **100** | **90.7** |

*Table 5.* Experiment Execution scores (1–10) on Soundness (Sdn.), Consistency (Cst.), Completeness (Cpl.), Novelty (Nvl.), Insightfulness (Isf.), Significance (Sgn.), Design Soundness (D.S.), Design Completeness (D.C.), Algorithm–Experiment Alignment (AEA.), Experimental Effectiveness (E.E.), Writeup Readability (W.R.), and Hallucination Suspicion (H.S.). Scores are averaged over $n$ successfully completed tasks. **Bold**: best; underline: second best.

| System | $n$ | Sdn. | Cst. | Cpl. | Nvl. | Isf. | Sgn. | D.S. | D.C. | AEA. | E.E. | W.R. | H.S. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ReasFlow (GPT-5.4) | 35 | **8.49** | **8.23** | **7.97** | 6.42 | **7.85** | **4.99** | **7.99** | **8.00** | **8.95** | **8.07** | **8.17** | **9%** |
| ReasFlow (GPT-5.1) | 35 | 7.34 | 6.87 | 6.18 | 5.29 | 6.41 | 3.48 | 6.91 | 6.60 | 7.53 | 6.11 | 6.83 | 39% |
| ARIS (Claude Sonnet 4.6) | 35 | 5.32 | 5.21 | 4.77 | 5.51 | 4.93 | 3.35 | 6.23 | 5.70 | 6.84 | 5.04 | 4.59 | 46% |
| Codex (GPT-5.4) | 35 | 6.50 | 6.55 | 6.17 | 4.81 | 5.95 | 3.44 | 6.01 | 5.91 | 6.61 | 5.83 | 5.65 | 26% |
| Claude Code (Claude Sonnet 4.6) | 35 | 5.91 | 5.68 | 5.53 | 4.98 | 5.42 | 3.40 | 5.86 | 5.72 | 6.50 | 5.40 | 5.11 | 43% |
| AI Scientist-v2 (Default) | 35 | 1.28 | 3.07 | 1.72 | 5.64 | 4.94 | 2.35 | 6.39 | 6.14 | 7.09 | 3.16 | 3.84 | 100% |

*Table 6.* IntroductionAgent vs. base model (mean over 5 runs, averaged across evaluators).

| Paper | Metric | Base | RF | Adv. |
|---|---|---|---|---|
| He et al. (2026) | OB | 0.580 | 0.739 | +27.3% |
| | OTF | 0.547 | 0.719 | +31.6% |
| Zhu et al. (2026) | OB | 0.578 | 0.663 | +14.7% |
| | OTF | 0.538 | 0.625 | +16.3% |
| Zhang et al. (2026) | OB | 0.511 | 0.679 | +32.9% |
| | OTF | 0.466 | 0.657 | +40.9% |
| Overall Avg. | OB | 0.557 | 0.694 | +24.6% |
| | OTF | 0.517 | 0.667 | **+29.1%** |

tist, 57.3) by over 20 points. The advantage holds across all three reviewer backends and is especially prominent in mathematical rigor and theory-experiment consistency.

### 4.3. Evaluation on Section-Level Generation

We evaluate each specialized agent individually; detailed setups are in Appendix B.

#### 4.3.1. Survey Agent

The SurveyAgent is compared against the base LLM on three survey tasks using four backbones, evaluated on Content Accuracy and Citation Relevance (scale 1-10 each). Table 2 shows an overall advantage of **+3.26** points, driven by higher factual accuracy (+2.92 on average). The agent wins in **28/36** paired comparisons (77.8% win rate).

#### 4.3.2. Algorithm Agent

Following MLR-Bench (Chen et al., 2026), we introduce **OR-Bench**, a benchmark designed to evaluate AI agents' algorithmic design and implementation capabilities, with a focus on open-ended optimization research. We compare the AlgorithmAgent against ARIS, Codex, Claude Code, and AI Scientist-v2 across 35 tasks (20 from MLR-Bench and 15 from OR-Bench). Given the same initial prompt, each agent is required to correctly design and implement the target algorithm within 3,600 seconds. As reported in Table 3, ReasFlow with GPT-5.4 achieves a leading first-run success rate of 100%, compared with 44%-97% for the baselines, demonstrating the robustness of ReasFlow's AlgorithmAgent.

#### 4.3.3. Prover Agent

The ProverAgent generates a full convergence analysis for Zhu et al. (2026), a novel algorithm without prior public theory. Baselines include general-purpose agents,

*Table 7.* WritingAgent evaluation on He et al. (2026) (equal-weight mean across dimensions and reviewers).

| System (Executor, Skills, Model) | I.Faith. | II.Cov. | III.Write. | IV.Subm. | Avg. |
|---|---|---|---|---|---|
| Claude Code + ARIS (`GPT-5.1`) | 7.67 | 6.33 | 5.42 | 2.00 | 5.36 |
| Codex, vanilla (`GPT-5.4`) | 6.47 | 5.33 | 4.58 | 6.71 | 5.77 |
| Codex + ARIS (`GPT-5.4`) | 8.47 | 4.33 | 5.58 | 7.24 | 6.41 |
| ReasLingo (`GPT-5.4`) | 8.47 | 4.80 | 5.08 | 7.87 | 6.56 |
| Claude Code, vanilla (`GPT-5.1`) | 7.87 | 7.20 | 7.17 | 4.62 | 6.72 |
| Codex, vanilla (`GPT-5.1`) | 7.67 | 5.67 | 7.33 | 7.82 | 7.12 |
| ReasLingo (`GPT-5.1`) | 8.47 | 7.60 | 6.25 | 8.84 | 7.79 |
| Claude Code + ARIS (`GPT-5.4`) | 9.07 | 8.13 | 7.33 | 7.56 | 8.02 |
| Codex + ARIS (`GPT-5.1`) | 7.60 | 8.07 | 7.75 | 8.89 | 8.08 |
| Claude Code, vanilla (`GPT-5.4`) | 8.20 | 9.13 | 7.67 | **10.00** | 8.75 |
| ReasFlow `WritingAgent` (`GPT-5.1`) | **9.73** | **9.47** | **8.16** | 9.87 | **9.31** |

ReasLingo (an agentic system built on Gemini CLI), ProofGrader (Ma et al., 2025), and QED (An et al., 2026). All systems use a single trial; ReasFlow agents run at `medium` reasoning effort.

A `ProofReviewAgent` scores reports on five dimensions (formatting, relevance, consistency, logical correctness, significance). Table 4 shows ReasFlow (`GPT-5.4`) achieves the highest average of **90.7**, surpassing the best baseline (QED, 73) by 17.7 points. Notably, ReasFlow attains perfect 100 scores from the `Gemini-3-Pro` evaluator across all backbones. Per-factor breakdowns are in Appendix 4.3.3.

#### 4.3.4. Experiment Agent

We evaluate the `ExperimentAgent` under a setup similar to Sec. 4.3.2. Each agent receives its own algorithm design outputs and must execute real experiments within 7,200s. Evaluations are conducted by `GPT-5.4` and `Claude Sonnet 4.6` judges (each performing three independent runs) across soundness, consistency, completeness, novelty, insightfulness, significance, design soundness, design completeness, algorithm-experiment alignment, experimental effectiveness, and writeup readability (1-10 scale, averaged). A binary hallucination flag is raised if the agent's workspace lacks evidence of genuine execution (*e.g.*, result files, logs) or if the available outputs appear trivial or synthetic. As shown in Table 5, ReasFlow with `GPT-5.4` outperforms baselines on most dimensions, notably in design-related aspects, while also incurring the lowest hallucination suspicion, thereby demonstrating robustness and efficiency.

#### 4.3.5. Introduction Agent

The `IntroductionAgent` is compared against the base model (`GPT-4o`) across the three papers, with 5 independent runs each. Three evaluators assess Content Coverage, Faithfulness, and Content Quality; two composite scores are reported: Overall Balanced (OB) and Overall Technical Focus (OTF). Table 6 shows average gains of **+24.6%** on OB and **+29.1%** on OTF, with the largest improvement on the hardest paper (Zhang et al., 2026). Per-evaluator breakdowns are in Appendix B.6.

#### 4.3.6. Writing Agent

The `WritingAgent` assembles all assets of He et al. (2026) into a compilable CSIAM-AM manuscript. Eleven systems are compared (Claude Code, Codex, ReasLingo, with various skill packs and backends). Evaluation covers Faithfulness, Coverage, Writing Quality, and Submittability (detailed scoring in Appendix B.7).

Table 7 shows ReasFlow achieves the highest average of **9.31**, leading the runner-up by 0.56 and ranking first on three of four dimensions.

## 5. Conclusion and Future Work

We introduced **ReasFlow**, a multi-agent system for reasoning-centric research in applied mathematics. It combines an internal verification loop with automated knowledge-card retrieval to reduce reliance on expert supervision. Across five case studies in distributed optimization, ReasFlow produced rigorous theoretical results and consistently outperformed open-source research agents under LLM-based review.

Current limitations, including a domain-specific knowledge base and the absence of formal proof guarantees, motivate future work on generalizable knowledge extraction and integration with interactive theorem provers. ReasFlow is publicly available at the ReasLab platform[1] as a collaborative workspace to accelerate theory-driven research.

[1] https://model.reaslab.io

## Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2024YFA1012902).

## References

An, C., Ye, Q., Pan, M., and Zhang, J. Qed: An open-source multi-agent system for generating mathematical proofs on open problems. *arXiv preprint arXiv:2604.24021*, 2026.

Asai, A., He, J., Shao, R., Shi, W., Singh, A., Chang, J. C., Lo, K., Soldaini, L., Feldman, S., D'arcy, M., et al. Openscholar: Synthesizing scientific literature with retrieval-augmented LMs. *arXiv preprint arXiv:2411.14199*, 2024.

Berthier, R., Bach, F., and Gaillard, P. Accelerated gossip in networks of given dimension using jacobi polynomial iterations. *SIAM Journal on Mathematics of Data Science*, 2(1):24–47, 2020. doi: 10.1137/19M1244822.

Chen, H., Xiong, M., Lu, Y., Han, W., Deng, A., He, Y., Wu, J., Li, Y., Liu, Y., and Hooi, B. Mlr-bench: Evaluating ai agents on open-ended machine learning research. *Advances in Neural Information Processing Systems*, 38, 2026.

Chu, Z., Ai, Q., Tu, Y., Li, H., and Liu, Y. Pre: A peer review based large language model evaluator. *arXiv preprint arXiv:2401.15641*, 2024.

Feng, T., Trinh, T. H., Bingham, G., Hwang, D., Chervonyi, Y., Jung, J., Lee, J., Pagano, C., Kim, S.-h., Pasqualotto, F., et al. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026.

Gao, X., Ruan, J., Zhang, Z., Gao, J., Liu, T., and Fu, Y. Reviewagents: Bridging the gap between human and AI-generated paper reviews. *arXiv preprint arXiv:2503.08506*, 2025.

He, Y., Li, P., Hu, Y., Chen, C., and Yuan, K. Subspace optimization for large language models with convergence guarantees. In *International Conference on Machine Learning*, pp. 22468–22522. PMLR, 2025.

He, Y., Huang, Z., and Geng, J. Fedslop: Memory-efficient federated learning with low-rank gradient projection. *arXiv preprint arXiv:2604.24012*, 2026.

Hubert, T., Mehta, R., Sartran, L., Horváth, M. Z., Žužić, G., Wieser, E., Huang, A., Schrittwieser, J., Schroecker, Y., Masoom, H., et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pp. 1–3, 2025.

Jakovetić, D., Xavier, J., and Moura, J. M. F. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014. doi: 10.1109/TAC.2014.2298712.

Ju, H., Chen, L., Wu, P., Dai, B., and Dong, B. Matlas: A semantic search engine for mathematics. *arXiv preprint arXiv:2604.17484*, 2026a.

Ju, H., Gao, G., Jiang, J., Wu, B., Sun, Z., Chen, L., Wang, Y., Wang, Y., Wang, Z., He, W., et al. Automated conjecture resolution with formal verification. *arXiv preprint arXiv:2604.03789*, 2026b.

Li, S. and Hu, J. A retraction-free extra method for decentralized optimization on the stiefel manifold. *arXiv preprint arXiv:2604.23754*, 2026.

Lou, R., Xu, H., Wang, S., Du, J., Kamoi, R., Lu, X., Xie, J., Sun, Y., Zhang, Y., Ahn, J. J., et al. Aaar-1.0: Assessing AI's potential to assist research. *arXiv preprint arXiv:2410.22394*, 2024.

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

Ma, W., Cojocaru, A., Kolhe, N., Louie, B., Sharif, R. S., Zhang, H., Zhuang, V., Zaharia, M., and Min, S. Reliable fine-grained evaluation of natural language math proofs. *arXiv preprint arXiv:2510.13888*, 2025.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

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.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. *Proceedings of the 35th International Conference on Machine Learning*, 80: 3027–3036, 2018.

Skarlinski, M. D., Cox, S., Laurent, J. M., Braza, J. D., Hinks, M., Hammerling, M. J., Ponnapati, M., Rodriques, S. G., and White, A. D. Language agents achieve superhuman synthesis of scientific knowledge. *arXiv preprint arXiv:2409.13740*, 2024.

Sun, M. and Yuan, K. Accelerated decentralized stochastic gradient descent for strongly convex optimization. *arXiv preprint arXiv:2606.07496*, 2026.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Weng, Y., Zhu, M., Bao, G., Zhang, H., Wang, J., Zhang, Y., and Yang, L. Cycleresearcher: Improving automated research via automated review. *arXiv preprint arXiv:2411.00816*, 2024.

Weng, Y., Zhu, M., Xie, Q., Sun, Q., Lin, Z., Liu, S., and Zhang, Y. Deepscientist: Advancing frontier-pushing scientific findings progressively. *arXiv preprint arXiv:2509.26603*, 2025.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Yang, H., Fang, M., and Liu, J. Achieving linear speedup with partial worker participation in non-iid federated learning. *Proceedings of ICLR*, 2021.

Yang, R., Li, Y., and Li, S. Aris: Autonomous research via adversarial multi-agent collaboration. *arXiv preprint arXiv:2605.03042*, 2026.

Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P.-Y., et al. Justice or prejudice? quantifying biases in LLM-as-a-judge. *arXiv preprint arXiv:2410.02736*, 2024.

Zhang, H., Kong, B., Geng, J., and Huang, Z. Suda-muon: Structural design principles and boundaries for fully decentralized muon. *arXiv preprint arXiv:2604.23980*, 2026.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. *Advances in neural information processing systems*, 36: 46595–46623, 2023.

Zhu, M., Weng, Y., Yang, L., and Zhang, Y. Deepreview: Improving LLM-based paper review with human-like deep thinking process. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29330–29355, 2025.

Zhu, S., Huang, Z., Xu, Y., and Li, P. Subspace optimization for efficient federated learning under heterogeneous data. *arXiv preprint arXiv:2604.25467*, 2026.

# A. Detailed Agent Specifications

This appendix catalogs the tools and detailed default workflows of each specialized agent in ReasFlow, as referenced in Sec. 3.3.

## A.1. Survey Agent

**Tools:**

1. `BasicTools`: Read, write, edit files; list directory contents.
2. `OutlineWriterTool`: Writes a structured survey outline.
3. `SurveyWriterTool`: Queries the built-in paper database and compiles a survey report.
4. `RelatedWorkWriterTool`: Drafts the *Related Work* section with `bibtex` entries.
5. `LiteratureSearchTool`: Searches for papers beyond the internal database (e.g., via Semantic Scholar).
6. `PaperLibrarySearchTool`: Searches the built-in FAISS vector database for semantically relevant knowledge cards.
7. `LiteratureDetailTools`: Retrieves individual paper metadata, citation networks, and reference lists.
8. `ReviewTool`: Evaluates the drafted *Related Work* along four dimensions (format integrity, citation relevance, key reference coverage, instruction following) and returns a scored report.
9. `BibtexFormattingSkill`: Provides instructions for generating correct `bibtex` entries.

**Default Workflow:** (See Fig. 4 in main text)

1. **Outline writing:** The agent identifies the survey topic and invokes `OutlineWriterTool` to produce a structured outline.
2. **Survey composition:** It calls `LiteratureSearchTool` to locate additional papers, then uses `SurveyWriterTool` to generate a comprehensive survey report drawing from both the internal database and newly retrieved papers.
3. **Related work drafting:** The agent activates `BibtexFormattingSkill` and calls `RelatedWorkWriterTool` to draft the *Related Work* section with correct citations.
4. **Quality review:** The agent invokes `ReviewTool` to automatically evaluate the draft along four dimensions. If the review score falls below the acceptance threshold, it iteratively revises and re-reviews until quality requirements are met.

## A.2. Algorithm Agent

**Tools:**

1. `BasicTools`
2. `KnowledgeCardSearch`: Searches knowledge cards for relevant algorithmic structures.
3. `LatexWriterAgent`: Sub-agent for iterative pseudocode and report generation in LaTeX.
4. `TerminalCommandTool`: Executes shell commands.
5. `SmartPlotTool`: Generates diagnostic figures (e.g., loss curves) from logs.
6. `ImageReaderTool`: Verifies visual results via a multimodal LLM.

**Default Workflow:** (See Fig. 5 in main text)

1. **Initialization:** Set up coding environment, identify core algorithmic idea, and prepare workspace.
2. **Knowledge search:** Invoke `KnowledgeCardSearch` to locate prior work and extract critical structural components.
3. **Pseudocode generation:** Design algorithm and leverage `LatexWriterAgent` to produce formal pseudocode.
4. **Python implementation:** Write algorithm and test scripts, execute them, and visualize dynamics via `SmartPlotTool`.
5. **Result verification:** Check experimental outputs against expected behavior using `ImageReaderTool`. If validation fails, loop back to refine.
6. **Finalization:** Compile a final report summarizing algorithmic idea, pseudocode, implementation, and results.

### A.3. Prover Agent

**Tools:**

1. `BasicTools`
2. `RefDownloaderSkill`: Instructs the agent on downloading LaTeX source from arXiv.
3. `MathContentSearch`: Locates relevant mathematical content across references.
4. `CardGeneratorSkill`: Extracts proof techniques and summarizes them into knowledge cards.
5. `KnowledgeSearchTool`: Searches for related knowledge cards.
6. `CardRetrieverSkill`: Instructs the agent on iterative card retrieval.
7. `LemmaProverSkill`: Manages sub-agent sessions for lemma proving.
8. `LemmaProverAgent`: Sub-agent that generates lemma proofs.
9. `LemmaVerifierAgent`: Sub-agent that checks correctness of drafted proofs.

