Another Turn, Better Output? :
A Turn-Wise Analysis of Iterative LLM Prompting

Anonymous ACL submission

Abstract

Large language models are increasingly used through multi-turn refinement loops, but most evidence for iterative improvement comes from settings with informative feedback, external tools, full conversation history, or structured self-critique. Less is known about a configuration, uninformative memoryless refinement, where a model repeatedly revises only its previous output under vague instructions such as "improve it". In this paper, we establish a framework to analyze when this uninformative memoryless refinement configuration improves LLM responses across ideation, coding, and mathematical reasoning. We evaluate seven LLMs on a 450-task pool under vague and domain-specific feedback. We score every turn and measure both task outcomes and trajectory dynamics, including semantic drift, volatility, lexical novelty, and output growth. This trajectory view reveals a consistent discovery–retention gap: models often produce better intermediate outputs before overwriting them. The effect is most severe in coding, where correctness falls from 40.7% at T1 to 2.5% at T12, while best@12 exceeds final@12 by 42.5 percentage points. Math shows model- and prompt-dependent behavior, with GPT-OSS-20B improving under elaboration but several models drifting downward. Ideation benefits from explicit steering, but vague feedback produces rhetorical inflation and feasibility loss. Our analysis stress the importance of turn-wise, trajectory-aware evaluation, and we recommend approaches to improve the prompts in multi-turn scenarios.

1 Introduction

Large language models (LLMs) are increasingly used as interactive collaborators rather than one-shot text generators. Users ask models to revise code, elaborate proofs, sharpen ideas, simplify explanations, or improve a previous answer across several turns. These workflows often assume that another turn gives the model a useful opportunity to revise its own output and move the response closer to the user's goal. Yet this assumption is not guaranteed: an additional turn can also overwrite a correct answer, amplify earlier mistakes, or expand an output without improving it.

Researchers have identified multiple configurations to study the source of the response quality improvement in multi-turn iterations. These configurations differ according to the feedback given to the LLMs. The feedback is mostly informative. The informative feedback may carry explicit dialog acts, including following up, supplementing information, and providing corrections (Li et al., 2025b). This type of informative feedback serve as instructions, and the LLMs following the instructions can improve the response quality. The feedback may come from the environments, including the tool execution results (Li et al., 2025a), and other signals returned by the environment (Yao et al., 2023). The LLMs respond to the environmental signals, therefore improve the response quality. In an interesting configuration, the feedback may even be generated by the LLM itself. SELF-REFINE (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) showed that the self-generated feedback can elicit some self-reflection behavior of the LLM, which may lead to performance improvements in coding and dialogue tasks.

However, a unique configuration has not been studied. We refer to it as *uninformative memoryless refinement*, a configuration that resembles a wide variety of real-world use cases. Here's an example that illustrates the configuration. At each conversation turn, the user appends an identical instruction, "improve it", to the previous turn response and sends them to the model. This example is signified by two properties: *uninformative* and *memoryless*. The user's instruction does not contain targeted instructions, tool execution results, or other concrete information. The input to the next

step does not contain the memory from previous turns. The model repeatedly edits its own output using only its previous response as context.

The uninformative memoryless configuration is helpful for exposing the LLMs' cause of the response quality in multi-turn iterations in a controlled setting. LLMs are shown to struggle when not provided additional information (Laban et al., 2025). They are usually not able to self-correct without external evidence (Ji et al., 2023; Gou et al., 2024; Xu et al., 2024). We avoid providing concrete information. LLM performances may be diluted at long contexts (Liu et al., 2024), and the repeated dependence on model outputs raises the broader risk of degenerative feedback loops (Mohamed et al., 2025). We do not allow the model to revisit prior context. In uninformative memoryless iteration, we are able to attribute the improvement in response pattern to the model's self-reasoning capability, without the impact of the instruction or context grounding.