**Default Workflow:** (See Fig. 6 in main text)

1. **Initialization:** Check workspace status, record problem details, and prepare necessary files.
2. **Reference preparation:** Use `RefDownloaderSkill` to obtain materials and `MathContentSearch` to locate relevant sections.
3. **Knowledge extraction (optional):** Create customized knowledge cards via `CardGeneratorSkill`, summarizing lemmas, assumptions, and proof strategies.
4. **Proof planning:** Search for critical cards with `KnowledgeSearchTool` and `CardRetrieverSkill`, and generate a draft proof plan.
5. **Prover stage:** Use `LemmaProverAgent` (guided by `LemmaProverSkill`) to generate draft proofs for individual statements, one at a time.
6. **Verifier stage:** `LemmaVerifierAgent` checks logical soundness, tightness, and completeness. If issues are found, loop back to prover stage.
7. **Finalization:** Generate a comprehensive report with problem statement, assumptions, and all lemmas/theorems/corollaries.

### A.4. Experiment Agent

**Tools:**

1. `BasicTools`
2. `KnowledgeCardSearch`: Retrieves dataset/task recommendations and baseline pseudocode.
3. `TerminalCommandTool`
4. `AutoTuningTool`: Performs automatic hyperparameter optimization.
5. `SmartPlotTool`: Generates publication-quality figures.
6. `PlotAnalysisTool`: Analyzes figures for clarity and anomalies.
7. `LatexWriterAgent`: Assists in composing the final experiment report.

**Default Workflow:** (See Fig. 8 in main text)

1. **Initialization:** Set up coding environment and organize workspace.
2. **Experiment planning:** Design experimental setting (datasets, metrics, baselines, hyperparameters) using `KnowledgeCardSearch`.
3. **Dataset preparation:** Download required datasets via `TerminalCommandTool`.
4. **Coding:** Write runnable experiment scripts.
5. **Experiment conduction:** Execute scripts, tune hyperparameters with `AutoTuningTool`, and generate figures with `SmartPlotTool`.
6. **Result verification:** Check results with `PlotAnalysisTool`; loop back if needed.
7. **Finalization:** Compile a comprehensive report using `LatexWriterAgent`.

### A.5. Introduction Agent

**Tools:**

1. `BasicTools`
2. `ExtractionTools`: `ExtractSurveyInfo`, `ExtractMethodInfo`, `ExtractExperimentInfo`, `ExtractTheoryInfo`, and `OrganizeExtractedInfo` that generate structured section summaries.
3. `IntroductionWriterTool`: Writes the *Introduction* section in LaTeX with inline hallucination annotations. Supports three writing styles (ML, Math, Default).
4. `LiteratureSearchTool`: Tools including `LiteratureGetPaper`, `LiteratureGetCitations` and `LiteratureGetReferences` that retrieve information via Google Scholar and Semantic Scholar.
5. `BibtexFormattingSkill`
6. `EvalIntroduction`: Evaluates the generated introduction on coverage, faithfulness and content quality.
7. `RefineIntroduction`: Performs targeted local refinement based on the evaluation report.

**Default Workflow:** (See Fig. 9 in main text)

1. **Initialization & extraction:** Check workspace, read progress documents, and invoke extraction tools to produce structured summaries.

2. **Information integration:** Call `OrganizeExtractedInfo` to synthesize key contributions, research gaps, and logical flow.

3. **Introduction writing:** Select writing style, invoke `IntroductionWriterTool` to generate LaTeX draft (with hallucination annotations), and optionally supplement references via `LiteratureSearchTool` and `BibtexFormattingSkill`.

4. **Evaluation:** `EvalIntroduction` assesses coverage, faithfulness, and content quality, producing a structured report with per-dimension scores.

5. **Refinement:** `RefineIntroduction` performs targeted local edits on low-scoring dimensions; the evaluation–refinement cycle can be iterated.

### A.6. Writing Agent

The `WritingAgent` uses a two-tier Coordinator + SubWriter architecture.

**Coordinator Tools:**

1. `BasicTools`

2. `TerminalCommandTool`

3. `LatexCompileTool`: Compiles the LaTeX project and returns structured logs.

4. `ScreenshotAnalysisTool`: Renders PDF pages and inspects them via a vision LLM.

5. `DispatchTools` (`DispatchTask`, `wait_for_next`): Asynchronously launches SubWriter instances and collects their results.

6. `ReviewCompletenessTool`: Runs 4 parallel LLM checks (structure, reference verification, visual layout, compilation) + asset citation coverage, returns a fix-task list.

7. `ReviewWritingQualityTool`: Multi-model parallel writing review + meta-review aggregation with venue-aware criteria.

**SubWriter Tools:**

1. `BasicTools`, `TerminalCommandTool`, `LatexCompileTool`, `ScreenshotAnalysisTool` (same as Coordinator)

2. `GenerateMethodFigureTool`: Multi-agent image pipeline (Retriever → Planner → Stylist → Visualizer → Critic) for structural diagrams.

**Default Workflow:** (See Fig. 10 in main text)

1. **Initialization:** Coordinator explores workspace, reads asset inventory, and copies venue-specific LaTeX template suite.

2. **Parallel chapter writing:** Coordinator dispatches tasks to SubWriters; SubWriters independently collect materials, write, compile, and visually verify chapters; Coordinator incrementally adds cross-chapter coherence.

3. **Integration & compilation:** Coordinator integrates all chapters, merges bibliographies, and compiles, iteratively fixing errors/warnings.

4. **Review & revision:** Invoke `ReviewCompleteness` and address issues, then `ReviewWritingQuality` for multi-reviewer feedback. Perform final page-level visual inspection. Repeat until publication-ready.

# B. Experimental Details

## B.1. Experimental Details for Sec. 4.2

### B.1.1. Generation Details

In this subsection, we clarify the prompt used to generate the papers for comparison, which is shown in Fig. 11.

**Prompt for Automatic Paper Generation**

FedAvg is a classic distributed optimization algorithm in federated learning. Recently, subspace optimization methods such as GoLore have sparked a wave of research on memory-efficient large-scale distributed algorithms, which reduce memory usage by projecting gradients onto a low-rank subspace, allowing optimizer states (e.g., the first- and second-order moments in Adam) to be stored in a low-rank memory-saving version. Consider designing the algorithm SFedAvg-GoLore, which introduces the subspace technique from GoLore into FedAvg. It is required to consider only local momentum, without global momentum, and to switch the subspace and reset the momentum to zero after each communication round. Please conduct academic research and writing focusing on SFedAvg-GoLore.

References:
− <https://arxiv.org/abs/2101.11203>: Proof of FedAvg
− <https://arxiv.org/abs/2410.11289>: Convergence of subspace algorithms

*Figure 11.* Prompt for automatic paper generation in Sec. 4.2.

### B.1.2. Evaluation Details

In this subsection, we provide detailed evaluation metrics and results for evaluating the generated papers in Sec. 4.2. As discussed, we design nine critical scholarly aspects. For each aspect, we design finer dimensions for assessment. Below we elaborate the detailed prompts for each aspect.

**A. Mathematical rigor.** For mathematical rigor, we consider five dimensions including: (i) academic formalism and formatting (10 points), (ii) target alignment and relevance (10 points), (iii) consistency of assumptions and notation (10 points), (iv) logical deduction and correctness (50 points), and (v) theoretical significance and strength (20 points). The prompt is as listed in Fig. 12. Corresponding evaluation results are listed in Table 8.

**System prompt for evaluating *Mathematical rigor*.**

You are the **Mathematical Rigor Reviewer**. Evaluate proofs on five dimensions (total 100 pts). Any fatal logical gap can invalidate the proof.
Dimension rubric

- Dim1 Academic Formalism & Formatting (10): Standard proof structure (Lemma/Theorem/Proof), proper LaTeX/math. If wall-of-text with no math or no proof framework, give 0 here and strongly penalize total.
- Dim2 Target Alignment & Relevance (10): Conclusions, loss functions, and update steps align exactly with the stated problem/setting. If irrelevant (e.g., proving SGD when FL requested, assuming convexity in non-convex setting), set this to 0 and cap total $\leq 30$.
- Dim3 Consistency of Assumptions & Notation (10): Only provided assumptions used; notation defined and consistent. Each undeclared strong assumption deduct 5; undefined/inconsistent symbols deduct 1-3 each.
- Dim4 Logical Deduction & Correctness (50, core): Line-by-line check.

- Trivial skips: standard algebra/inequalities $\rightarrow$ no deduction.
- Suspicious skips: non-trivial jumps or "it is easy to see/similarly" without steps $\rightarrow$ deduct 5-10 each; if many, Dim4 $\leq 20$.
- Explicit errors/fake proofs: wrong inequality direction, illegal expectation moves, dimension mismatch, fabricated steps $\rightarrow$ any single one sets Dim4=0 and total $\leq 40$.

- Dim5 Theoretical Significance & Strength (20): Strongest conclusions with weakest assumptions. Reward tight rates/general cases; penalize "trivializing" extra assumptions that make the problem easy.

Scoring rules

- Use 0-100 mapped by weights above (10/10/10/50/20). Apply caps noted in Dim2/Dim4 explicit errors.
- Be evidence-based; cite specific formulas/steps from the provided text only.

Output

- Return **JSON only** per the given schema (sub_scores + summary). Do not add fields.

*Figure 12.* System prompt for evaluating mathematical rigor.

*Table 8.* Detailed Scores for **mathematical rigor**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Dim 4 Score | | | Dim 5 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 70 | 50 | 10 | 90 | 75 | 80 | 82 | 45 | 70 | 60 | 15 | 0 | 65 | 40 | 20 | 67.2 | 32.5 | 20 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 90 | 88 | 50 | 90 | 82 | 90 | **92** | 76 | 95 | 85 | 28 | 15 | 78 | 45 | 40 | 85.3 | 47.6 | 39 |
| Gemini (`Gemini-3.1-Pro`) | 70 | 35 | 70 | 85 | 70 | 80 | 80 | 35 | 80 | 60 | 10 | 0 | 65 | 35 | 0 | 66.5 | 26 | 16 |
| Claude Pro (`Claude Opus 4.6`) | **95** | 78 | 95 | **95** | 72 | **95** | 82 | 20 | 95 | **88** | 0 | 90 | **90** | 25 | **90** | **89.2** | 47.6 | 91.5 |
| `CycleResearcher-ML-12B` | 50 | 45 | 20 | 70 | 20 | 10 | 30 | 10 | 0 | 20 | 0 | 0 | 40 | 15 | 0 | 33 | 10.5 | 3 |
| DeepScientist (`GPT-5.4`) | 85 | 86 | 0 | 90 | **88** | 80 | 85 | 78 | 80 | 80 | 42 | 0 | 70 | 22 | 0 | 80 | 50.6 | 16 |
| AI Scientist-v2 (`GPT-5.4`) | 60 | 15 | 0 | 85 | 55 | 60 | 75 | 45 | 30 | 40 | 5 | 0 | 60 | 10 | 0 | 54 | 16 | 9 |
| ARIS (`GPT-5.4`) | 75 | 85 | 20 | 90 | 82 | 90 | 80 | 78 | 90 | 55 | 35 | 0 | 70 | 55 | 10 | 66 | 53 | 22 |
| ReasFlow (`GPT-5.4`) | 92 | **90** | **100** | 90 | **88** | **95** | 88 | **82** | **100** | 85 | **76** | **95** | 80 | **68** | 85 | 85.5 | **77.6** | **94** |

**B. Originality and significance.** For originality and significance, we consider three dimensions including: (i) method novelty (40 points), (ii) problem importance (30 points), and (iii) differentiation (30 points). The prompt is as listed in Fig. 13. Corresponding evaluation results are listed in Table 9.

**System prompt for evaluating *Originality and significance*.**

You are the **Originality & Significance Reviewer**.
Dimension rubric
- Dim1 Method Novelty (40): substantive technical innovation vs trivial tweak; clarity of the new idea.
- Dim2 Problem Importance (30): relevance and impact of the problem setting in math/application.
- Dim3 Differentiation (30): explicit, evidence-backed comparison to closest prior work; stated improvements.

Evaluation steps

1. Extract the core idea/algorithm; enumerate what is genuinely new.
2. Align against closest prior art listed; check material differences and evidence.
3. Judge importance of the problem/benchmark/context and stated impact.

Scoring rules

- Inflated novelty with no evidence → heavy deduct Dim1.
- Weak/contrived problem → deduct Dim2.
- Missing/vague comparison → deduct Dim3 (cap total at 40 if over-claiming).
- Evidence only from provided text.

Output

- JSON only with the provided schema.

*Figure 13.* System prompt for evaluating originality and significance.

**C. Numerical experiments.** For numerical experiments, we consider five dimensions including: (i) sufficiency (25 points), (ii) baseline fairness (25 points), (iii) parameter specification (15 points), (iv) reproducibility (20 points), and (v) statistical reliability (15 points). The prompt is as listed in Fig. 14. Corresponding evaluation results are listed in Table 10.

**System prompt for evaluating *Numerical experiments*.**

You are the **Numerical Experiments Reviewer**.
Dimension rubric
- Dim1 Sufficiency (25): coverage of typical + edge cases; multiple difficulty levels.
- Dim2 Baseline Fairness (25): fair comparison vs SOTA/classical baselines with matched settings/metrics.

*Table 9.* Detailed Scores for **originality and significance**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 70 | 45 | 50 | 85 | 75 | 80 | 45 | 30 | 20 | 67 | 49.5 | 50 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 75 | 52 | 45 | 85 | 74 | 80 | 80 | 58 | 75 | 79.5 | 60.4 | 64.5 |
| Gemini (`Gemini-3.1-Pro`) | 65 | 45 | 60 | 85 | 75 | 80 | 55 | 30 | 80 | 68 | 49.5 | 72 |
| Claude Pro (`Claude Opus 4.6`) | **80** | 45 | **85** | **90** | 75 | **90** | 85 | 54 | **90** | **84.5** | 56.7 | **88** |
| `CycleResearcher-ML-12B` | 45 | 20 | 20 | 75 | 70 | 60 | 25 | 10 | 10 | 48 | 32 | 29 |
| DeepScientist (`GPT-5.4`) | 60 | 45 | 10 | 80 | 72 | 70 | 75 | 68 | 80 | 70.5 | 60 | 49 |
| AI Scientist-v2 (`GPT-5.4`) | 75 | 55 | 40 | 80 | 72 | 80 | 65 | 50 | 40 | 73.5 | 58.6 | 52 |
| ARIS (`GPT-5.4`) | 78 | **72** | 60 | 85 | **78** | 80 | 80 | **80** | 80 | 80.7 | **76.2** | 72 |
| ReasFlow (`GPT-5.4`) | 78 | 58 | **85** | 85 | 76 | **90** | **90** | 72 | **90** | 83.7 | 67.6 | **88** |

- Dim3 Parameter Specification (15): all key params (grid/step/tol/seeds) stated.
- Dim4 Reproducibility (20): rerun detail-code/links/splits/seeds/ablations.
- Dim5 Statistical Reliability (15): mean/std over multiple runs when stochastic; variance reported.

Evaluation steps

1. List tasks/datasets/cases; judge diversity and relevance.
2. Inspect baselines and fairness (same splits, metrics, tuning budgets).
3. Collect hyper/solver params; note missing ones.
4. Check for code/links/seeds/ablations; assess rerun feasibility.
5. For stochastic settings, verify repeats and variance reporting.

Scoring rules

- Single easy case or missing baselines → large deduct Dim1/Dim2.
- Missing key params/seeds → deduct Dim3/Dim4.
- No repeats in stochastic setting → deduct Dim5 (or 70 "not applicable" if deterministic).
- Fabricated-looking tables or impossible gains → cap total at 30 pending credibility.

Output

- JSON only with given schema.

*Figure 14.* System prompt for evaluating numerical experiments.

*Table 10.* Detailed Scores for **numerical experiments**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Dim 4 Score | | | Dim 5 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 5 | 10 | 0 | 0 | 5 | 0 | 30 | 10 | 0 | 10 | 10 | 0 | 20 | 5 | 0 | 10.8 | 8 | 0 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 5 | 5 | 0 | 0 | 0 | 0 | 30 | 10 | 0 | 20 | 5 | 0 | 10 | 0 | 0 | 11.3 | 3.8 | 0 |
| Gemini (`Gemini-3.1-Pro`) | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 20 | 5 | 0 | 0 | 0 | 0 | 5.3 | 2.5 | 0 |
| Claude Pro (`Claude Opus 4.6`) | 5 | 0 | 0 | 0 | 0 | 0 | 25 | 5 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 11 | 0.8 | 0 |
| `CycleResearcher-ML-12B` | 15 | 15 | 0 | 20 | 10 | 0 | 20 | 10 | 0 | 15 | 5 | 0 | 10 | 0 | 0 | 16.3 | 8.8 | 0 |
| DeepScientist (`GPT-5.4`) | 55 | 28 | 10 | **80** | **63** | 60 | **70** | **74** | **90** | **60** | **42** | 20 | 30 | 18 | 0 | 60.8 | 45 | 35 |
| AI Scientist-v2 (`GPT-5.4`) | 60 | 30 | 30 | 65 | 45 | 40 | 30 | 15 | 0 | 30 | 10 | 0 | 25 | 10 | 0 | 45.5 | 24.5 | 17.5 |
| ARIS (`GPT-5.4`) | 10 | 15 | 0 | 15 | 25 | 0 | 30 | 30 | 0 | 20 | 15 | 0 | 10 | 10 | 0 | 16.3 | 19 | 0 |
| ReasFlow (`GPT-5.4`) | **70** | **35** | **70** | 72 | 52 | **85** | 65 | 66 | 75 | **60** | 38 | **60** | **80** | **78** | **90** | **69.3** | **51** | **75.5** |

**D. Theory-experiment consistency.** For originality and significance, we consider three dimensions including: (i) convergence order verification (40 points), (ii) error bound verification (35 points), and (iii) discrepancy explanation (25 points). The prompt is as listed in Fig. 15. Corresponding evaluation results are listed in Table 11.

**System prompt for evaluating *Theory-experiment consistency*.**

You are the **Theory-Experiment Consistency Reviewer**.
Dimension rubric
- Dim1 Convergence Order Verification (40): theoretical $O(h^p)$ vs empirical rates.
- Dim2 Error Bound Verification (35): theoretical error bounds observed in results.
- Dim3 Discrepancy Explanation (25): reasonable, specific explanations for mismatches.

Evaluation steps

1. Extract theoretical rates/bounds and required conditions.
2. Check convergence tables/plots for matching slopes; note mismatches.
3. If mismatches, assess explanations for adequacy.

Scoring rules

- Missing empirical verification → deduct Dim1/Dim2.
- Unexplained discrepancies → heavy deduct Dim3; may cap total at 40.
- If no theory, but empirical convergence shown, score Dim1/Dim2 on evidence and note absence of theory.

Output

- JSON only per schema.

*Figure 15.* System prompt for evaluating theory-experiment consistency.

*Table 11.* Detailed Scores for **theory-experiment consistency**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 10 | 5 | 0 | 10 | 5 | 0 | 40 | 10 | 0 | 17.5 | 6.3 | 0 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 10 | 5 | 0 | 10 | 5 | 0 | 15 | 10 | 0 | 11.3 | 6.3 | 0 |
| Gemini (`Gemini-3.1-Pro`) | 5 | 0 | 0 | 0 | 0 | 0 | 40 | 5 | 0 | 12 | 1.3 | 0 |
| Claude Pro (`Claude Opus 4.6`) | 10 | 0 | 0 | 10 | 0 | 0 | 40 | 9 | 0 | 17.5 | 2.3 | 0 |
| `CycleResearcher-ML-12B` | 15 | 5 | 0 | 10 | 5 | 0 | 20 | 0 | 0 | 14.5 | 3.8 | 0 |
| DeepScientist (`GPT-5.4`) | 20 | 15 | 0 | 20 | 10 | 0 | 60 | **62** | 0 | 30 | 25 | 0 |
| AI Scientist-v2 (`GPT-5.4`) | 30 | 5 | 0 | 25 | 5 | 0 | 50 | 10 | 0 | 33.3 | 6.3 | 0 |
| ARIS (`GPT-5.4`) | 15 | 10 | 0 | 15 | 10 | 0 | 40 | 15 | 0 | 21.3 | 11.3 | 0 |
| ReasFlow (`GPT-5.4`) | **50** | **18** | **70** | **50** | **15** | **60** | **85** | 58 | **90** | **58.8** | **27** | **71.5** |

**E. Writing and presentation.** For numerical experiments, we consider five dimensions including: (i) structure completeness (20 points), (ii) abstract quality (15 points), (iii) notation consistency (25 points), (iv) language quality (20 points), and (v) definitions and notation (20 points). The prompt is as listed in Fig. 16. Corresponding evaluation results are listed in Table 12.

**System prompt for evaluating *Writing and presentation*.**

You are the **Writing & Presentation Reviewer**.
Dimension rubric
- Dim1 Structure Completeness (20): standard flow Intro → Method → Analysis → Experiments → Conclusion.
- Dim2 Abstract Quality (15): accurate summary of problem, method, main results.
- Dim3 Notation Consistency (25): symbols defined before use; consistent.
- Dim4 Language Quality (20): grammar/spelling; clarity.
- Dim5 Definitions & Notation (20): key terms/notation clearly defined before first use.

Evaluation steps

1. Check abstract and section flow; note missing parts.
2. Track definitions/notation; spot inconsistencies/overloading.

3. Note language issues that impede understanding.

Scoring rules

- Missing sections/undefined notation → deduct Dim1/Dim3/Dim5.
- Ambiguous or overloaded symbols → deduct Dim3/Dim5.
- Poor grammar hurting clarity → deduct Dim4.

Output

- JSON only per schema.

*Figure 16.* System prompt for evaluating writing and presentation.

*Table 12.* Detailed Scores for **writing and presentation**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Dim 4 Score | | | Dim 5 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 75 | 65 | 40 | 80 | 80 | 85 | 80 | 50 | 80 | 85 | 80 | 90 | 80 | 55 | 70 | 80 | 64.5 | 72.8 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 85 | 88 | 40 | 90 | 84 | 85 | **90** | 76 | 95 | 92 | 86 | 95 | **90** | 78 | 90 | 89.4 | 82 | 80.5 |
| Gemini (`Gemini-3.1-Pro`) | 75 | 70 | 40 | 80 | 75 | 80 | 80 | 60 | 80 | 85 | 85 | 90 | 80 | 55 | 90 | 80 | 68.3 | 76 |
| Claude Pro (`Claude Opus 4.6`) | **95** | 90 | 60 | 90 | 82 | 95 | 80 | 28 | 95 | 92 | 78 | 95 | 88 | 42 | 95 | 88.5 | 61.3 | 88 |
| `CycleResearcher-ML-12B` | 65 | 30 | 75 | 60 | 45 | 60 | 35 | 15 | 10 | 75 | 55 | 50 | 35 | 15 | 10 | 52.8 | 39.5 | 29.5 |
| DeepScientist (`GPT-5.4`) | **95** | 90 | 60 | 90 | **90** | 95 | **90** | **82** | 95 | 95 | 88 | 95 | **90** | 80 | 95 | **92** | 85.6 | 88 |
| AI Scientist-v2 (`GPT-5.4`) | 85 | 55 | 50 | 80 | 65 | 40 | 80 | 45 | 30 | 85 | 70 | 70 | 75 | 40 | 20 | 81 | 54 | 41.5 |
| ARIS (`GPT-5.4`) | 90 | **92** | 70 | **92** | **90** | 90 | 85 | 78 | 90 | 92 | **90** | 90 | 85 | **82** | 90 | 88.5 | **85.8** | 86 |
| ReasFlow (`GPT-5.4`) | **95** | 90 | **100** | **92** | 88 | **100** | 88 | **82** | **100** | **96** | 84 | **100** | **90** | **82** | **100** | **92** | 84.9 | **100** |

**F. References.** For originality and significance, we consider three dimensions including: (i) recency (30 points), (ii) key reference coverage (45 points), and (iii) citation format (25 points). The prompt is as listed in Fig. 17. Corresponding evaluation results are listed in Table 13.

**System prompt for evaluating *References*.**

You are the **References Reviewer**.
Dimension rubric

- Dim1 Recency (30): fraction of refs in last 3 years (target ≥30%).
- Dim2 Key Reference Coverage (45): foundational/closest works cited.
- Dim3 Citation Format (25): consistent formatting; DOIs where available.

Evaluation steps

1. Estimate year distribution; compute recent fraction.
2. Check coverage of foundational/closest works mentioned.
3. Inspect formatting consistency and DOIs.

Scoring rules

- Recent fraction <15% → strong deduct Dim1; 15-30% moderate deduct.
- Missing obvious key works → deduct Dim2; multiple absences may cap total at 40.
- Messy formatting/absent DOIs → deduct Dim3.

Output

- JSON only per schema.

*Figure 17.* System prompt for evaluating references.

**G. Internal consistency.** For originality and significance, we consider three dimensions including: (i) numerical consistency (30 points), (ii) symbol consistency (30 points), and (iii) claim-evidence consistency (40 points). The prompt is as listed in Fig. 18. Corresponding evaluation results are listed in Table 14.

*Table 13.* Detailed Scores for **references**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 20 | 0 | 0 | 25 | 10 | 0 | 20 | 0 | 0 | 22.3 | 4.5 | 0 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 85 | 82 | 90 | 80 | 55 | 40 | 80 | 72 | 70 | 81.5 | 67.4 | 62.5 |
| Gemini (`Gemini-3.1-Pro`) | 90 | 70 | **100** | 40 | 20 | 40 | 80 | 60 | 80 | 65 | 45 | 68 |
| Claude Pro (`Claude Opus 4.6`) | 90 | **84** | 90 | 90 | 72 | 95 | 80 | 72 | 85 | 87.5 | 75.6 | 91 |
| `CycleResearcher-ML-12B` | 10 | 0 | 0 | 5 | 0 | 0 | 5 | 0 | 0 | 6.5 | 0 | 0 |
| DeepScientist (`GPT-5.4`) | 90 | 82 | **100** | 85 | 68 | 90 | 85 | 72 | 70 | 86.5 | 73.2 | 88 |
| AI Scientist-v2 (`GPT-5.4`) | 85 | 75 | **100** | 80 | 45 | 80 | 80 | 60 | 50 | 81.5 | 57.8 | 78.5 |
| ARIS (`GPT-5.4`) | 85 | 72 | 70 | 88 | **82** | **90** | 90 | **78** | **90** | 87.6 | **78** | 84 |
| ReasFlow (`GPT-5.4`) | **95** | 74 | **100** | **92** | 76 | **100** | 88 | 44 | **90** | **91.9** | 67.4 | **97.5** |

**System prompt for evaluating *Internal consistency*.**

You are the **Internal Consistency Reviewer**.
Dimension rubric
- Dim1 Numerical Consistency (30): numbers match across abstract, main text, tables.
- Dim2 Symbol Consistency (30): variables use the same symbols throughout.
- Dim3 Claim-Evidence Consistency (40): every claim has theorem/proof/experiment support.

Evaluation steps

1. Cross-check key metrics/figures across sections.
2. Track symbol definitions vs later usage.
3. For each major claim, locate supporting evidence; flag missing links.

Scoring rules

- Conflicting numbers/symbols → heavy deduction Dim1/Dim2.
- Unsupported claims → deduct Dim3; repeated issues may cap total at 40.

Output

- JSON only per schema.

*Figure 18.* System prompt for evaluating internal consistency.

*Table 14.* Detailed Scores for **internal consistency**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | **90** | 25 | 80 | 85 | 45 | 90 | 60 | 20 | 20 | 76.5 | 29 | 59 |
| ChatGPT Pro (`GPT-5.4-Pro`) | **90** | 70 | 85 | 92 | 78 | 95 | 88 | 60 | 70 | 89.8 | 68.4 | 82 |
| Gemini (`Gemini-3.1-Pro`) | 85 | 65 | 60 | 85 | 60 | 90 | 55 | 20 | 25 | 73 | 45.5 | 55 |
| Claude Pro (`Claude Opus 4.6`) | **90** | 55 | 95 | 80 | 20 | 95 | 85 | 33 | 95 | 85 | 35.7 | 95 |
| `CycleResearcher-ML-12B` | 60 | 25 | 20 | 35 | 10 | 10 | 25 | 5 | 0 | 38.5 | 12.5 | 9 |
| DeepScientist (`GPT-5.4`) | **90** | **86** | **100** | **90** | **82** | **100** | 88 | 76 | **100** | 89.2 | **80.8** | **100** |
| AI Scientist-v2 (`GPT-5.4`) | 75 | 55 | 80 | 85 | 35 | 60 | 60 | 40 | 10 | 72 | 43 | 46 |
| ARIS (`GPT-5.4`) | 85 | 78 | **100** | 80 | 80 | 95 | 88 | 80 | 0 | 84.7 | 79.4 | 58.5 |
| ReasFlow (`GPT-5.4`) | **90** | 78 | **100** | 88 | 80 | **100** | **92** | **82** | **100** | **90.2** | 80.2 | **100** |

**H. Visual quality.** For numerical experiments, we consider five dimensions including: (i) figure quality (25 points), (ii)

equation typesetting (20 points), (iii) page layout (20 points), (iv) captions (15 points), and (v) grayscale readability (20 points). The prompt is as listed in Fig. 19. Corresponding evaluation results are listed in Table 15.

**System prompt for evaluating *Visual quality*.**

You are the **Visual Quality Reviewer**.
Dimension rubric
- Dim1 Figure Quality (25): resolution, readable fonts, labeled axes with units.
- Dim2 Equation Typesetting (20): correct rendering, no overflow/broken symbols.
- Dim3 Page Layout (20): consistent margins/spacing/fonts per template.
- Dim4 Captions (15): numbered captions for all figures/tables.
- Dim5 Grayscale Readability (20): distinguishable when printed in B/W.

Evaluation steps

1. Scan pages for figures/tables; check captions and readability.
2. Inspect equations for rendering issues.
3. Note layout consistency and grayscale legibility.

Scoring rules

- Blurry/uncaptioned figures → deduct Dim1/Dim4.
- Broken equations → deduct Dim2.
- Layout issues → deduct Dim3; poor grayscale → deduct Dim5.

Output

- JSON only per schema.

*Figure 19.* System prompt for evaluating visual quality.

*Table 15.* Detailed Scores for **visual quality**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Dim 4 Score | | | Dim 5 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 80 | 35 | 0 | 90 | 75 | 90 | 85 | 80 | 85 | 85 | 60 | 80 | 90 | **85** | **100** | 85.8 | 65.8 | 67 |
| ChatGPT Pro (`GPT-5.4-Pro`) | 80 | 60 | 0 | 92 | **85** | 95 | 90 | 82 | 90 | 85 | 75 | 80 | **95** | 70 | **100** | 88.2 | 73.7 | 69 |
| Gemini (`Gemini-3.1-Pro`) | 60 | 20 | 0 | 85 | 75 | 90 | 85 | 70 | 90 | 60 | 10 | 0 | 90 | 50 | **100** | 76 | 45.5 | 56 |
| Claude Pro (`Claude Opus 4.6`) | 75 | 55 | 0 | 90 | 80 | 95 | 85 | 82 | 90 | 85 | 68 | 80 | 80 | **85** | **100** | 82.5 | 73.4 | 69 |
| `CycleResearcher-ML-12B` | 50 | 25 | 0 | 70 | 40 | 20 | 75 | 65 | 50 | 70 | 50 | 10 | 80 | 50 | 80 | 68 | 44.8 | 31.5 |
| DeepScientist (`GPT-5.4`) | 80 | 50 | 0 | **95** | 82 | 95 | 90 | **84** | 95 | 85 | 78 | 40 | 85 | 80 | **100** | 86.8 | 73.4 | 64 |
| AI Scientist-v2 (`GPT-5.4`) | 80 | 65 | **90** | **90** | 55 | 90 | 85 | 75 | 90 | **90** | 80 | 90 | 70 | 40 | 70 | 82.5 | 62.3 | 86 |
| ARIS (`GPT-5.4`) | 70 | 35 | 0 | 90 | 84 | 90 | 85 | 82 | 90 | 80 | 72 | 0 | 80 | 80 | **100** | 80.5 | 68.8 | 56 |
| ReasFlow (`GPT-5.4`) | **90** | **76** | 85 | **95** | 82 | **100** | **92** | 80 | **100** | **90** | **84** | **100** | 85 | 52 | 85 | **90.4** | **74.4** | **93.3** |

**I. AI and ethics compliance.** For originality and significance, we consider three dimensions including: (i) GenAI disclosure (30 points), (ii) data authenticity (45 points), and (iii) originality declaration (25 points). The prompt is as listed in Fig. 20. Corresponding evaluation results are listed in Table 16.

**System prompt for evaluating *AI and ethics compliance*.**

You are the **AI & Ethics Compliance Reviewer**.
Dimension rubric
- Dim1 GenAI Disclosure (30): if AI tools used, disclosure present; if no evidence of use, disclosure not required.
- Dim2 Data Authenticity (45): signs of fabricated/manipulated data (too-perfect results, repeated patterns).
- Dim3 Originality Declaration (25): statements about no simultaneous submission / no plagiarism.

Evaluation steps

1. Search for AI/tool acknowledgments; if AI evident but undisclosed → penalize.
2. Inspect experimental results for suspicious perfection; note any red flags.
3. Look for originality/ethics statements; mark absence.

Scoring rules

- No AI use evidence & no disclosure needed → around 80 for Dim1; AI evident but undisclosed → severe deduction (∼20).
- Suspicious data → deduct Dim2; genuine-looking data → around 85.
- Missing originality declaration → deduct Dim3 (common omission ∼50).

Output

- JSON only per schema.

*Figure 20.* System prompt for evaluating AI and ethics compliance.

*Table 16.* Detailed Scores for **AI and ethics compliance**. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. Full scores for each dimension are 100. The final score is the weighted average of all dimensions. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score | | | Dim 2 Score | | | Dim 3 Score | | | Final Score | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | **80** | **80** | **80** | 85 | 60 | 85 | 50 | 40 | 50 | 74.8 | 61 | 74.8 |
| ChatGPT Pro (`GPT-5.4-Pro`) | **80** | **80** | **80** | **90** | 78 | 85 | 50 | 45 | 50 | 77 | 70.4 | 74.8 |
| Gemini (`Gemini-3.1-Pro`) | **80** | **80** | **80** | 85 | 65 | 80 | 50 | 45 | **60** | 74.8 | 64.5 | 75 |
| Claude Pro (`Claude Opus 4.6`) | **80** | **80** | **80** | **90** | 65 | 85 | 50 | 41 | 50 | 77 | 63.5 | 74.8 |
| `CycleResearcher-ML-12B` | **80** | **80** | 0 | 75 | 45 | 0 | 50 | 40 | 0 | 70.3 | 54.3 | 0 |
| DeepScientist (`GPT-5.4`) | **80** | **80** | **80** | **90** | 78 | 85 | 50 | 45 | 50 | 77 | 70.4 | 74.8 |
| AI Scientist-v2 (`GPT-5.4`) | **80** | **80** | 20 | 85 | 55 | 20 | 50 | 35 | 50 | 74.8 | 57.5 | 27.5 |
| ARIS (`GPT-5.4`) | **80** | **80** | **80** | **90** | 72 | 85 | 50 | **50** | 50 | 68.9 | **77** | 74.8 |
| ReasFlow (`GPT-5.4`) | **80** | **80** | **80** | **90** | **84** | **100** | **60** | 40 | 50 | **79.5** | 71.8 | **81.5** |

**Overall score.** The overall score of the evaluated paper is computed via the following weighted average formula:

$$S_{\text{full}} = 0.25 \cdot S_A + 0.15 \cdot S_B + 0.2 \cdot S_C + 0.1 \cdot S_D + 0.1 \cdot S_E + 0.05 \cdot S_F + 0.05 \cdot S_G + 0.05 \cdot S_H + 0.05 \cdot S_I,$$

where $S_A, S_B, \cdots, S_I$ denote the scores of the nine scholarly aspects, respectively.