In this paper, we study this empirical question: *when does memoryless uninformative refinement improve a response, and when does the model simply transforms, bloats, or degrades the response?*

We establish a framework to study memoryless uninformative refinement. We treat iterative prompting as a *turn-wise dynamical process*. Rather than asking only whether the final answer is good, we ask how the trajectory evolves: does the model preserve a correct intermediate answer, drift semantically from the original intent, become repetitive, inflate output length, or discover a better solution late? This perspective is important because different domains impose different constraints. In coding, the output must satisfy executable tests; in math, the final answer may be objectively checkable while the reasoning path can drift; in ideation, success is inherently multi-objective, balancing originality, feasibility, clarity, and resistance to linguistic inflation. A single final score can hide these trajectory-level behaviors. We intentionally study this austere setting not because it is optimal, but because it isolates a common failure mode of self-referential rewriting: once the original task disappears from context, refinement can become transformation rather than improvement.

Operationally, each task begins with an initial answer at Turn 1. For Turns 2–12, the model receives only its previous response and an improvement instruction. We compare vague feedback against domain-specific steering prompts that target natural trade-offs: novelty versus practicality for ideas, performance versus maintainability for code, and elaboration versus alternative reasoning for math. We ask three research questions:

- **RQ1:** Does memoryless uninformative refinement improve final output quality, or does it primarily transform outputs over turns?
- **RQ2:** How do refinement trajectories vary across task domains, models, and feedback strategies?
- **RQ3:** Which trajectory dynamics, semantic drift, turn-to-turn volatility, lexical novelty, and output growth—help explain domain-specific refinement success or failure?

We make three contributions. First, we introduce a controlled 12-turn evaluation protocol for studying iterative LLM refinement as a trajectory rather than as a final-turn outcome. Second, we operationalize discovery and retention through best@k, final@12, drift rate, and recovery rate, and pair these outcome measures with behavioral dynamics such as semantic drift, turn-to-turn volatility, lexical novelty, and output growth. Third, across a 450-task pool spanning ideation, coding, and math, we identify domain-specific fingerprints of refinement failure and translate them into practical policies for when to steer, stop, checkpoint, or restart.

The findings are consistent and quantitatively sharp. In coding, refinement is destructive: mean correctness falls from 40.7% at T1 to 2.5% at T12, while the gap between best@12 and final@12 reaches 42.5 pp. In math, iteration is model- and prompt-dependent: GPT-OSS-20B improves substantially under elaboration, but several other models drift away from correct answers. In ideation, explicit steering shifts judge-rated quality along the intended axis, whereas vague or novelty-seeking prompts can trigger rhetorical inflation and feasibility loss. Overall, under memoryless uninformative refinement, another turn is not inherently better; iteration must be governed by task-aware feedback, intrinsic checks, and stopping rules. These findings establish uninformative memoryless refinement as a diagnostic stress test for LLM self-revision, offering a controlled way to study when additional turns reveal capability, amplify errors, or overwrite correct intermediate outputs.

2 Related Work

A line of work studies whether LLMs can improve their own outputs through feedback. SELF-

REFINE alternates generation, feedback, and revision and reports 5–40% absolute gains across tasks (Madaan et al., 2023). Reflexion similarly uses verbal feedback to help agents revise behavior over time (Shinn et al., 2023). More recent work trains refinement ability: ARIES stimulates self-refinement by iteratively collecting refinement data and optimizing preferences (Zeng et al., 2025). Multi-agent systems such as MAgICoRe use coarse-to-fine collaboration to address error localization and excessive refinement (Chen et al., 2024), while ACT learns clarifying-question policies for mixed-initiative interaction (Chen et al., 2025). These works show that structured feedback can be powerful. Our question is complementary: how do refinement loops behave when the feedback is weak, domain-specific only in small ways, and not supported by an external verifier at every turn?

Recent evidence suggests that multi-turn use is not automatically easier or more reliable than single-turn prompting. Multi-turn simulations report a 39% average drop relative to single-turn generation across six tasks, driven largely by unreliability rather than lack of underlying ability (Laban et al., 2025). MultiChallenge similarly finds that frontier models remain below 50% on realistic multi-turn conversations despite strong one-shot performance elsewhere (Deshpande et al., 2025). In self-correction settings, asking a model to recheck itself can worsen accuracy and calibration (Krishna et al., 2024). Broader surveys and studies of critique, verification, and self-revision emphasize that self-correction often requires either external evidence or a stronger verifier (Ji et al., 2023; Gou et al., 2024; Xu et al., 2024). Our work extends this concern from single corrections to sustained 12-turn trajectories.

Evaluation of iterative systems must be careful because models can sound increasingly confident while becoming less correct. LLM-as-judge setups may overstate confidence (Tian et al., 2025); calibration studies document persistent overconfidence across model sizes and settings (Chhikara, 2025); and RLHF can induce verbalized overconfidence unless rewards are calibrated (Leng et al., 2025). Iteration also raises broader feedback-loop concerns. Recursive training on generated data can lead to model collapse (Shumailov et al., 2024), although accumulation with real data can mitigate collapse in some regimes (Gerstgrasser et al., 2024); theoretical work shows that collapse-like effects can still arise under particular assumptions (Barzilai and Shamir, 2025). While our setting is inference-time refinement rather than training-time data recursion, the shared concern is that repeated reliance on model-generated content can amplify errors.

Surveys of multi-turn agents emphasize the need for process-sensitive evaluation rather than final-answer-only scoring (Guan et al., 2025). Self-iterative label refinement offers one example of a safer iterative template by denoising pseudo-labels across rounds (Asano et al., 2025). Our contribution is a domain-spanning measurement of refinement trajectories. We do not introduce a new prompting algorithm; instead, we isolate how models behave under repeated improvement instructions and show that the same loop can have opposite effects depending on the domain, model, and feedback type.

3 Methodology

We study iterative refinement across three domains with different verification regimes: ideation is open-ended and multi-objective, coding has executable correctness, and math has ground-truth final answers but complex reasoning paths. For each domain, we curate **150 tasks** and evaluate them under seven models and five feedback strategies, so 450 is the shared task pool rather than the number of tasks per condition. The target design contains $7 \times 150 \times 5 = 5{,}250$ trajectories per domain; after API and execution-side dropouts, we retain 4,871–4,929 completed trajectories per domain, totaling ∼14,700 12-turn trajectories and ∼176K turn-level observations. Each trajectory yields $\binom{12}{2} = 66$ within-trajectory turn-pair comparisons, or roughly 970K descriptive comparisons, which we use only for trajectory diagnostics rather than as independent task-level observations.

3.1 Task Domains and Dataset Curation

For **ideation**, we sample scientific idea-generation tasks from LiveIdeaBench using stratification to maintain coverage across scientific areas (Ruan et al., 2025). These tasks are useful because there is no single correct answer: refinement can improve originality, feasibility, or clarity, but it can also degrade one axis while improving another. For **coding**, we use DS-1000, a benchmark of Python data-science programming tasks with unit tests (Lai et al., 2022). We curate tasks using a quota-based strategy that mirrors the benchmark's distribution across core Python libraries, so that results are not

Domain	Dataset	Primary outcome signal	Trajectory diagnostics
Ideation	LiveIdeaBench	LLM-assisted originality, feasibility, clarity, and buzzword count	Drift, volatility, lexical novelty, and word-count growth
Coding	DS-1000	Sandboxed unit-test pass/fail at each turn	Drift, volatility, lexical novelty, and lines-of-code growth
Math	Omni-MATH	Ground-truth–anchored final-answer equivalence at each turn	Drift, volatility, lexical novelty, word-count growth, and supplementary reasoning-soundness deltas

Table 1: Summary of the three-domain evaluation design. We use objective correctness signals where available and behavioral metrics for all domains to capture how outputs evolve over the 12-turn trajectory.