### B.2. Experimental Details for Sec. 4.3.1

To evaluate the *Related Work* section produced by the `SurveyAgent`, we compare ReasFlow's agent against a base LLM baseline across three survey tasks (He et al. (2026), Zhu et al. (2026), and Zhang et al. (2026)). Each task is generated with four different LLM backbones (`DeepSeek`, `GPT-5.1`, `GPT-5.4`, and `Claude Sonnet 4.6`), and independently scored by three evaluation models for robustness. The evaluation employs two dimensions, each scored on a 1-10 scale:

- **Content Accuracy (Dim1)**: assesses the factual correctness of cited paper descriptions. For each article, 15 citations are sampled and verified against a 339-paper FAISS vector database. Papers with valid arXiv IDs are verified via full-text extraction; others are matched through semantic search. Each citation receives an accuracy level from 0 (fabricated) to 3 (essentially correct).
- **Citation Relevance (Dim2)**: evaluates whether citations align with the survey plan and are topically appropriate. The score combines key reference coverage, per-citation relevance ratings (0–2), citation count bonuses, and off-topic penalties.

#### B.2.1. Generation Details

Each of the three tasks is associated with a `survey_plan.md` file that specifies the research topic, required structure, and a list of key references. This plan is provided verbatim to both ReasFlow and the baseline as the sole input.

**ReasFlow (`SurveyAgent`).** The `SurveyAgent` is invoked through its CLI in fully autonomous mode. The agent reads the survey plan and executes a structured three-step workflow: (i) `SurveyWriteOutline` generates a section outline; (ii)

`SurveyWriteSurvey` compiles a comprehensive survey report by querying both the built-in FAISS paper database (339 knowledge cards) and external sources via the Semantic Scholar API; and (iii) `SurveyWriteRelatedWork` produces the final `related_works.tex` and `references.bib`. The agent has access to the full tool suite described in Appendix A, including semantic paper search, citation network traversal, and an automated review tool that evaluates the draft across four quality dimensions and triggers iterative revision.

**Baseline (Base LLM).** The baseline uses an identical LLM backbone but operates through a minimal agent that lacks the structured survey tools. It receives the same survey plan and is instructed to produce the same output files using only basic literature search tools (`LiteratureSearch`, `LiteratureGetReferences`, `LiteratureGetCitations`) and file-writing capabilities. No structured outline, multi-step survey compilation, or automated review is available; the baseline generates the final output in a single agent pass.

**Variant matrix and orchestration.** Each of the two agents is paired with four LLM backbones, `DeepSeek-v3.2`, `GPT-5.1`, `GPT-5.4`, and `Claude Sonnet 4.6`, yielding $2 \times 4 = 8$ generation variants per task and $8 \times 3 = 24$ total generation runs. Each (task, variant) pair executes in an isolated workspace directory to prevent cross-contamination. The entire pipeline is driven by a batch script that reads a YAML configuration specifying the task list, variant definitions (agent type, LLM backbone, and mode), and evaluation model list. For each (task, variant), the script (i) creates an isolated workspace, (ii) writes the survey plan into the workspace, (iii) updates the agent's model configuration to the target backbone, (iv) invokes the corresponding agent CLI, and (v) copies the resulting `related_works.tex` and `references.bib` to the shared evaluation directory. Generation runs that do not produce valid output files are logged as failures and excluded from evaluation.

### B.2.2. Evaluation Details

The evaluation pipeline proceeds in two phases. First, for **Content Accuracy**, each of the 15 sampled citations undergoes a multi-step verification: (i) if the citation has a valid arXiv ID, the full PDF is downloaded and its first 3 000 characters are extracted as evidence; (ii) otherwise, we query Semantic Scholar with the title and authors, and an LLM disambiguates the best match from the candidate list; (iii) the evaluator LLM receives the paper's actual content alongside the description in the *Related Work* and assigns an accuracy level on a 0–3 scale. The accuracy levels are then mapped to a normalized score: level 3 $\rightarrow$ 1.0, level 2 $\rightarrow$ 0.65, level 1 $\rightarrow$ 0.15, level 0 $\rightarrow$ 0.0, and rescaled to $[1, 10]$.

Second, for **Citation Relevance**, the pipeline (i) checks all key references specified in the survey plan for coverage; (ii) scores each sampled citation's relevance (0–2 scale: strongly relevant, weakly relevant, or irrelevant) against the plan; and (iii) computes the final score via:

$$\text{Dim2} = \min\Big(10,\ \underbrace{r_{\text{cov}} \times 7}_{\text{key coverage}} + \underbrace{r_{\text{rel}} \times 2}_{\text{quality bonus}} + \underbrace{\min(3, n_{\text{cite}}/40 \times 3)}_{\text{richness bonus}} - \underbrace{\max(0, (r_{\text{irr}} - 0.15)/0.85 \times 2.5)}_{\text{off-topic penalty}}\Big),$$

where $r_{\text{cov}}$ is the key reference coverage rate, $r_{\text{rel}}$ is the fraction of strongly relevant citations, $n_{\text{cite}}$ is the total citation count, and $r_{\text{irr}}$ is the fraction of irrelevant citations. Fabricated papers detected in Dim1 are automatically counted as irrelevant.

The complete prompts for the two evaluation dimensions are presented in Fig. 21 and Fig. 22.

**Prompt for content accuracy verification (Dim1)**

You are an academic integrity reviewer. Judge whether the description in the Related Works accurately represents the cited paper.
**Important Note: Multi-Paper Joint Citations**
If the citation appears together with other papers (e.g., cite a, b, c), the description may be a joint summary of the entire group. In this case, only judge whether **this paper supports some reasonable part** of the description. If another paper in the group covers a portion the current paper does not address, do not penalize the current paper for that.
**Source Paper Information**
- Title: [paperTitle]
- Authors: [paperAuthors]
- Year: [paperYear]
- Paper excerpt (first 3000 chars): [paperContent]

**Description in Related Works** [description]
**Task** Check whether the description matches the paper's actual content:

1. Is the paper's main method/contribution accurately described?

2. Are there exaggerations, distortions, or clearly incorrect statements?
3. Are key data/conclusions wrong?
4. For multi-paper citations, only judge if this paper supports some reasonable part.

Output JSON: accuracyLevel (0/1/2/3), hallucinationType (null / "misrepresentation" / "exaggeration" / "wrongMetadata"), details ("explanation or null if accurate").
accuracyLevel rubric:

- 3 = Essentially correct: consistent with paper content, no factual errors
- 2 = Minor deviation: does not affect reader understanding
- 1 = Substantial error in method/contribution: would mislead readers
- 0 = Completely fabricated or entirely irrelevant to this paper

*Figure 21.* Prompt for verifying content accuracy of individual citations in the Related Works (Dim1).

**Prompt for citation relevance scoring (Dim2)**

You are an academic paper reviewer evaluating the relevance of a citation in a Related Works section.
**Survey Plan** [plan]
**Citation Under Evaluation**

- Title: [title]
- Authors: [authors]
- Year: [year]
- Venue: [venue]

**Scoring Rubric (0–2)**
Judge relevance strictly against the Survey Plan:
**2 = Strongly relevant**: satisfies any of the following:

- Explicitly listed in the Plan's "Key References"
- A directly representative work of a core research direction described in the Plan
- A background work that directly motivates this research, clearly mentioned or highly aligned with the Plan

**1 = Weakly relevant**: satisfies any of the following:

- An adjacent technique to the Plan's core direction, with some overlap but not central
- Broader context for the research background; reasonable to mention but not emphasized by the Plan

**0 = Irrelevant**:

- No apparent connection to any core direction in the Plan
- Clearly belongs to a different research area

Output JSON: relevance (0–2), reason ("brief explanation").

*Figure 22.* Prompt for scoring per-citation relevance against the survey plan (Dim2).

The total score is Dim1 + Dim2 (out of 20). Results are summarized in Table 17. To assess evaluation robustness, we employ three independent evaluation models (`GPT-5.1`, `Sonnet 4.6`, and `DeepSeek`) and report per-judger total scores for each generation backbone. The `SurveyAgent` achieves an overall advantage of **+3.26 points** over the base model. The improvement is predominantly driven by Content Accuracy (Dim1: +2.92), as the agent's citations originate from the verified paper library and thus achieve high retrieval hit rates. Across all 36 paired comparisons (4 backbones $\times$ 3 tasks $\times$ 3 evaluation models), the `SurveyAgent` wins in **28 cases (77.8% win rate)**.

### B.3. Experimental Details for Sec. 4.3.2

#### B.3.1. Generation Details

Table 18 lists all 35 evaluated tasks, 20 of which are drawn from MLR-Bench. The generation prompts are presented in Fig. 23 (for ReasFlow) and Fig. 24 (for the baselines). We provide the baselines with more detailed requirements to ensure they can produce comparable results.

*Table 17.* Per-task evaluation of the `SurveyAgent` vs. the base LLM across four generation backbones and three evaluation models. Each cell reports the total score (Content Accuracy + Citation Relevance, out of 20). For each evaluation model, the best ReasFlow score and the best advantage are highlighted in **bold**.

| Task | Backbone | Judger: `GPT-5.1` | | | Judger: `Sonnet 4.6` | | | Judger: `DeepSeek` | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Base | RF | Adv. | Base | RF | Adv. | Base | RF | Adv. | Base | RF | Adv. |
| He et al. (2026) | `DeepSeek` | 11.70 | 18.34 | +6.64 | 14.39 | **18.25** | +3.86 | 11.87 | 17.19 | +5.32 | 12.65 | 17.93 | +5.27 |
| | `GPT-5.1` | 19.27 | **18.83** | −0.44 | 19.32 | 18.08 | −1.24 | 19.00 | 16.44 | −2.56 | 19.20 | 17.78 | −1.41 |
| | `GPT-5.4` | 8.99 | 17.09 | +8.10 | 9.26 | 17.01 | +7.75 | 8.99 | 16.15 | +7.16 | 9.08 | 16.75 | +7.67 |
| | `Sonnet 4.6` | 10.86 | 17.77 | +6.91 | 11.31 | 17.87 | +6.56 | 11.04 | 16.07 | +5.03 | 11.07 | 17.24 | +6.17 |
| | Average | 12.71 | 18.01 | +5.30 | 13.57 | 17.80 | +4.23 | 12.72 | 16.46 | +3.74 | 13.00 | 17.42 | +4.42 |
| Zhu et al. (2026) | `DeepSeek` | 13.63 | 14.56 | +0.93 | 13.04 | 14.75 | +1.71 | 13.18 | 12.79 | −0.39 | 13.28 | 14.03 | +0.75 |
| | `GPT-5.1` | 14.24 | 16.27 | +2.03 | 15.76 | 16.90 | +1.14 | 13.55 | 16.71 | +3.16 | 14.52 | 16.63 | +2.11 |
| | `GPT-5.4` | 13.68 | 15.51 | +1.83 | 13.81 | 17.15 | +3.34 | 12.21 | 15.80 | +3.59 | 13.23 | 16.15 | +2.92 |
| | `Sonnet 4.6` | 14.37 | 15.50 | +1.13 | 13.43 | 15.42 | +1.99 | 14.28 | 13.79 | −0.49 | 14.03 | 14.90 | +0.88 |
| | Average | 13.98 | 15.46 | +1.48 | 14.01 | 16.05 | +2.04 | 13.30 | 14.77 | +1.47 | 13.77 | 15.43 | +1.66 |
| Zhang et al. (2026) | `DeepSeek` | 6.52 | 18.25 | **+11.73** | 7.56 | **18.25** | **+10.69** | 10.33 | **18.59** | **+8.26** | 8.14 | **18.36** | **+10.23** |
| | `GPT-5.1` | 6.00 | 16.70 | +10.70 | 6.00 | 15.86 | +9.86 | 9.00 | 15.54 | +6.54 | 7.00 | 16.03 | +9.03 |
| | `GPT-5.4` | 12.32 | 13.77 | +1.45 | 12.99 | 13.90 | +0.91 | 12.19 | 13.33 | +1.14 | 12.50 | 13.67 | +1.17 |
| | `Sonnet 4.6` | 17.47 | 11.38 | −6.09 | 17.87 | 13.29 | −4.58 | 16.62 | 10.18 | −6.44 | 17.32 | 11.62 | −5.70 |
| | Average | 10.58 | 15.03 | +4.45 | 11.11 | 15.32 | +4.22 | 12.04 | 14.41 | +2.37 | 11.24 | 14.92 | +3.68 |
| Overall Average | | 12.42 | 16.16 | +3.74 | 12.90 | 16.39 | +3.50 | 12.69 | 15.22 | +2.53 | 12.67 | 15.92 | +3.26 |

**Prompt for algorithmic tests (ReasFlow).**

Please design an algorithm based on the following.
Task: {task}
Research Idea: {idea}
Please work in AUTO MODE and complete the full algorithm design without waiting for user confirmation.

*Figure 23.* ReasFlow prompt for algorithmic tests.

**Prompt for algorithmic tests (Baseline).**

You are an expert algorithm designer. Given a research task and a research idea, produce TWO outputs in the current working directory:

1. A complete Python implementation saved as `code/algorithm.py`
2. A detailed algorithm design report saved as `document/algorithm_design.md`

CRITICAL INSTRUCTIONS FOR FILE CREATION:

- Create each file with a SEPARATE `apply_patch` command (one patch per file)
- Create `code/algorithm.py` FIRST, then `document/algorithm_design.md` SECOND
- Keep each patch simple - avoid backticks, special characters, or long heredocs inside the content

The Python implementation should:

- Define a class with "Algorithm" or "Optimizer" in its name
- Implement an `optimize()` method (or equivalent main entry point)
- Implement a `get_convergence_history()` method that returns optimization history
- Be self-contained and runnable with standard ML libraries (numpy, scipy, torch, etc.)
- Include a `if __name__ == "__main__":` block with a toy example demonstrating usage

The algorithm design report MUST include all of the following sections:

1. **Algorithm Overview** - what problem it solves and the core intuition
2. **Key Ideas and Innovations** - what is novel about this approach
3. **Algorithm Steps** - detailed procedural description (numbered steps)
4. **Pseudocode** - in algorithmic/LaTeX-style notation
5. **Complexity Analysis** - time and space complexity with justification
6. **Implementation Notes** - key implementation considerations, dependencies, frameworks
7. **Hyperparameters** - table of hyperparameters with defaults and ranges

8. **Expected Behavior and Limitations** - what the algorithm should achieve and its known limitations

IMPORTANT: Do NOT include experimental design, dataset descriptions, or evaluation metrics. Focus ONLY on the algorithm itself.
The `document/` and `code/` directories already exist in your working directory.

---

Task: {task}
Research Idea: {idea}

---

Please create both files now.

*Figure 24.* Baseline prompt for algorithmic tests.

*Table 18.* Task list for evaluation. Each task specifies a concrete research problem and a reference algorithm design document used to seed the evaluation.

| Source | Task ID | Description |
|---|---|---|
| ICLR 2025 workshops (MLR-Bench) | `iclr2025_mldpr` | Machine Learning for Drug Pharmacology Research |
| | `iclr2025_scsl` | Self-Supervised and Contrastive Learning |
| | `iclr2025_verifai` | Verification and Formal Methods for AI |
| | `iclr2025_bi_align` | Bidirectional Alignment in LLMs |
| | `iclr2025_buildingtrust` | Building Trust in AI Systems |
| | `iclr2025_data_problems` | Data-centric ML Problems |
| | `iclr2025_dl4c` | Deep Learning for Code |
| | `iclr2025_question` | Question Answering and Reasoning |
| | `iclr2025_scope` | Scope and Generalisation in NLP |
| | `iclr2025_wsl` | Weakly Supervised Learning |
| Optimization workshops (MLR-Bench) | `neurips2023_opt` | NeurIPS 2023 optimization workshop |
| | `neurips2024_opt` | NeurIPS 2024 optimization workshop |
| | `neurips2023_otml` | optimization and ML (NeurIPS 2023) |
| | `neurips2023_heavytails` | Heavy-tail phenomena in optimization |
| | `neurips2023_federated_learning` | Federated optimization |
| | `neurips2024_fitml` | Fitting ML models at scale |
| | `icml2023_sods` | Sparse and over-parameterised deep systems |
| | `icml2023_differentiable` | Differentiable programming |
| | `icml2023_fl` | Federated learning (ICML 2023) |
| | `icml2024_hdlearning` | High-dimensional learning |
| Fine-grained optimization (OR-Bench) | `opt_adam_variants` | Adaptive optimisers (Adam variants) |
| | `opt_lora_lowrank` | Low-rank adaptation / LoRA |
| | `opt_quantization_training` | Quantisation-aware training |
| | `opt_sparsification` | Gradient and weight sparsification |
| | `opt_curriculum_data` | Curriculum and data ordering |
| | `opt_multiobjective` | Multi-objective optimization |
| | `opt_zeroth_order` | Zeroth-order and black-box optimization |
| | `opt_memory_efficient` | Memory-efficient training |
| | `opt_second_order` | Second-order methods |
| | `opt_bilevel` | Bilevel optimization |
| | `opt_online_learning` | Online learning and regret minimisation |
| | `opt_diffusion_training` | Diffusion model training optimization |
| | `opt_distributed` | Distributed and asynchronous optimization |
| | `opt_hpo` | Hyperparameter optimization |
| | `opt_mixed_precision` | Mixed-precision training |

### B.3.2. Evaluation Details

We evaluate whether the generated Python code for the algorithm is directly runnable. A first-run success is defined as the script executing without runtime errors on its initial attempt. If it fails, the agent is instructed to retry the task until it either succeeds or the total time limit is reached.

## B.4. Experimental Details for Sec. 4.3.3

### B.4.1. Generation Details

In this subsection, we clarify the prompt used to generate the proof reports for further evaluation and comparison, which is shown in Fig. 25.