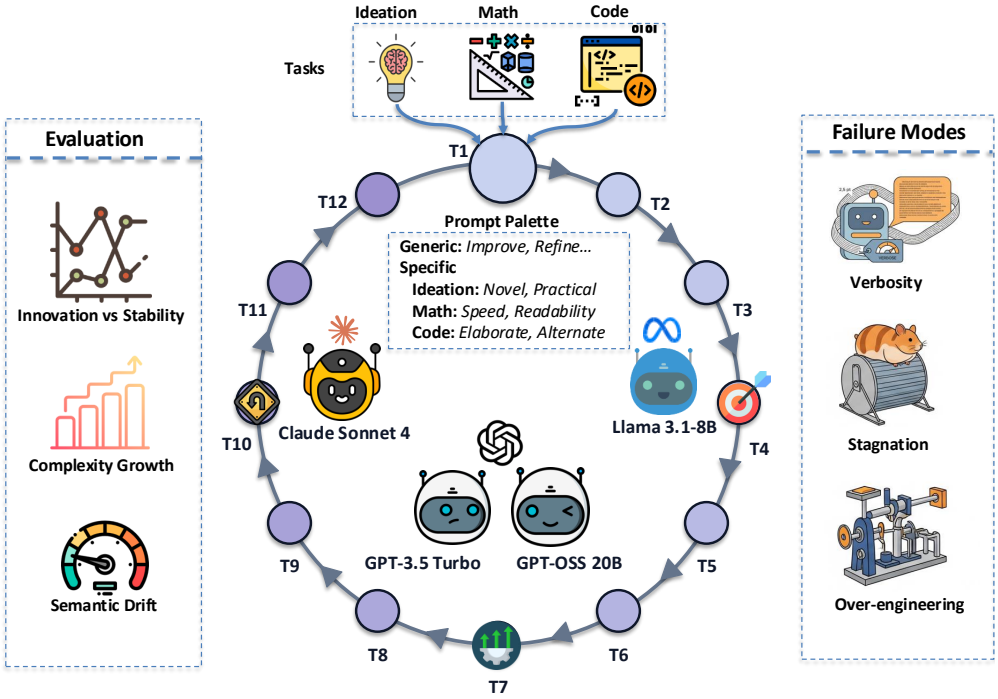

Figure 1: Overview of the experimental framework. Each task is solved once and then refined for 11 additional turns under either vague feedback or domain-specific steering. We score every turn, not only the final output, and use both outcome metrics and behavioral dynamics to characterize refinement trajectories.

dominated by a single package family. For **math**, we use Omni-MATH, filtering for problems with difficulty greater than 7/10 to focus on multi-step reasoning rather than trivial answer recovery (Gao et al., 2024).

3.2 Iterative Refinement Protocol

Each run is a fixed 12-turn trajectory. We choose 12 turns because prior multi-turn benchmarks commonly use histories of roughly ten turns, for example, MultiChallenge and MT-Eval structure multi-turn interactions around comparable depths and we add two extra turns to capture late-stage recovery or collapse (Deshpande et al., 2025; Kwan et al., 2024). At Turn 1, the model receives the original task prompt and produces an initial output. At Turns 2–12, the model receives only the previous response plus a short improvement instruction. The original task is not reinserted after Turn 1. This memoryless protocol is intentionally austere. A system would preserve the original user goal, keep constraints visible, call tools, and test intermediate outputs. We remove those aids to isolate the model's self-referential rewriting behavior. These numbers should be interpreted as a lower-bound stress test: they reveal what can go wrong when iterative prompting is treated as "just ask again," and they motivate why re-grounding, checkpointing, and stop rules are necessary. We also compare iterative refinement to best-of-N parallel sampling as an alternative; results for GPT-3.5-Turbo ($N = 12$) are in Table 3 (Appendix).

3.3 Feedback Strategies

We compare 2 feedback families. The **vague feedback** family tests whether ordinary improvement

language provides a stable refinement signal or instead encourages indefinite rewriting. We use 3 near-synonyms: `v1_improve` ("This {idea/code/solution} is good, improve it."), `v2_better` ("This {idea/code/solution} is good, make it better."), and `v3_refine` ("This {idea/code/solution} is good, refine it."). This design tests whether trajectories depend on a specific verb or on the broader semantics of improvement. The **specific steering** family uses lightweight domain-aware instructions that encode trade-offs. For ideation, `s1_novel` asks for a more novel and surprising idea, while `s2_practical` asks for a more practical and feasible one. For coding, `s1_performance` asks for maximum execution speed, while `s2_maintainability` asks for readability and clarity. For math, `s1_elaboration` asks the model to elaborate each step with more detail, while `s2_exploration` asks for an alternative method or different logical approach. These prompts are intentionally simple rather than highly engineered, allowing us to test whether small, changes in feedback steer the refinement trajectory.

3.4 Models

Model	Access	Max output
GPT-3.5-Turbo (Ye et al., 2023)	API	4,096
Claude Sonnet 4 (Anthropic, 2025)	API	10,000
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	HF/local	10,000
GPT-OSS-20B (OpenAI, 2025)	HF/local	10,000
Gemma 4 26B-A4B-it (Google DeepMind, 2026)	HF/local	10,000
Qwen2.5-Coder-7B-Instruct (Hui et al., 2024)	HF/local	10,000
DeepSeek-Math-7B-Instruct (Shao et al., 2024)	HF/local	10,000

Table 2: Model configuration summary. All models use temperature 0.7; GPT-3.5-Turbo's smaller output budget is treated as a coding-retention confound.

3.5 Outcome Metrics

For code, we extract snippets using a fenced-code parser, a `<code>` fallback, and a heuristic line-based fallback. Extracted code is executed against DS-1000 unit tests in a sandboxed subprocess with a 3s timeout. For math, we use a layered, ground-truth–anchored answer extractor. We first parse all `\boxed{...}` candidates and normalize LaTeX; if this fails, we use relaxed containment and candidate matching from display-math, final-answer, and final-line regions. Only after these fallbacks fail do we assign `no_boxed_found` or `wrong_answer`. This design reduces false negatives for answers that are correct but not formatted exactly as expected.

Let $y_{i,t} \in \{0, 1\}$ indicate whether task i is correct at turn t. We compute four trajectory-level outcome metrics:

$$\text{best@}k_i = \mathbb{1}[\exists\, t \le k : y_{i,t} = 1]\,,$$
$$\text{final@}12_i = y_{i,12}.$$

For aggregate retention and recovery, let

$$E = \{i : \exists\, t \le 12,\ y_{i,t} = 1\},$$
$$D = \{i \in E : y_{i,12} = 0\},$$
$$W = \{i : y_{i,1} = 0\},$$
$$R = \{i \in W : y_{i,12} = 1\}.$$

We then define:

$$\text{drift_rate} = \frac{|D|}{|E|}, \qquad \text{recovery_rate} = \frac{|R|}{|W|}.$$

Here, best@k measures whether a correct answer appears by turn k, while final@12 measures whether the final returned answer is correct. Drift rate captures failures to retain a correct answer once discovered, and recovery rate captures cases that are wrong initially but correct by the final turn.

For ideation, there is no gold label. We use Gemini 2.5 Pro as a rubric-based judge for originality, feasibility, clarity, and buzzword count, following the LLM-as-a-Judge paradigm (Zheng et al., 2023; Google, 2025). Judge scores are used as relative trajectory signals, capturing directional trends across turns rather than absolute quality values (Tian et al., 2025). We validate reliability by re-scoring a stratified 50% sample with Claude Sonnet 4 using the same rubric; both turn-level and trajectory-level agreement are reported in Table 4 (Appendix), with directional agreement reaching 92% for the most policy-critical dimension (feasibility under `s1_novel`). For coding and math, rubric scores are supplementary only; all correctness claims are grounded in unit tests or answer equivalence.

3.6 Behavioral Dynamics Metrics

Outcome metrics say whether a turn is good; behavioral metrics describe how the output changes. Let $V(t)$ be the sentence-embedding vector for the response at turn t. We define **Drift from Origin** as distance from the first response and **Turn-to-Turn**

Iterative Prompt Template (Turns 2–12)

The following is a previous response:
`{previous_response}`
`{improvement_instruction}`

Contents of {improvement_instruction} by Technique

Vague (all tasks, subject depends on task):
V1_IMPROVE: *This {subject} is good, improve it.*
V2_BETTER: *This {subject} is good, make it better.*
V3_REFINE: *This {subject} is good, refine it.*

Subject mapping: IDEAS → *idea*, MATH → *solution*, CODING → *code*.