**Prompt for Proof Report Generation**

**Algorithm 3** Subspace SCAFFOLD

**Require:** Initial model $x^0$ and control variables $\{c_i^0\}_{i=1}^N$, $c^0$; local learning rate $\eta_l$; global learning rate $\eta_g$; local steps $K$; number of sampled clients $S$; subspace dimension $r$; outer cycles $T$
  **Initialize subspace projector**: obtain $\mathbf{P}_0 \in O(\mathbb{R}^{r\times d})$ by common random seed
  **for** $t = 0, \cdots, T-1$ **do**
    **Decompose model and control**: $x_{proj}^t = P_t x^t$, $x_{res}^t = x^t - P_t^\top x_{proj}^t$, $c_{proj}^t = P_t c^t$, $c_{res}^t = c^t - P_t^\top c_{proj}^t$
    **for** all client $i$ in parallel **do**
      **Client control split**: $c_{i,proj}^t = P_t c_i^t$, $c_{i,res}^t = c_i^t - P_t^\top c_{i,proj}^t$
      Initialize $y_{i,proj}^{t,0} = x_{proj}^t$
      **for** $k = 0, \ldots, K-1$ **do**
        Reconstruct full model: $y_i^{t,k} = P_t^\top y_{i,proj}^{t,k} + x_{res}^t$
        Compute mini-batch gradient: $g_i^{t,k} = \nabla F(y_i^{t,k}; \xi_i^{t,k})$
        Update projected model: $y_{i,proj}^{t,k+1} = y_{i,proj}^{t,k} - \eta_l(P_t g_i^{t,k} - c_{i,proj}^t + c_{proj}^t)$
      **end for**
      Update client control: $c_i^{t+1} = (I - P_t^\top P_t)c_i^t + \frac{1}{K}P_t^\top P_t \sum_{k=0}^{K-1} g_i^{t,k}$
    **end for**
    Update projected model: $x_{proj}^{t+1} = x_{proj}^t - \eta_g \frac{\eta_l K}{N} \sum_i \frac{x_{proj}^t - y_{i,proj}^{t,K}}{\eta_l K}$
    **Backfill to full space**: $x^{t+1} = P_t^\top x_{proj}^{t+1} + x_{res}^t$, $c^{t+1} = \frac{1}{NK}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} P_t^\top P_t g_i^{t,k} + c_{res}^t$
    Update projector: generate new $P_{t+1}$ for next round
  **end for**

Prove the convergence result of this algorithm under stochastic non-convex scenario, with the following assumptions:

**Assumption B.1** ($L$-smoothness)**.** Each client loss $F_i : \mathbb{R}^d \to \mathbb{R}$ satisfies $\|\nabla F_i(x) - \nabla F_i(y)\| \le L\|x - y\|$ for all $x, y$.

**Assumption B.2** (Bounded variance and independence across samples)**.** For stochastic gradients $g_i(x;\xi)$, $\mathbb{E}[g_i(x;\xi) \mid x] = \nabla F_i(x)$ and $\mathbb{E}[\|g_i(x;\xi) - \nabla F_i(x)\|^2] \le \sigma^2$. In addition, all stochastic gradient noises $g_i(x;\xi) - \nabla F_i(x)$ are independent across different clients $i$.

**Assumption B.3** (Shared random subspace)**.** In each outer round $k$, all clients use the same orthonormal $P_k \in O(\mathbb{R}^{r\times d})$, with $P_k P_k^\top = I_r$ and $\mathbb{E}_{P_k}[P_k^\top P_k] = (r/d)I_d$.

Note that you should not introduce bounded gradient assumption or assumptions on data heterogeneity. You should try your best to give the correct proofs and generate a latex file as a final report of the task. The report should be written like the theory section of an academic paper.

*Figure 25.* Prompt for generating proof reports for evaluation.

### B.4.2. Evaluation Details

In this subsection, we provide detailed implementation and results for evaluating the proof reports generated by ReasFlow and baselines. The complete prompt for the `ProofReviewAgent` is as presented in Fig. 26. The detailed assessment including scores for each different dimension is as listed in Table 19.

**System prompt for the `ProofReviewAgent`**

## Role

You are a top-tier machine learning theoretical scientist and a senior reviewer for prestigious academic conferences such as NeurIPS and ICML. You are renowned for your extremely rigorous, objective, and fastidious academic attitude.

## Task

Your task is to evaluate the quality of a theoretical proof generated by an AI. I will provide the [Problem Definition & Settings], [Target Algorithm & Conclusion], and the [AI-Generated Proof Process]. You must meticulously scrutinize every step of the proof and provide a comprehensive score (0-100) along with a detailed review report.

## Evaluation Criteria

Please strictly follow the four dimensions below for review and scoring. Note that mathematical proofs require absolute rigor; any single fatal logical gap may render the entire proof invalid.

### Dimension 1: Academic Formalism & Formatting (10 pts)

- **Full Score Standard**: Possesses the standard proof structure of an academic paper (e.g., clear Lemmas, Theorems, Proof environments, and professional LaTeX formatting).
- **Deductions**: If the response is merely a wall of text lacking mathematical formulas or fails to provide a basic proof framework (e.g., "Proof:", "Q.E.D."), deduct all points for this dimension.

### Dimension 2: Target Alignment & Relevance (10 pts)

- **Full Score Standard**: The proven conclusion, loss functions used, and algorithmic update steps are perfectly consistent with the [Problem Definition & Settings].
- **Deductions**: If the proof conclusion is irrelevant to the requested algorithm or setting (e.g., proving a standalone SGD when Federated Learning was required, or assuming convexity when a non-convex setting was specified), assign 0 points for this dimension.

### Dimension 3: Consistency of Assumptions & Notation (10 pts)

- **Full Score Standard**: Strictly uses only the provided assumptions (e.g., L-smoothness, bounded variance); notations must be consistent throughout the text and clearly defined.
- **Deductions**:
- Introducing undeclared strong assumptions (e.g., "assuming bounded gradients" or "assuming i.i.d. data"): Deduct 5 points per occurrence.
- Inconsistent notation or using undefined variables (e.g., using $x$ in one step and suddenly switching to $w$): Deduct 1-3 points depending on severity.

### Dimension 4: Logical Deduction & Correctness (50 pts) [Core Dimension]

Examine the formula derivations line-by-line. LLMs frequently "hallucinate" in this area. Categorize logical leaps into the following three types:

- **Trivial Skips**: Basic algebraic simplifications, combining like terms, or applying standard inequalities (e.g., Cauchy-Schwarz, Peter-Paul). No deduction.
- **Suspicious Skips**: Consolidating several complex equations into a non-trivial new expression without intermediate steps or cross-term analysis; or claiming "it is easy to see" or "similarly" for steps that are not immediately obvious. Deduct 5-10 points per "Suspicious Skip." If the proof contains excessive suspicious leaps that make verification impossible, the score for this dimension cannot exceed 20 points.
- **Explicit Errors / Fake Proofs**: Incorrect inequality directions, expectation operators penetrating non-linear functions, matrix dimension mismatches, or "fake proofs" intentionally fabricated to reach the target result. Any single "Explicit Error" results in 0 points for this dimension.

### Dimension 5: Theoretical Significance & Strength (20 pts)

- **Criteria**: Evaluate the "Price-Performance Ratio" of the theorem. A high-quality proof should aim for the **strongest possible conclusion** using the **weakest possible assumptions**.
- **Scoring Logic**:
- **Strong Conclusion**: Does it provide a tight convergence rate (e.g., $O(1/\epsilon^2)$ vs $O(1/\epsilon)$)? Does it handle the most general case (e.g., non-convex)?
- **Weak Assumptions**: Does it avoid restrictive assumptions like "Bounded Gradients" or "Strong Convexity"?
- **Penalty**: If the proof achieves the goal by adding "trivializing" assumptions (making the problem too easy), deduct points from the total score.

## Output Format

Please output and save your review results in JSON format to ensure it can be parsed:

```
{
"Review_Process": {
"Step_by_Step_Critique": "Analyze the proof process line-by-line here. Explicitly
↪  identify which steps are normal, which are suspicious skips, and which are
↪  explicit errors. You must include specific formula references.",
"Assumption_Check": "Analyze whether the author abused assumptions or omitted critical
↪  given assumptions.",
"Alignment_Check": "Analyze whether the proof truly addresses the specified problem
↪  and algorithm.",
"Strength_Check": "Analyze the theoretical significance of the provided theorem."
},
"Scores": {
"Dim1_Formalism": [0-10],
"Dim2_Relevance": [0-10],
"Dim3_Consistency": [0-10],
"Dim4_Logical_Rigor": [0-50],
"Dim5_Strength": [0-20]
},
"Total_Score": [0-100],
"Final_Verdict": "Accept / Weak Reject / Strong Reject (A one-sentence summary of
↪  whether the proof is credible)"
}
```

Use the write() tool provided to save the above JSON review to a file if the user provides a file path.

*Figure 26.* System prompt for the `ProofReviewAgent`.

*Table 19.* Detailed Scores of the theoretical reports reviewed by the `ProofReviewAgent`. S1, S2, S3 corresponds to evaluations performed with `GPT-5.1`, `GPT-5.4` and `Gemini-3-Pro` as the reviewer's LLM backend, respectively. **Bold**: best; underlined: second best.

| AI Agent | Dim 1 Score (10) | | | Dim 2 Score (10) | | | Dim 3 Score (10) | | | Dim 4 Score (50) | | | Dim 5 Score (20) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| ChatGPT (`GPT-5.3`) | 8 | 8 | 2 | 4 | 2 | 0 | 5 | 1 | 5 | 0 | 0 | 0 | 12 | 4 | 0 |
| Gemini (`Gemini-3.1-Pro`) | **10** | 9 | **10** | 9 | 6 | **10** | 8 | 3 | **10** | 38 | 0 | 0 | 16 | 6 | 18 |
| ReasLingo (`Gemini-3-Pro`) | **10** | 9 | **10** | 9 | **8** | **10** | 9 | 3 | **10** | 45 | 18 | 0 | **18** | 10 | 10 |
| ReasLingo (`GPT-5.1`) | **10** | 9 | **10** | 9 | 6 | **10** | 9 | 5 | **10** | 42 | 8 | 0 | 17 | 7 | 10 |
| ReasLingo (`GPT-5.4`) | **10** | 9 | **10** | 9 | 5 | 0 | **10** | 5 | **10** | 46 | 32 | 40 | 17 | 6 | 0 |
| ProofGrader (`GPT-5.1`) | **10** | 9 | **10** | 9 | 2 | 0 | **10** | 3 | **10** | 46 | 16 | **50** | 17 | 4 | 0 |
| ProofGrader (`GPT-5.4`) | **10** | 9 | **10** | 9 | 7 | **10** | **10** | 3 | **10** | 46 | 12 | 40 | **18** | 8 | 5 |
| QED (`GPT-5.4`) | **10** | **10** | **10** | **10** | 2 | 0 | **10** | **9** | **10** | **50** | 24 | **50** | **18** | 6 | 0 |
| ReasFlow (`Gemini-3-Pro`) | **10** | 9 | **10** | 9 | **8** | **10** | 9 | 6 | **10** | 45 | 24 | **50** | **18** | **15** | **20** |
| ReasFlow (`GPT-5.1`) | **10** | 9 | **10** | 9 | **8** | **10** | **10** | 5 | **10** | 46 | 16 | **50** | **18** | 7 | **20** |
| ReasFlow (`GPT-5.4`) | **10** | **10** | **10** | 9 | **8** | **10** | **10** | 8 | **10** | 47 | **38** | **50** | **18** | 14 | **20** |

### B.5. Evaluation Details for Sec. 4.3.4

#### B.5.1. Generation Details

We evaluate the agents by having them continue to design and implement experiments based on their outputs from Sec. 4.3.2. The generation prompts are provided in Fig. 27 (for ReasFlow) and Fig. 28 (for the baselines). We provide the baselines with more detailed requirements to ensure they can produce comparable results.

**Prompt for experimental tests (ReasFlow).**

You are given an algorithm design report and implementation script for the following research.
**Task:** {task}
**Research Idea:** {idea}
**Algorithm Design Report (summary):** {`alg_report`[:3000]}
**Algorithm Script Location:** {`alg_script_hint`}
Please work in AUTO MODE:

1. Read the algorithm design report and script carefully
2. Design a comprehensive experimental plan (`experiment_plan.md`)
3. Execute all planned experiments
4. Generate `experiment_report.md` with full results and analysis
5. Create publication-quality figures

Complete all experiments without waiting for user confirmation. Time limit: {N} hours.

*Figure 27.* ReasFlow prompt for experimental tests.

**Prompt for experimental tests (Baseline).**

You are an expert ML researcher. You previously designed an algorithm. Now you must ACTUALLY EXECUTE experiments to evaluate it.
The algorithm design report and implementation code are available in the `document/` and `code/` directories of your current working directory:

- `document/algorithm_design.md` - the algorithm design report
- `code/algorithm.py` - the algorithm implementation

## STEP-BY-STEP INSTRUCTIONS (follow in order):

### Step 1: Install dependencies

Run: `pip install numpy scipy torch matplotlib scikit-learn pandas` Install any additional packages required by `code/algorithm.py`.

### Step 2: Verify the algorithm runs

Run: `cd code && python algorithm.py` If it fails, fix the import errors or missing dependencies, then re-run.

### Step 3: Write experiment scripts

Create `code/run_experiments.py` that:

- Imports and instantiates the algorithm from `algorithm.py`
- Defines 2-3 baseline methods for comparison (e.g., random search, grid search, SGD)
- Runs all methods on 2-3 test problems (e.g., Rosenbrock, Rastrigin, or task-relevant benchmarks)
- Records metrics: convergence speed, final objective value, runtime
- Saves results to `results/` directory as CSV or JSON

### Step 4: Run experiments

Execute: `cd code && python run_experiments.py` Capture all output and results.

### Step 5: Write experiment report

Save `document/experiment_report.md` containing:

1. **Experimental Setup** - datasets/problems, metrics, baselines, hyperparameters used
2. **Implementation Details** - how experiments were set up and run
3. **Main Results** - performance tables with ACTUAL numbers from your runs
4. **Ablation Studies** - effect of key hyperparameters
5. **Analysis and Discussion** - why the method works (or doesn't)

CRITICAL RULES:

- You MUST actually run code and report real numbers. Do NOT fabricate results.
- If a script fails, debug it and fix it before reporting.
- All numerical results must come from actual execution output.
- Create the `results/` directory if needed: `mkdir -p results`

---

Task: {task} Research Idea: {idea}

*Figure 28.* Baseline prompt for experimental tests.

### B.5.2. Evaluation Details

We use both `GPT-5.4` and `Claude Sonnet 4.6` to evaluate the outputs. The prompts provided in Fig. 29 are adapted from MLR-Bench, with metrics we propose marked as `[NEW]`.

**Prompt for evaluating experimental results.**

You will be given an experimental document produced by an AI agent. Unlike the standard MLR-Bench setup, this agent received an algorithm design report and script (not a full research proposal) as input, and was responsible for BOTH autonomously designing the experimental plan AND executing the experiments. Therefore, please evaluate both the quality of the experimental design (which the agent produced autonomously) and the quality of the execution results.
Please evaluate along all dimensions below (1-10 scale) and respond in valid JSON.

---

## EVALUATION DIMENSIONS

**[NEW] DESIGN SOUNDNESS (1-10)**

How rigorous and well-structured is the experimental plan that the agent autonomously designed? Consider: appropriateness of baselines, clarity of evaluation metrics, inclusion of ablation studies, and internal logic of the plan.
9-10 - The experimental design is highly rigorous, with appropriate baselines, evaluation metrics, and ablation studies clearly defined.
7-8 - The design is mostly sound with only minor gaps (e.g., one missing baseline).
5-6 - The design has some weaknesses but covers the main experimental objectives.
3-4 - The design has significant gaps or logical flaws in the experimental plan.
1-2 - The experimental design is absent or fundamentally unsound.

**[NEW] DESIGN COMPLETENESS (1-10)**

Does the experimental plan include all necessary components: baseline comparisons, appropriate evaluation metrics, and ablation studies?
9-10 - All necessary components are present and well-defined (baselines, metrics, ablations, statistical significance considerations).
7-8 - Most components are present with minor omissions (e.g., no ablation study).
5-6 - Some important components are missing (e.g., no baseline or no ablation).
3-4 - Many key components are absent; plan is skeletal.
1-2 - The experimental plan is severely incomplete or nonexistent.

**[NEW] ALGORITHM-EXPERIMENT ALIGNMENT (1-10)**

Does the experimental design fully leverage the core contribution of the provided algorithm script and design report? Is there a clear logical connection between the algorithm and the experimental setup?
9-10 - The experiments are perfectly tailored to demonstrate the algorithm's core contribution; every experiment directly tests a key claim.
7-8 - Strong alignment with only minor disconnects (some tangential experiments).
5-6 - Partial alignment; some experiments are not directly motivated by the algorithm.
3-4 - Weak alignment; experiment and algorithm feel largely independent.
1-2 - No meaningful connection between the algorithm and experimental design.

**SOUNDNESS (1-10) [Most Important]**

Are the experimental results based on REAL execution rather than synthetic/fabricated data? This is the most critical dimension. Check result files (`results/*.csv`, `results/*.json`, `data/*.csv`, `data/*.json`) AND execution logs for evidence of actual runs. Note: some agents output results directly to a `results/` directory rather than logs.
**MANDATORY RULE - apply before any other scoring:** If BOTH of the following are true: (a) `execution_artifacts` shows [NO DATA FILES FOUND] (no CSV/JSON result files exist), AND (b) log shows [NO LOG FOUND] or contains no actual execution traces (no stack traces, no printed values, no timing information - just setup text), THEN: Soundness MUST be scored 1-2, and `has_hallucination` MUST be true. A polished `experiment_report.md` without any execution evidence is fabricated by definition.
9-10 - Results clearly stem from genuine execution with verifiable logs and outputs.
7-8 - Results appear genuine with minor documentation gaps.
5-6 - Mostly genuine but with some suspicious or unverifiable claims.
3-4 - Significant portions appear fabricated or hallucinated.
1-2 - Results are clearly synthetic/fabricated; no genuine experiments were run.

**CONSISTENCY (1-10)**

Does the execution faithfully follow the experimental plan? Are the reported results consistent with the experimental setup described?

9-10 - Execution precisely matches the plan; all reported numbers are traceable.
7-8 - Mostly consistent with minor deviations (well-explained).
5-6 - Some inconsistencies between plan and execution.
3-4 - Significant gaps between stated plan and actual execution.
1-2 - Execution is entirely inconsistent with or ignores the plan.

**COMPLETENESS (1-10)**

Were all designed experiments actually completed and reported?
9-10 - All planned experiments completed; all results reported with full detail.
7-8 - Most experiments completed; minor omissions explained.
5-6 - Roughly half of planned experiments completed.
3-4 - Only a small fraction of planned experiments completed.
1-2 - Almost no experiments completed or reported.

**NOVELTY (1-10)**

How novel and creative is the experimental design in terms of evaluation methodology?
9-10 - Innovative evaluation approach that provides unique insights.
7-8 - Some novel evaluation choices beyond standard benchmarks.
5-6 - Standard experimental design; nothing unusual.
3-4 - Very conventional; misses opportunities for deeper evaluation.
1-2 - Trivial or copied experimental design.