IDEAS-specific:
S1_NOVEL: *Make this idea more novel and surprising.*
S2_PRACTICAL: *Make this idea more practical and feasible.*

CODING-specific:
S1_PERFORMANCE: *Refactor the previous code snippet to maximize execution speed.*
S2_MAINTAINABILITY: *Refactor the previous code snippet to maximize readability and clarity.*

MATH-specific:
S1_ELABORATION: *This is previous response, now elaborate on each step with more detail.*
S2_EXPLORATION: *Provide an alternative method or a different logical approach to the one used.*

Figure 2: Prompt structure for Turns 2–12. The model receives its previous response and one improvement instruction; the original task is not reinserted after Turn 1.

Volatility as local change between adjacent turns:

$$\text{Drift}(t) = 1 - \frac{V(1) \cdot V(t)}{\|V(1)\|\|V(t)\|},$$

$$\text{Volatility}(t) = 1 - \frac{V(t-1) \cdot V(t)}{\|V(t-1)\|\|V(t)\|}.$$

We use `Qwen3-Embedding-0.6B` to compute embeddings, selected for its balance of efficiency and strong MTEB performance (Zhang et al., 2025; Muennighoff et al., 2023). We also compute **Lexical Novelty**, the share of bi-grams and tri-grams in the current turn that did not appear in earlier turns (Li et al., 2016). Finally, **Growth Factor** measures output expansion: word count normalized by T1 for ideation and math, and lines of code normalized by T1 for coding (Laban et al., 2025). Together these metrics distinguish exploration, repetition, bloat, and fixation.

4 Results

We summarize the main trajectory-level findings in text and provide the full visual plots in the appendix. Turn-wise correctness for math and coding is reported in Appendix Figure 3; best@k versus final@12 comparisons in Appendix Figure 4; behavioral dynamics across domains in Appendix Figure 12; trajectory categories in Appendix Figure 5; and LLM-judge trajectories in Appendix Figure 13. Additional model-wise, technique-wise, and heatmap analyses are reported in Appendix Figures 19–24 and Figures 16–18.

4.1 Coding: Early Success, Then Degenerative Refinement

Coding exhibits the clearest case of destructive refinement. Across all seven models and five feedback techniques, mean correctness falls from **40.7% at T1 to 2.5% at T12** (−38.2 pp). The decline is not limited to weaker systems: Claude Sonnet 4 drops from 74.4% to 2.0%, Gemma 4 26B-A4B-it from 55.8% to 0.0%, and GPT-OSS-20B from 55.2% to 0.0%. GPT-3.5-Turbo is the only model retaining non-trivial T12 correctness (12.8%), largely because `s2_maintainability` remains partially stable.

The central failure is not inability to ever produce correct code, but failure to retain it. Across coding cells, mean **best@12 is 44.9%** while **final@12 is 2.4%**, yielding a 42.5 pp discovery–retention gap. Large gaps appear even for strong models and sensible prompts: Claude under `s2_maintainability` moves from 80.5% best@12 to 1.3% final@12, Gemma under `s1_performance` from 57.7% to 0.0%, and GPT-OSS-20B under `v3_refine` from 59.7% to 0.0%. Drift rates are correspondingly extreme: nearly all completed model–technique

coding cells have drift_rate above 70%, and many reach 100%, meaning every task that was ever correct is wrong by T12. The behavioral metrics explain this collapse as fixation plus bloat. Coding volatility peaks at T2 (0.229), falls by T5 (0.137), and reaches 0.127 at T12, suggesting that models lock into a solution path early. Lexical novelty falls to 0.410 by T12, while growth reaches 9.9× T1. Thus, code becomes much longer without becoming meaningfully more exploratory. T12 drift is 0.468, lower than ideation despite far larger output growth, supporting a "fixation and bloat" interpretation: the model repeatedly edits around the same program idea, adding wrappers, edge cases, or alternative formulations that often break correctness. Prompt steering changes style more than correctness. The LLM-judge rubric indicates that `s2_maintainability` best preserves readability (8.07→6.64), while vague schedules damage pragmatism and readability more strongly. However, these style gains do not rescue unit-test correctness. The practical implication is direct: iterative coding systems should test after each turn, checkpoint passing solutions, and return the best verified code rather than the latest revision.