**INSIGHTFULNESS (1-10)**

How deep and thoughtful is the analysis of the experimental results?
9-10 - Rich analysis connecting results to algorithmic mechanisms; identifies failure modes and proposes explanations.
7-8 - Good analysis with meaningful observations.
5-6 - Adequate analysis; mostly describes results without deeper interpretation.
3-4 - Shallow analysis; mostly just reports numbers.
1-2 - No meaningful analysis; results are presented without any interpretation.

**SIGNIFICANCE (1-10)**

Based on the experimental results, how significant is the algorithm's demonstrated contribution to the field?
9-10 - Results demonstrate substantial improvements over baselines with clear statistical significance.
7-8 - Meaningful improvements demonstrated across most metrics/datasets.
5-6 - Modest improvements; results are mixed across conditions.
3-4 - Negligible improvements; algorithm barely outperforms baselines.
1-2 - Algorithm underperforms baselines; results are negative.

**[NEW] EXPERIMENTAL EFFECTIVENESS (1-10)**

Does the experimental evidence actually support or meaningfully test the algorithm's claims? This dimension evaluates the Claim-to-Evidence loop: whether the experiments produce results that constitute genuine evidence (positive or negative) for the algorithm's stated contributions.
NOTE: Honestly reporting negative results (algorithm underperforms) is NOT penalized if the experimental design was sound and the analysis is insightful. What matters is whether the experiments were designed to TEST the claims and whether the results INFORM the reader about the claims' validity.
9-10 - Every major claim has direct experimental evidence; results clearly confirm or refute each claim with appropriate metrics and baselines; ablation studies isolate individual contributions.
7-8 - Most claims are supported by experimental evidence; one or two claims lack direct evidence; overall the experiments convincingly test the core contribution.
5-6 - Some claims have supporting evidence but others are tested only indirectly or with insufficient baselines; the evidence-claim mapping is partial.
3-4 - Weak connection between claims and evidence; experiments run but results do not clearly inform any specific claim; generic benchmarking without claim focus.
1-2 - No discernible connection between the algorithm's claims and the experimental results; experiments appear arbitrary or disconnected from the contribution.

**[NEW] WRITEUP READABILITY (1-10)**

How well-structured is this experiment report for a downstream writing agent to produce the Experiments section of a research paper? Evaluate whether the report provides all elements needed for paper writing.
Consider:

- Does the report follow a clear structure (Experimental Setup, Baselines, Metrics, Main Results, Ablation Studies, Analysis/Discussion)?
- Are data tables formatted with mean +/- std across runs?
- Are figures generated with proper captions, axis labels, and legends that are paper-ready?
- Is reproducibility information included (hyperparameters, seeds, compute environment)?

- Are result claims explicitly tied to specific table/figure references?
- Does the report include a clear comparison narrative (what beats what and by how much)?
- Would a writing agent be able to extract a complete Experiments + Results section?

9-10 - Excellent: Report is publication-ready; structured sections, formatted tables with mean+/-std, captioned figures, full reproducibility info.
7-8 - Good: Most elements present; minor formatting or structural gaps.
5-6 - Moderate: Basic results present but significant gaps in formatting, tables, or figure quality for paper writing.
3-4 - Weak: Report is mainly unstructured text with inline numbers; writing agent would need major reformatting.
1-2 - Poor: No usable structure; raw output dumps or missing key sections.

---

## HALLUCINATION CHECK

**Step 1 - Hard rule (check first):** If `execution_artifacts` = [NO DATA FILES FOUND] AND log = [NO LOG FOUND] or log contains no execution traces (no output values, no timing, no errors): → set `has_hallucination` = true, Soundness = 1-2. Do NOT let a polished report override this.
**Step 2 - Soft checks (apply when Step 1 does not trigger, i.e., evidence files DO exist):** IMPORTANT: When result files exist, the bar for flagging hallucination is HIGH. Stylistic gaps, narrative imprecision, or partial report coverage are NOT hallucination. Only flag `has_hallucination` = true here if:

- The KEY numerical claims in the report (e.g., "our method achieves 92% accuracy") directly contradict the actual values in the result files (e.g., files show 61%), OR
- The result files contain clearly synthetic/templated data (identical values across all conditions, suspiciously round numbers like exactly 0.5000, 1.0000 everywhere), OR
- The report claims results for datasets/baselines that are entirely absent from all files.

Do NOT flag hallucination merely because:

- The report narrative is incomplete or partially inconsistent with the experimental plan
- Some secondary metrics or ablations in the report lack corresponding files
- The evaluation setup differs from what the task description suggested

---

## INPUT

**Task Description:** {`task`}
**Research Idea:** {`idea`}
**Algorithm Design Report:** {`algorithm_report`}
**Algorithm Script (`code/algorithm.py` summary or key excerpts):** {`algorithm_script`}
**Experiment Report (`document/experiment_report.md`):** {`experiment_report`}
**Execution Log (`logs/` or `log.txt`, truncated if long; may show [NO LOG FOUND] if agent stored outputs in result files instead):** {`log`}
**Execution Artifacts (data files / CSV results / figures generated during experiments):** {`execution_artifacts`}

---

OUTPUT FORMAT
Respond with ONLY valid JSON matching this schema:

```
{
  "Hallucination": {
    "has_hallucination": <true/false>,
    "details": "Evidence of fabrication or confirmation of genuine results"
  },
  "DesignSoundness":              {"score": <1-10>, "justification": "..."},
  "DesignCompleteness":           {"score": <1-10>, "justification": "..."},
  "AlgorithmExperimentAlignment": {"score": <1-10>, "justification": "..."},
  "Soundness":                    {"score": <1-10>, "justification": "..."},
  "Consistency":                  {"score": <1-10>, "justification": "..."},
  "Completeness":                 {"score": <1-10>, "justification": "..."},
  "Novelty":                      {"score": <1-10>, "justification": "..."},
  "Insightfulness":               {"score": <1-10>, "justification": "..."},
  "Significance":                 {"score": <1-10>, "justification": "..."},
  "ExperimentalEffectiveness":    {"score": <1-10>, "justification": "..."},
  "WriteupReadability":           {"score": <1-10>, "justification": "..."},
  "OverallAssessment": {
    "score": <1-10>,
```

```
    "strengths": ["...", "..."],
    "weaknesses": ["...", "..."]
  },
  "Confidence": <1-5>
}
```

*Figure 29.* Prompt for judging experimental results.

### B.6. Evaluation Details for Sec. 4.3.5

To evaluate the quality of the *Introduction* section generated by the `IntroductionAgent`, we compare it against a base model (`GPT-4o`) baseline across three papers. For each configuration, we perform five independent runs and report mean scores. The evaluation employs three complementary metrics, each scored on a 0-1 scale:

- **Content Coverage**: measures how well the introduction captures the reference material. It is a weighted combination of raw weighted coverage (weight 0.34), anchor coverage for main results and theorems (0.18), core scientific path coverage for method and evidence chains (0.22), structure alignment for the problem→gap→method→evidence flow (0.26), and a survey dominance penalty ($-0.04$) that penalizes overly background-heavy introductions.
- **Faithfulness**: quantifies hallucination avoidance, defined as $1 - (\#\text{unsupported claims}/\#\text{total claims})$. Claims are categorized into common background (lenient), paper-specific (strict), and overclaim/speculation (very strict) thresholds.
- **Content Quality**: a rubric-based score assessing six dimensions, domain/application emphasis, algorithm mechanism or formula details, complete background description, core research idea clarity, explicit research goals/contributions, and concrete experimental or theoretical depth.

#### B.6.1. Generation Details

Each paper's upstream agent outputs, including the experiment report, theoretical proof report, survey/related work draft, and algorithm description, serve as the shared reference material from which both ReasFlow and the baseline must compose the introduction.

**ReasFlow (`IntroductionAgent`).** The `IntroductionAgent` first collects structured information from all upstream agent outputs using dedicated extraction tools that parse survey reports, method descriptions, experimental results, and theoretical contributions into structured summaries, grounding all content in source materials rather than relying on the LLM's parametric knowledge. The agent then composes the introduction and enters an automated *evaluation–refinement* loop: a built-in evaluation tool assesses the draft along three quality dimensions (Coverage, Faithfulness, Content Quality), and a refinement tool performs targeted local edits on low-scoring aspects while preserving high-quality content. This loop iterates until scores plateau or a maximum iteration count is reached.

**Baseline (`GPT-4o`).** The baseline receives the same upstream assets, the complete set of `.tex` files from the prover, experiment, survey, and algorithm agents, concatenated as context, together with a single prompt instructing the model to write a self-contained introduction. No structured extraction, automated evaluation, or iterative refinement is performed; the baseline produces the introduction in a single inference call.

**Runs and reproducibility.** For each of the three papers, five independent runs are performed for both ReasFlow and the baseline to account for stochastic variation, and results are averaged.

Before evaluation, two preprocessing steps transform each generated introduction and its reference material into structured representations amenable to automated scoring.

**Reference Preprocessing** (`preprocess_refe.py`): All reference `.tex` files from the paper (experiment reports, theoretical proofs, related work sections) are fed to an LLM, which performs a structured decomposition into three categories, *experiment_results*, *prover_results*, and *related_work_claims*. A second LLM call then classifies each item's importance as `critical` (directly supports the main conclusion), `important` (supports the conclusion but is not the most central), or `supplementary` (minor details not essential for the introduction).

**Prediction Claim Extraction** (`preprocess_pred.py`): The generated introduction together with its bibtex file `references.bib` is parsed by an LLM to extract all explicit contribution and factual statements as minimal semantic

units (*prediction_claims*), each classified as either a "contribution" or "fact" type. These structured representations, the categorized reference items and the extracted prediction claims, serve as inputs to the evaluation stage described below.

### B.6.2. Evaluation Details

The evaluation proceeds in three stages.

**Stage 1 (Reference Preprocessing)**: All reference `.tex` files from the paper are fed to an LLM, which extracts three structured categories, *experiment_results*, *prover_results*, and *related_work_claims*, each annotated with an importance label (`critical`, `important`, or `supplementary`) based on its role in supporting the paper's main conclusions.

**Stage 2 (Claim Extraction)**: The generated introduction is parsed by an LLM to extract all explicit contribution and factual statements as minimal semantic units (*prediction_claims*).

**Stage 3 (Evaluation)**: Two LLM calls are made, one for joint Coverage & Faithfulness assessment, and one for Content Quality, using the structured references and extracted claims as context.

**Coverage scoring.** The final coverage score is a weighted combination of four sub-metrics:

$$\text{Coverage} = 0.34 \cdot C_{\text{raw}} + 0.18 \cdot C_{\text{anchor}} + 0.22 \cdot C_{\text{core}} + 0.26 \cdot C_{\text{struct}} - 0.04 \cdot P_{\text{survey}},$$

where $C_{\text{raw}}$ is the importance-and-category-weighted coverage (critical items weight 1.0, important 0.6, supplementary 0.0; experiment results scaled $\times 1.15$, prover results $\times 1.10$, related work $\times 0.55$); $C_{\text{anchor}}$ is the coverage rate of anchor items (main/core results); $C_{\text{core}}$ weights only experiment and prover results to capture the method-to-evidence chain; $C_{\text{struct}}$ measures alignment with the problem$\rightarrow$gap$\rightarrow$method$\rightarrow$contribution$\rightarrow$ evidence narrative arc; and $P_{\text{survey}}$ penalizes overly background-heavy introductions.

The complete prompts for the two evaluation calls are presented in Fig. 30 and Fig. 31.

**System Prompt for Coverage & Faithfulness Evaluation**

You are a rigorous and meticulous LLM-as-a-judge specialized in evaluating scientific introductions.
**Coverage (0.00–1.00)** The proportion of reference information (experimentResults, proverResults, relatedWorkClaims) meaningfully covered by the introduction.

- Mark an item as **covered** when the introduction captures its substantive meaning, even if compressed, paraphrased, or merged with nearby claims.
- Do NOT require lexical overlap or one-to-one sentence matching.
- For method/theory/result items, a conservative summary counts as covered even if minor details are omitted.
- Reward introductions preserving the chain: problem/background, limitation/gap, method/contribution, evidence preview.

**Faithfulness (0.00–1.00)** Faithfulness = 1 - (unsupportedClaims / totalClaims). Classify each prediction claim before judging:

1. commonBackground: standard field knowledge, faithful unless contradicted;
2. paperSpecific: claims about method, theory, experiments, require direct support or conservative inference;
3. overclaimOrSpeculation: mark unfaithful if contradicted, overstated, numerically unsupported, or unreasonably inferred.

**Output Format (JSON)** Return coverage (float in [0, 1]), faithfulness (float in [0, 1]), rationale, claimFaithfulnessDetails (list of claimIndex, claimText, status), and referenceCoverageDetails (experimentResults, proverResults, relatedWorkClaims with index, descriptionExcerpt, importance, status).

*Figure 30.* System prompt for the joint Coverage & Faithfulness evaluation of generated introductions.

**System Prompt for Content Quality Evaluation**

You are a rigorous LLM-as-a-judge focusing on **Content Quality**. Apply strict standards.
**Content Quality (0.00–1.00)** Score based on the following dimensions (lacking any dimension incurs deductions):

1. **Domain emphasis**: Does the introduction clearly identify the research/application domain?
2. **Algorithm/formula detail**: Does it include key algorithmic steps or core mathematical formulas with correct explanations?
3. **Background completeness**: Is the background coherent and sufficient to set up the subsequent content?
4. **Core idea clarity**: Does it clearly present the paper's core method and innovation?
5. **Research goals**: Are research objectives and contributions explicitly stated and verifiable?
6. **Concrete depth**: Does it include specific experimental settings/results or theoretical conclusions, avoiding vague generalities?

**Bonus criteria** (should significantly raise the score if present):

- Deep discussion of mathematical principles, model assumptions, derivations, or convergence analysis.
- Detailed description of algorithm steps, optimization strategies, or evaluation design.

**Not scored:** Literature review completeness (for reference only, not factored into contentQuality).
Output JSON: contentQuality (float in [0,1]), rationale ("brief explanation").

*Figure 31.* System prompt for the Content Quality evaluation of generated introductions.

Two composite scores are computed as weighted sums of the six sub-metrics (raw coverage, anchor coverage, core path coverage, structure alignment, faithfulness, and content quality). The *Overall Balanced* score distributes weights evenly across coverage sub-metrics (0.10 each) while giving moderate emphasis to faithfulness (0.28) and content quality (0.32). The *Overall Technical Focus* score up-weights core scientific path coverage (0.15) and content quality (0.40) to prioritize method and evidence depth over background breadth.

To assess evaluation robustness, we employ three independent evaluation models (`GPT-5.4`, `Claude Opus`, and `Gemini`) and report per-judger results, each averaged over 5 independent runs. The results are summarized in Table 20. The `IntroductionAgent` consistently outperforms the base model across all metrics and all three papers. The most substantial improvement is observed in Content Quality, with an average gain of +52.3% and a peak of **+121.4%** on Zhang et al. (2026)(Opus judger), indicating that the agent produces introductions with significantly richer technical depth. Faithfulness also improves on average (+7.7%), though individual judgers vary: GPT-5.4 and Opus report large gains, while Gemini, which already assigns high faithfulness to the base model, shows smaller margins. Notably, the improvements are most pronounced on Zhang et al. (2026), where the base model struggles the most, suggesting that the `IntroductionAgent` scales better on harder papers.

*Table 20.* Per-paper evaluation of the `IntroductionAgent` vs. the base model (`GPT-4o`) across three evaluation models (`GPT-5.4`, `Claude Opus`, `Gemini`). Each cell reports the mean over 5 independent runs. For each metric, the best ReasFlow score and the best advantage are highlighted in **bold**.

| Paper | Judger | Coverage | | | Faithfulness | | | Content Quality | | | Overall Balanced | | | Overall Tech. Focus | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Base | RF | Adv. | Base | RF | Adv. | Base | RF | Adv. | Base | RF | Adv. | Base | RF | Adv. |
| He et al. (2026) | `GPT-5.4` | 0.482 | 0.621 | +28.8% | 0.780 | 0.874 | +12.1% | 0.372 | 0.564 | +51.6% | 0.554 | 0.683 | +23.2% | 0.520 | 0.661 | +27.0% |
| | `Opus` | 0.354 | **0.677** | **+91.4%** | 0.820 | 0.880 | +7.3% | 0.320 | 0.620 | +93.8% | 0.510 | 0.739 | **+45.0%** | 0.461 | 0.708 | **+53.6%** |
| | `Gemini` | 0.489 | 0.674 | +37.7% | 0.950 | 0.890 | −6.3% | 0.610 | **0.830** | +36.1% | 0.677 | **0.795** | +17.3% | 0.658 | **0.788** | +19.8% |
| | Average | 0.441 | 0.657 | +48.8% | 0.850 | 0.881 | +3.7% | 0.434 | 0.671 | +54.7% | 0.580 | 0.739 | +27.3% | 0.547 | 0.719 | +31.6% |
| Zhu et al. (2026) | `GPT-5.4` | 0.455 | 0.492 | +8.2% | 0.766 | 0.942 | +23.0% | 0.400 | 0.490 | +22.5% | 0.552 | 0.642 | +16.4% | 0.509 | 0.603 | +18.3% |
| | `Opus` | 0.350 | 0.519 | +48.3% | 0.900 | 0.860 | −4.4% | 0.320 | 0.520 | +62.5% | 0.527 | 0.642 | +21.6% | 0.474 | 0.602 | +27.0% |
| | `Gemini` | 0.348 | 0.519 | +49.0% | 0.978 | **0.950** | −2.9% | 0.650 | 0.640 | −1.5% | 0.656 | 0.705 | +7.6% | 0.629 | 0.672 | +6.7% |
| | Average | 0.384 | 0.510 | +32.7% | 0.881 | 0.917 | +4.1% | 0.457 | 0.550 | +20.4% | 0.578 | 0.663 | +14.7% | 0.538 | 0.625 | +16.3% |
| Zhang et al. (2026) | `GPT-5.4` | 0.360 | 0.495 | +37.4% | 0.704 | 0.874 | **+24.1%** | 0.304 | 0.580 | +90.8% | 0.465 | 0.648 | +39.5% | 0.418 | 0.622 | +48.8% |
| | `Opus` | 0.379 | 0.424 | +11.9% | 0.770 | 0.928 | +20.5% | 0.280 | 0.620 | **+121.4%** | 0.482 | 0.657 | +36.2% | 0.432 | 0.629 | +45.5% |
| | `Gemini` | 0.382 | 0.452 | +18.4% | 0.941 | **1.000** | +6.3% | 0.450 | 0.780 | +73.3% | 0.587 | 0.734 | +24.9% | 0.548 | 0.719 | +31.3% |
| | Average | 0.374 | 0.457 | +22.3% | 0.805 | 0.934 | +16.0% | 0.345 | 0.660 | +91.5% | 0.511 | 0.679 | +32.9% | 0.466 | 0.657 | +40.9% |
| Overall Average | | 0.400 | 0.541 | +35.4% | 0.845 | 0.911 | +7.7% | 0.412 | 0.627 | +52.3% | 0.557 | 0.694 | +24.6% | 0.517 | 0.667 | +29.1% |

## B.7. Evaluation Details for Sec. 4.3.6

This appendix documents the full setup for the `WritingAgent` sub-experiment of Sec. 4.3.6: the eleven compared systems and their configurations, the shared input bundle and task prompt, the four-phase evaluation pipeline, the equal-weight scoring rule, the reviewer-model ensemble, and the representative evaluation prompts. Per-dimension sub-score tables are provided at the end.

### B.7.1. Generation Details

**Compared Systems.** We compare the ReasFlow `WritingAgent` against ten baselines drawn from three families of paper-writing systems. The first family consists of two general-purpose coding agents, *Claude Code* and *Codex*, each run in two configurations: a plain *vanilla* configuration with no additional domain skills, and an *ARIS* (Yang et al., 2026) configuration that augments the coding agent with a scientific-writing skill pack covering paper-structure planning, asset integration, and a compile-then-review loop. Each of the four resulting variants is run once under `GPT-5.1` and once under

`GPT-5.4`, giving the eight coding-agent systems in a $2 \times 2 \times 2$ matrix. The second family is the scientific-writing agent *ReasLingo*, a publicly available system on the ReasLab platform built on top of the Gemini CLI; we evaluate it twice, once with a `GPT-5.1` backend and once with `GPT-5.4`. Our own `WritingAgent` (Sec. **??**) is run with a `GPT-5.1` backend, giving eleven systems in total.

The fairness of the comparison rests on giving every one of the eleven systems exactly the same input bundle that the ReasFlow `WritingAgent` itself receives when it writes He et al. (2026). This bundle consists of an asset directory with the full upstream research artifacts (proved theorems, problem setup and algorithm pseudocode, integrated proof report, experiment code with raw numerical outputs, main experiment report, pre-generated experiment figures, literature survey, draft introduction, and reference `.bib` files); the target venue's LaTeX template, including the document class and the algorithm style dependencies needed for compilation; and the shared task prompt (reproduced below), which describes the task, the inputs, the required deliverables, the non-negotiable content-completeness requirements, and the compile-cleanliness criterion. The asset bundle and the template are byte-identical across all compared systems and identical to what the `WritingAgent` reads, and the task prompt is delivered verbatim as each system's initial instruction.

Every system is then run as a black box inside its own isolated working directory and is allowed to complete the task autonomously, producing its deliverables under a `./paper/` subdirectory. ReasLingo is the only compared system without a local CLI; we submit the same inputs through its hosted platform and collect the resulting `main.tex`, `references.bib`, and compiled PDF for evaluation. No system-specific tuning, prompt engineering, or intermediate supervision is applied, and evaluation operates on whatever deliverables each system produces when its own run terminates.

**Shared Task Prompt.** The task prompt below is delivered verbatim to every compared system. It is the same prompt that the ReasFlow `WritingAgent` receives for He et al. (2026), with no system-specific adaptation.

**Shared task prompt provided to every compared system**

You are an AI agent tasked with producing a publication-ready research paper from a bundle of paper-writing assets placed in your working directory. The topic, methods, theory, experimental data, and references must all come from the assets; do not invent anything from memory or general knowledge.
The working directory contains two input folders. The assets directory holds the full research materials for the paper; begin by reading its top-level README, which is the authoritative overview of the bundle and maps paper sections to their source files. The template directory holds the official LaTeX template for the target venue, including the document class, a skeleton example file, and the algorithm style dependencies. Write every deliverable inside a paper subdirectory of the working directory: the master TeX file must begin with the template's documentclass line, the references file must merge every bibliography file from the assets, and the final PDF must be produced by running latexmk on the master TeX file inside that paper subdirectory.
The following content requirements are hard red lines and non-negotiable. Every algorithm environment in the assets must be preserved in the paper. Every theorem, lemma, corollary, proposition, and definition must be preserved in full; phrasing may be edited, but conditions, conclusions, constants, and assumptions must remain exact. Every proof must be preserved completely, with no "proof omitted" short-circuits; if proofs live in the appendix, the full version must appear there. Every number, table, and experiment figure from the assets must be presented faithfully, with numbers matching the raw data exactly and no fabrication or approximation. Every bibliography entry from the assets must be merged into the paper's references file, and every in-text citation must resolve to such an entry.
The final PDF must compile with zero LaTeX errors and zero warnings of any kind, including undefined references, missing citations, overfull or underfull hboxes, and font warnings. Run latexmk iteratively: after each compile, read the log, fix every error and every warning, and recompile until the log is clean.
Complete the entire task in a single autonomous run. Do not pause to ask for confirmation or clarification; for any uncertainty, make the most reasonable default choice and continue. Do not deliver partial results.
The paper must be written in English. Keep the author block as an anonymous placeholder and do not write any real names, affiliations, emails, or acknowledgements.

*Figure 32.* Shared task prompt delivered to all eleven compared systems, reproduced with only whitespace normalized for typesetting.

### B.7.2. Evaluation Details

Once every system has produced a manuscript, evaluation proceeds in four phases (one per reporting dimension). All phases consume, when relevant, a common Phase 0 artifact bundle that is extracted programmatically from each output; no phase reads another phase's output.

**Phase 0 (programmatic preparation).** For every system we recursively read the content-facing asset `.tex` files; reconstruct the full generated LaTeX source by following `\input` and `\include`; extract text from the generated PDF; render

representative PDF pages as JPEG images for vision input; parse the LaTeX compilation log; extract citation keys from both the assets and the generated paper; and verify bibliography entries against arXiv and CrossRef. Phase 0 records only facts (sets of keys, file lists, page counts, compile error counts), never judgments.

**Phase 1 (Faithfulness, dimension I).** A reviewer model, given the assets, the baseline-generated paper, citation-set facts, and programmatic reference/paper signals, scores five sub-dimensions on a 0–10 integer scale: I1 theorem fidelity, I2 algorithm fidelity, I3 numerical fidelity, I4 figure fidelity, I5 citation fidelity. The reviewer must flag every *fabrication* (statement with no asset source), *distortion* (content present in assets but altered in the paper), and *phantom citation* (plausible-looking but non-existent bib entry), supported by a quoted paper snippet as evidence.

**Phase 2 (Coverage, dimension II).** A reviewer model scores five sub-dimensions: II1 theorem usage, II2 algorithm usage, II3 experiment usage, II4 figure usage, II5 citation usage. The rubric is deliberately abstract: reviewers identify landmark references and core assets from their training knowledge together with the programmatic signals extracted in Phase 0, without the rubric naming any paper-specific artifact. This keeps the evaluation generalizable across papers.

**Phase 3 (Writing Quality, dimension III).** The reviewer is given only the generated paper and paper-side signals (no assets), so that writing quality is judged self-referentially. The submitted `references.bib` is pre-filtered to the cited subset so that systems which copy the entire source bibliography are not unfairly punished on prompt length while a majority of entries are never actually cited. Four sub-dimensions are scored: III1 clarity, III2 coherence (project focus), III3 argumentation, III4 evidence use.

**Phase 4 (Submittability, dimension IV).** Three sub-components are scored: IV1 *compile*, a programmatic check of the LaTeX compile log (number of errors, undefined references, overfull/underfull hboxes, font warnings; 10 iff all zero; 0 iff no PDF is produced); IV2 *visual layout*, a two-step vision pipeline in which `gemini-3.1-flash-lite-preview` first produces a structured description of every PDF page (identifying placeholders, unresolved references, empty bib fields, rendering anomalies, page-level layout issues) and then the three reviewer models each score three visual sub-dimensions (IV2_1 layout compliance, IV2_2 rendering integrity, IV2_3 content presentation) from those descriptions; and IV3 *template compliance*, a programmatic check of the `\documentclass` line and of the absence of banned package families. IV1 and IV3 are binary-style integer scores in $\{0, \ldots, 10\}$; IV2 is a real-valued mean across the three visual sub-dimensions and three reviewers.

**Scoring Rule (Equal Weight at Every Level).** Let $S^r_{d,k}$ denote the integer score $\in [0, 10]$ given by reviewer $r \in \{\texttt{GPT-5.1}, \texttt{GPT-5.4}, \texttt{Gemini-3-Pro}\}$ to sub-dimension $k$ of dimension $d \in \{\text{I, II, III, IV}\}$. Let $K_d$ be the number of sub-dimensions of $d$ ($K_{\text{I}} = K_{\text{II}} = 5$, $K_{\text{III}} = 4$, $K_{\text{IV}} = 3$). Equal-weight aggregation proceeds in three steps:

$$\bar{S}_{d,k} = \tfrac{1}{3}\sum_{r} S^r_{d,k}, \qquad S_d = \tfrac{1}{K_d}\sum_{k=1}^{K_d} \bar{S}_{d,k}, \qquad \text{Overall} = \tfrac{1}{4}\big(S_{\text{I}} + S_{\text{II}} + S_{\text{III}} + S_{\text{IV}}\big).$$

For Phase IV, $S_{\text{IV}}$ is itself an equal-weight mean of D1, D2, D3 (each already an equal-weight mean over its own components). When no PDF is produced, D1 and D2 are both set to 0 and only D3 contributes to $S_{\text{IV}}$.

**Reviewer Ensemble.** To reduce reviewer-model bias we use three backends in parallel for every LLM-based sub-dimension: `GPT-5.1` (reviewer 1), `GPT-5.4` (reviewer 2), and `Gemini-3-Pro` (reviewer 3). Every reviewer receives the same prompt and returns a strict-JSON report with integer sub-scores and evidence-bearing findings; we aggregate by straight mean. For the Phase 4 visual stage, page descriptions are produced once by a single cheap vision describer (`gemini-3.1-flash-lite-preview`) and then consumed by the three scoring reviewers, so that vision noise does not propagate through the ensemble. Prompts are designed for cache-friendly layout: shared asset content is placed at the front of the prompt and reviewer-specific task structure at the back, which allows the three reviewer backends to reuse their prefix cache across all eleven compared systems.

**Evaluation Prompts.** Every phase shares the same prompt layout. The prompt starts with a sequence of XML-style data blocks whose contents are filled in at run time from the Phase 0 artifacts of the system under review, and the phase-specific task description, scoring rubric, and output schema come after them. Placing the run-time data before the rubric lets the three reviewer backends reuse their prefix cache across all eleven compared systems, since only the part after the data blocks changes between phases.

Phases 1 and 2 use the same five data blocks. The ⟨`assets`⟩ block carries the full source materials for the paper under evaluation, concatenated into a single XML block by Phase 0 from the paper's proved theorems, problem setup and algorithm

pseudocode, integrated proof report, main experiment report, and source bibliography files. The ⟨reference_signals⟩ block carries the programmatically extracted anchor signals that Phase 0 computes directly from the assets (for example, which theorem files are declared core by the source proof report, or which citation keys appear in the draft introduction and related-work files). The ⟨paper⟩ block carries the manuscript produced by the system under review: Phase 0 reconstructs the full generated LaTeX source by following the paper's `input` and `include` statements, so that every section file appears in the block together with the paper's own references file. The ⟨cite_check⟩ block carries the set-operation facts that Phase 0 computes between the source bibliography keys and the paper's cited keys (keys cited from the source, keys unused from the source, keys added by the paper, broken citations). The ⟨paper_signals⟩ block carries the programmatic facts Phase 0 extracts from the paper itself, such as which figures are actually rendered and the section-heading structure. Phase 3 uses only the ⟨paper⟩ block (with its references file filtered down to the cited subset) and the ⟨paper_signals⟩ block, so that writing quality is judged from the manuscript alone. Phase 4 uses two purpose-built blocks: ⟨page_descriptions⟩ carries the concatenation of per-page structured descriptions produced by the vision stage below, and ⟨compile_facts⟩ and ⟨template_facts⟩ carry the programmatic compile-log counters and the documentclass / banned-package checks.

The phases are also distinguished by how many times each prompt is invoked. In Phases 1, 2, and 3 the prompt is sent, for each of the eleven systems under comparison, to three reviewer backends (`GPT-5.1`, `GPT-5.4`, `Gemini-3-Pro`) in parallel, giving 33 independent LLM calls per phase; the per-sub-dimension scores are then averaged across the three reviewers with equal weight. Phase 4 is a two-stage protocol. In the first stage, the cheap vision model `gemini-3.1-flash-lite-preview` is called once per rendered PDF page of each system, producing one structured description per page (several hundred calls in total across the eleven systems); the prompt used in this stage is shown in Fig. 36. In the second stage, those per-page descriptions are concatenated into the ⟨page_descriptions⟩ block and the combined prompt is sent, again for each of the eleven systems, to the same three reviewer backends in parallel for a further 33 scoring calls; the rubric used in this stage is shown in Fig. 37. The rubric portion of each prompt, which is independent of any particular paper and constitutes the reviewer's actual standard of judgment, is reproduced in the LLM boxes that follow.

**Phase 1 evaluation prompt: Faithfulness (dimension I)**

Run-time data blocks (inserted at the top of the prompt before the rubric):
<assets> ... full research materials for the paper under evaluation ... </assets> <reference_signals> ... programmatic anchor signals from the source assets ... </reference_signals> <paper> ... every .tex file plus the references file of the manuscript produced by the system under review ... </paper> <cite_check> ... citation-set operations between source bibliography keys and the paper's cited keys ... </cite_check> <paper_signals> ... figures actually used by the paper and its section-heading structure ... </paper_signals>
<task> You are the Faithfulness Reviewer for academic papers. Given a paper automatically generated by an AI system, together with the raw research assets the system had access to, judge whether every statement in the paper can be traced to an asset, and whether there is fabrication or distortion. Your output is a strict JSON report with integer scores from 0 to 10 on five sub-dimensions plus a list of findings, each supported by a quoted snippet from the paper as evidence; do not judge novelty, experimental design, or writing style, all of which belong to other reviewers. </task>
<definition> Faithfulness means that every verifiable statement in the paper has a matching source in the assets, or is a clearly legitimate generation such as an architecture diagram in a Method section. Unfaithfulness takes three specific forms: fabrication of a claim with no asset source; distortion of content that does exist in the assets but is altered in the paper; and phantom citation of a plausible-looking but non-existent bibliographic entry. </definition>
<rubric> The five sub-dimensions are scored independently. I1 Theorem Fidelity checks that every theorem, lemma, corollary, or proposition in the paper is mathematically consistent with a source in <assets>; cross-directory inclusion of asset files (so the paper is not a self-contained document) is itself a defect. I2 Algorithm Fidelity checks that the paper's pseudocode agrees with the source algorithm on step order, loop nesting, update rules, and communication timing. I3 Numerical Fidelity checks that every specific number in the paper (metric, hyper-parameter, dataset size, assumption constant) is traceable to a number in the source experiment report or algorithm specification. I4 Figure Fidelity checks that every figure used in the paper has a legitimate source, that its caption is consistent with that source, and that it is referenced from the main text. I5 Citation Fidelity checks that every newly added bibliography entry (those listed under `added_bib_keys` in <cite_check>) is a real reference, that every citation is placed in a contextually appropriate section, and that no citation in `broken_cites` is unresolved. Each sub-dimension is scored on the same 0-10 scale: 10 when fully consistent with the source; 8 or 9 for minor phrasing or symbol differences with no semantic change; 6 or 7 for one core-level issue or several minor ones; 4 or 5 for multiple core issues; 2 or 3 when core content is fabricated or severely distorted; 0 or 1 when fabrication or mathematical errors are pervasive. </rubric>
<output_schema> Return JSON only, with per-sub-dimension integer scores and a list of findings. Every finding must carry its severity, its location in the paper, the relevant asset reference when applicable, a short issue description, and a quoted paper snippet as evidence; all findings of severity high or medium must be listed, while low may be omitted. </output_schema>

*Figure 33.* Core prompt structure for Phase 1 (Faithfulness).

**Phase 2 evaluation prompt: Coverage (dimension II)**

Run-time data blocks (inserted at the top of the prompt before the rubric):
<assets> ... full research materials for the paper under evaluation ... </assets> <reference_signals> ... programmatic anchor signals from the source assets ... </reference_signals> <paper> ... the manuscript produced by the system under review ... </paper> <cite_check> ... citation-set operations between source and paper ... </cite_check> <paper_signals> ... figures used by the paper, figures present in assets but unused, and section-heading structure ... </paper_signals>
<task> You are the Asset Usage Judgment Reviewer. Given the input assets and the generated paper, judge whether the paper made reasonable editorial choices about which materials to include. Reasonable usage requires two things simultaneously: on the positive side, the core content of the source assets must be represented in the paper; on the negative side, the paper must not indiscriminately include non-core, redundant, or debug material as if it were important content. Your output is a strict JSON report with integer scores from 0 to 10 on five sub-dimensions plus a list of findings. Do not judge content correctness (Phase 1), writing style (Phase 3), or compilation and layout (Phase 4). </task>
<rubric> The five sub-dimensions each weigh the missing-core side and the junk-inclusion side independently. II1 Theorem Usage asks whether the core theorems (those declared core in <reference_signals>) are actually presented and whether technical lemmas are kept out of the main text rather than dumped wholesale. II2 Algorithm Usage asks whether the problem definition, assumptions, and pseudocode are all present and whether design-rationale drafts are rewritten into the method discussion rather than pasted verbatim. II3 Experiment Usage asks whether every standalone experiment section from the source is represented in the paper, whether raw data tables are accompanied by analysis rather than stacked into an appendix, and whether no experiment sections appear that are not present in the source. II4 Figure Usage asks whether the core figures identified via <paper_signals> are used and whether debug or intermediate figures have been kept out of the main text. II5 Citation Usage asks whether foundational references for the paper's topic are cited (reviewers identify these from training knowledge; the rubric itself never names any specific paper) and whether the overall citation count is reasonable. Each sub-dimension is scored on the same 0-10 scale: 10 when the core is fully covered and no junk has accumulated; 8 or 9 for one minor omission or a couple of pieces of junk with the core still intact; 6 or 7 for one core omission or three or more junk pieces; 4 or 5 for multiple core omissions or large-scale junk; 2 or 3 for both widespread missing core and large-scale junk; 0 or 1 when there is almost no reasonable editorial judgment. </rubric>
<output_schema> Return JSON only with per-sub-dimension integer scores and findings. The rubric itself must not name any specific paper, method, or reference; reviewers identify landmarks from their own training knowledge. </output_schema>

*Figure 34.* Core prompt structure for Phase 2 (Coverage).

**Phase 3 evaluation prompt: Writing Quality (dimension III)**

Run-time data blocks (inserted at the top of the prompt before the rubric):
<paper> ... every .tex file plus the references file, with the bib filtered to the cited subset only ... </paper> <paper_signals> ... section-heading structure and figures actually used by the paper ... </paper_signals>
<task> You are the Writing Quality Reviewer. Judge the paper purely as academic prose, that is, language, flow, argumentation, and integration of evidence. The assets are intentionally withheld from this phase so that writing quality is judged self-referentially. Your output is a strict JSON report with integer scores from 0 to 10 on four sub-dimensions plus findings with quoted evidence. Do not judge factual correctness (Phase 1), whether source material was used correctly (Phase 2), or compilation, layout, and template compliance (Phase 4). </task>
<rubric> Four orthogonal sub-dimensions are scored independently. III1 Clarity concerns sentence form, terminology consistency, precision, grammar, and typography at the sentence-to-paragraph level. III2 Coherence, the emphasized sub-dimension for this project, concerns paragraph-to-paragraph and section-to-section logical flow: whether forward and backward references resolve to real content, whether the paper has a complete narrative arc rather than a collection of disconnected sub-papers, and whether the abstract is aligned with what the body actually delivers. III3 Argumentation concerns whether every core claim comes with motivation, explicit statement, supporting evidence, and closure. III4 Evidence Use concerns whether figures, tables, and citations are integrated into the argument at the place where the reader needs them rather than left decorative. Each sub-dimension is scored on the same 0-10 scale: 10 when the paper reads like a polished, well-edited piece of work; 8 or 9 when it is solid with only minor non-distracting issues; 6 or 7 when it is readable but noticeably rough; 4 or 5 when it is consistently problematic and requires effort to follow; 2 or 3 when it has widespread problems that make it difficult to read; 0 or 1 when it is incoherent or unintelligible. </rubric>
<output_schema> Return JSON only with per-sub-dimension integer scores and findings. Every finding must carry its severity, its location in the paper, a short issue description, and a quoted snippet as evidence. </output_schema>

*Figure 35.* Core prompt structure for Phase 3 (Writing Quality).

**Phase 4a evaluation prompt: per-page vision description (stage 1 of Submittability)**

Run-time data block (a single page image is inserted, one per invocation):
<page_image> ... one rendered page of the PDF produced by the system under review ... </page_image>
<task> You analyze a single PDF page from an academic paper draft in the target venue's format. Your job is to identify the content and structural elements present on the page, flag visual defects that would concern a journal editor, and emit one coarse page-level quality signal. A downstream reviewer will aggregate these per-page analyses to score the whole PDF's visual submittability; you focus on one page and do not infer beyond what is visible. </task>
<context> The paper was auto-generated by an AI writing system given a LaTeX template. A journal editor would desk-reject if the PDF contains raw LaTeX as text, uses filenames as section headings, has data dumps replacing prose, or has non-English content. </context>
<definition> Content types controlled enum: prose, equations, table, figure, algorithm pseudocode, code, csv data, references, raw latex source, header only, blank. Structure elements: title, author block, abstract, section heading, subsection heading, caption, page number, running header, footnote, reference item. Anomalies are drawn from a controlled enum grouped into three families: layout and whitespace (text overflow, image cut off, content cut off, mostly blank, excessive whitespace, isolated float on page, large whitespace gap, float at top only, float at bottom only, table columns overflow, equation broken line, caption separated from float); rendering (rendering garbage, font inconsistency, latex commands unrendered); content presentation (filename as section heading, non-English characters, missing page number, placeholder text visible, unresolved reference, empty bibliography fields). A four-level page flag is also emitted: clean (no anomalies), minor (one or two low-severity anomalies), major (one severe anomaly such as whole page as raw LaTeX, filename as heading, page dominated by csv data, or non-English content), and severe (multiple severe anomalies or the page is unreadable as a paper page). Key distinctions to avoid false positives: typeset pseudocode is not raw latex source; natural end-of-section or end-of-document whitespace is not a layout anomaly; a word hyphenated across a page boundary is standard LaTeX hyphenation and not content cut off; placeholder text visible is flagged only for literal template tokens (bibliography "Author Name" / "Author X" / "Unknown Author", body text [TODO] / XXX / FIXME / <NAME> / <CITE> / (cite?), and stub author patterns like "A. Author" or generic "Firstname Lastname"), and is not flagged for methodology prose describing asset sources, for math subscripts such as $c_1$ or $x_i$, for citation keys rendered as short labels, for CSIAM slots the journal fills at acceptance ("Vol. x, No. x, pp. X-Y", "doi:", "©20xx Global-Science Press"), or for policy-required anonymization ("Anonymous Authors", "Anonymous Institution"); empty bibliography fields is flagged only when an entry clearly lacks author, year, or venue, and is not flagged for long author lists continuing onto the next page or for "et al." abbreviations; unresolved reference is flagged only for "??" or "[??]" marks actually rendered where a reference should appear. </definition>
<output_schema> Return JSON only with the fields content_type (list from the content-types enum), structure_elements (list from the structure-elements enum), anomalies (list from the anomalies enum, empty if none), anomaly_details (free text with specifics, or empty string), page_flag (one of clean, minor, major, severe), notable_text (any unusual literal string on the page, or empty string), and summary (one factual sentence of at most 150 characters). </output_schema>
<prohibited> Do not assign a 0-10 score at this stage (only the four-level page flag is used); do not judge content correctness, asset usage, writing, or compile status (those are other phases); do not invent anomaly codes outside the enum; do not add any text outside the JSON. </prohibited>

*Figure 36.* Stage 1 of Phase 4: per-page structured description produced by a cheap vision model (gemini-3.1-flash-lite-preview), invoked once per page of each system's rendered PDF.

**Phase 4b evaluation prompt: visual scoring (stage 2 of Submittability)**

Run-time data blocks (inserted at the top of the prompt before the rubric):
<page_descriptions> ... concatenation of every per-page JSON produced by stage 1, in page order ... </page_descriptions> <compile_facts> ... IV1 counters extracted programmatically from the compile log: number of errors, undefined references, overfull and underfull hboxes, font warnings ... </compile_facts> <template_facts> ... IV3 checks: the documentclass line of main.tex and the presence or absence of banned package families ... </template_facts>
<task> You are the Visual Layout Quality Reviewer for stage 2 of Phase 4. Given the per-page structured descriptions plus the programmatic compile and template facts, simulate a journal editor's thirty-second skim and score submittability on three sub-components on a 0-10 integer scale. You do not see the page images directly; score only from <page_descriptions>, and distinguish isolated issues affecting one or two pages from systemic issues repeated throughout the document, weighting the latter more heavily. </task>
<rubric> IV1 Compile is a programmatic score derived from <compile_facts>: 10 iff every counter is zero, 0 iff no PDF was produced at all, and intermediate values degrade with the counter totals. IV2 Visual Layout is the LLM-scored component and is decomposed into three axes. IV2_1 Layout Compliance covers margins, overflow, whitespace distribution, and float positioning (with signals text overflow, image cut off, excessive whitespace, isolated float on page, large whitespace gap, float at top or bottom only, table columns overflow, equation broken across lines, caption separated from float, mostly blank, missing page number). IV2_2 Rendering Integrity covers whether content is rendered as intended (signals latex commands unrendered and non-English characters are both critical and cap IV2_2 sharply, rendering garbage is critical, font inconsistency is a minor deduction). IV2_3 Content Presentation covers whether the PDF reads as an academic paper rather than a compilation artifact (critical signals filename

as section heading, placeholder text visible, unresolved reference, empty bibliography fields; caps also apply when csv data dominates pages, when the abstract is missing on page 1, or when there is no references section at the end). Each IV2 sub-axis uses the same 0-10 scale: 10 when the axis is essentially flawless; 8 or 9 when one or two isolated issues appear with no systemic pattern; 6 or 7 for noticeable but localized problems; 4 or 5 for widespread problems or multiple critical defects; 2 or 3 for systemic severe issues; 0 or 1 when the axis catastrophically fails. IV3 Template is a programmatic score from <template_facts>: 10 iff the documentclass matches the target template and no banned packages are used. The aggregation is equal weight: the final IV2 is the mean over IV2_1, IV2_2, IV2_3 and over the three reviewer models, and the final IV submittability score is the mean of IV1, IV2, and IV3; when no PDF is produced, IV1 and IV2 are both set to 0 and only IV3 contributes. </rubric> <output_schema> Return JSON only with per-sub-component integer scores, a per-page count summary (total pages and counts of clean, minor, major, severe pages), and findings that cite specific page numbers and quote short excerpts from <page_descriptions> as evidence. </output_schema>

*Figure 37.* Stage 2 of Phase 4: the three reviewer backends (GPT-5.1, GPT-5.4, Gemini-3-Pro) score visual submittability from the stage-1 page descriptions and the programmatic compile and template facts.

**Per-Dimension Sub-Score Tables.** Tables 21–24 report the sub-dimension breakdown underlying Table 7. Every entry is already averaged across the three reviewer models with equal weight; the rightmost column reproduces the dimension score from the main table.

*Table 21.* Phase 1 (Faithfulness) sub-dimension scores. I1 theorem, I2 algorithm, I3 numerical, I4 figure, I5 citation fidelity.

| System | I1 | I2 | I3 | I4 | I5 | I |
|---|---|---|---|---|---|---|
| ReasFlow WritingAgent (GPT-5.1) | **10.00** | **10.00** | 9.00 | **10.00** | 9.67 | **9.73** |
| Claude Code + ARIS (GPT-5.4) | 7.67 | 9.33 | **9.67** | **10.00** | 8.67 | 9.07 |
| Claude Code + ARIS (GPT-5.1) | 4.67 | 10.00 | 8.33 | 8.00 | 7.33 | 7.67 |
| Claude Code, vanilla (GPT-5.1) | 4.67 | 7.33 | 8.67 | 8.67 | **10.00** | 7.87 |
| Claude Code, vanilla (GPT-5.4) | 9.33 | 10.00 | 4.33 | 7.33 | **10.00** | 8.20 |
| Codex + ARIS (GPT-5.1) | 4.67 | 10.00 | 5.67 | 9.67 | 8.00 | 7.60 |
| Codex + ARIS (GPT-5.4) | 6.67 | 10.00 | 8.67 | 9.00 | 8.00 | 8.47 |
| Codex, vanilla (GPT-5.1) | 4.67 | 9.33 | 8.00 | 8.00 | 8.33 | 7.67 |
| Codex, vanilla (GPT-5.4) | 5.00 | 9.33 | 4.00 | 8.33 | 5.67 | 6.47 |
| ReasLingo (GPT-5.1) | 5.00 | 10.00 | 9.33 | 9.33 | 8.67 | 8.47 |
| ReasLingo (GPT-5.4) | 8.00 | 10.00 | 8.00 | 8.67 | 7.67 | 8.47 |

*Table 22.* Phase 2 (Coverage) sub-dimension scores. II1 theorem, II2 algorithm, II3 experiment, II4 figure, II5 citation usage.

| System | II1 | II2 | II3 | II4 | II5 | II |
|---|---|---|---|---|---|---|
| ReasFlow WritingAgent (GPT-5.1) | 8.67 | **9.33** | **10.00** | **10.00** | **9.33** | **9.47** |
| Claude Code + ARIS (GPT-5.4) | 5.00 | 6.33 | **10.00** | **10.00** | **9.33** | 8.13 |
| Claude Code + ARIS (GPT-5.1) | 4.67 | 7.67 | 5.33 | 6.33 | 7.67 | 6.33 |
| Claude Code, vanilla (GPT-5.1) | 4.67 | 8.00 | 7.33 | 7.00 | 9.00 | 7.20 |
| Claude Code, vanilla (GPT-5.4) | **9.33** | **9.33** | 8.33 | **10.00** | 8.67 | 9.13 |
| Codex + ARIS (GPT-5.1) | 5.00 | 8.00 | 9.67 | 9.00 | 8.67 | 8.07 |
| Codex + ARIS (GPT-5.4) | 3.67 | 6.00 | 2.67 | 4.00 | 5.33 | 4.33 |
| Codex, vanilla (GPT-5.1) | 4.33 | 8.00 | 2.67 | 6.67 | 6.67 | 5.67 |
| Codex, vanilla (GPT-5.4) | 3.33 | 6.00 | 3.33 | 8.67 | 5.33 | 5.33 |
| ReasLingo (GPT-5.1) | 6.00 | 8.67 | 8.33 | 7.00 | 8.00 | 7.60 |
| ReasLingo (GPT-5.4) | 3.67 | 6.00 | 2.67 | 4.33 | 7.33 | 4.80 |

**Discussion.** The sub-score tables above indicate that faithfulness alone is not discriminative enough to separate the compared systems. Most systems receive an I2 algorithm-fidelity score of at least 9.33, because the task prompt explicitly marks algorithm pseudocode as a hard red line and nearly every system copies the pseudocode correctly; faithfulness differentiates the compared systems mainly on I1 theorem fidelity, where WritingAgent scores 10.00 against a median of about 5.00 for baselines that drop most theorems, and on I3 numerical fidelity, where systems that copy numbers carefully separate from systems that round aggressively or regenerate.

Coverage is where the overall ranking is mostly decided. The six systems at the bottom of the overall table all have

*Table 23.* Phase 3 (Writing Quality) sub-dimension scores. III1 clarity, III2 coherence (project focus), III3 argumentation, III4 evidence use.

| System | III1 | III2 | III3 | III4 | III |
|---|---|---|---|---|---|
| ReasFlow `WritingAgent` (`GPT-5.1`) | 8.33 | **7.67** | **8.33** | **8.33** | **8.16** |
| Claude Code + ARIS (`GPT-5.4`) | 8.00 | 6.00 | 7.67 | 7.67 | 7.33 |
| Claude Code + ARIS (`GPT-5.1`) | 6.67 | 5.33 | 5.67 | 4.00 | 5.42 |
| Claude Code, vanilla (`GPT-5.1`) | 8.00 | 7.33 | 7.67 | 5.67 | 7.17 |
| Claude Code, vanilla (`GPT-5.4`) | **8.67** | 7.33 | 8.00 | 6.67 | 7.67 |
| Codex + ARIS (`GPT-5.1`) | 8.33 | 6.67 | 8.00 | 8.00 | 7.75 |
| Codex + ARIS (`GPT-5.4`) | 7.33 | 4.67 | 5.33 | 5.00 | 5.58 |
| Codex, vanilla (`GPT-5.1`) | 8.33 | 7.33 | 7.33 | 6.33 | 7.33 |
| Codex, vanilla (`GPT-5.4`) | 5.67 | 4.00 | 4.33 | 4.33 | 4.58 |
| ReasLingo (`GPT-5.1`) | 7.00 | 5.33 | 5.67 | 7.00 | 6.25 |
| ReasLingo (`GPT-5.4`) | 7.00 | 4.33 | 4.33 | 4.67 | 5.08 |

*Table 24.* Phase 4 (Submittability) sub-components. IV1 compile (programmatic), IV2 visual layout (three LLM sub-scores IV2_1 layout compliance, IV2_2 rendering integrity, IV2_3 content presentation, each averaged over three reviewers), IV3 template compliance (programmatic). A dash (—) in a IV2 sub-column indicates no PDF was produced; in that case IV1=IV2=0 and only IV3 contributes to IV.

| System | IV1 | IV2_1 | IV2_2 | IV2_3 | IV2 | IV3 | IV |
|---|---|---|---|---|---|---|---|
| ReasFlow `WritingAgent` (`GPT-5.1`) | **10** | 9.00 | **10.00** | **10.00** | 9.67 | **10** | 9.87 |
| Claude Code + ARIS (`GPT-5.4`) | 6 | **10.00** | **10.00** | 3.67 | 7.89 | **10** | 7.56 |
| Claude Code + ARIS (`GPT-5.1`) | 0 | — | — | — | 0.00 | **10** | 2.00 |
| Claude Code, vanilla (`GPT-5.1`) | 1 | 8.00 | 5.33 | 3.33 | 5.55 | **10** | 4.62 |
| Claude Code, vanilla (`GPT-5.4`) | **10** | **10.00** | **10.00** | **10.00** | **10.00** | **10** | **10.00** |
| Codex + ARIS (`GPT-5.1`) | **10** | 7.67 | 9.67 | 4.33 | 7.22 | **10** | 8.89 |
| Codex + ARIS (`GPT-5.4`) | **10** | 2.33 | 5.33 | 1.67 | 3.11 | **10** | 7.24 |
| Codex, vanilla (`GPT-5.1`) | **10** | 2.33 | 9.67 | 1.67 | 4.56 | **10** | 7.82 |
| Codex, vanilla (`GPT-5.4`) | **10** | 3.00 | 1.33 | 1.00 | 1.78 | **10** | 6.71 |
| ReasLingo (`GPT-5.1`) | **10** | 8.67 | **10.00** | 2.67 | 7.11 | **10** | 8.84 |
| ReasLingo (`GPT-5.4`) | **10** | 2.67 | 9.67 | 1.67 | 4.67 | **10** | 7.87 |

II Coverage no higher than 6, typically because they drop entire experiment sections or leave out parts of the algorithm specification; in contrast, systems that place above this threshold all cover the core experiments and algorithm ingredients, and the remaining gap between them is accounted for by writing quality and submittability.

Submittability finally exposes failure modes that Phases I to III cannot see. Two systems with otherwise reasonable content scores, *Claude Code + ARIS* under `GPT-5.1` and *Claude Code, vanilla* under `GPT-5.1`, fall to the bottom because of compilation or rendering failure: the former fails to produce a PDF at all because Chinese characters leak into the source and break the compile, while the latter compiles but accumulates 45 errors, 19 undefined citations, and 13 overfull hboxes. Such failures are invisible from the content-facing phases, yet they decisively determine whether the manuscript could actually be submitted, which is why the overall score is an equal-weight mean across all four dimensions rather than a weighted combination that could conceal them.