4.2 Math: Anchoring, Late Recovery, and Model-Specific Depth

Math shows a different refinement regime. Mean correctness declines only modestly overall, from 44.1% at T1 to 34.2% at T12 (−9.9 pp), but this aggregate hides substantial model- and prompt-specific variation. GPT-OSS-20B is the clearest beneficiary of depth: under `s1_elaboration`, it rises from 71.3% to 86.0%, and under `v3_refine`, from 68.8% to 84.6%. Llama-3.1-8B also improves under several schedules, including `v2_better` (24.7%→34.7%) and `v3_refine` (18.0%→29.3%). By contrast, Claude, Qwen-Coder, DeepSeek-Math, GPT-3.5-Turbo, and Gemma often lose correctness over turns, even under prompts that help GPT-OSS-20B. Math also exhibits retention failure, though less severely than coding. Across math cells, **best@12 is 58.1%** and **final@12 is 35.0%**, a 23.1 pp discovery–retention gap. Individual cells show larger losses: Claude under `s1_elaboration` has best@12=60.0% but final@12=15.3%; Llama under `v2_better` loses 38.0 pp between best@12 and final@12; and DeepSeek-Math under `s2_exploration` loses 40.0 pp. Thus, even in a domain with objective final answers, models can solve a problem somewhere in the trajectory and then refine themselves away from the correct answer.

The domain fingerprint is logical anchoring. Math has the lowest T12 drift (0.375), the lowest T12 volatility (0.108), and the smallest growth (3.2×) of the three domains. Lexical novelty falls quickly, averaging 0.439 at T12, with some cells rewriting nearly the same proof for multiple turns. This stability is double-edged: if the initial proof path is correct or nearly correct, elaboration can reinforce it; if the path is wrong, the same anchoring can preserve the error with increasing confidence.

The strongest schedules are those that preserve structure while adding useful detail. `s1_elaboration` works well for GPT-OSS-20B because it deepens an already strong proof prior, and `v3_refine` often helps because it is conservative rather than disruptive. We therefore interpret GPT-OSS-20B's gains as *late-turn improvement from a high T1 baseline*, not as evidence of unconstrained self-correction from failure. In contrast, `s2_exploration` asks the model to replace the proof path and often loses intermediate structure, especially for weaker math models. This suggests a practical math policy: check final answers at every turn, keep the best verified answer, and prefer conservative elaboration over alternative-method generation unless the current path is wrong.

4.3 Ideation: Controlled Steering Versus Rhetorical Inflation

Ideation is the most semantically mobile domain. Under `s1_novel`, Claude, Gemma, and GPT-OSS-20B reach high final-turn drift scores (0.727, 0.859, and 0.655), indicating large movement away from the initial idea when explicitly asked for novelty. Llama-3.1-8B remains more anchored under the same prompt (0.382), while Qwen2.5-Coder-7B lies between these extremes. Vague word choice also matters: `v3_refine` is consistently more conservative than `v2_better`, showing that small differences in feedback language shape the exploration radius. Lexical novelty separates models more cleanly than length, although we do not treat it as a direct measure of creativity. Under `s1_novel`, Claude, GPT-OSS-20B, and Gemma retain high T12 lexical novelty (0.835, 0.823, and 0.916), whereas Llama and Qwen-Coder collapse toward repetition (0.080 and 0.268). This contrast is not explained by output length: Llama's ideation outputs grow to 15.8× T1 under `s1_novel` and 13.0× under `v2_better`, yet show the lowest nov-

elty. Length is therefore not a reliable proxy for exploration.

Specific steering moves judge-rated quality along the intended axis. Under `s1_novel`, Claude's judge-rated originality increases from 6.40 to 8.89, while cohort-mean feasibility drops from 6.68 to 1.69, showing the cost of asking for surprise. Under `s2_practical`, Claude's judge-rated feasibility rises from 3.54 to 9.96 and clarity from 7.62 to 9.93; GPT-OSS-20B also improves feasibility (5.24→8.08) and clarity (7.40→8.79). In contrast, `v2_better` reduces mean originality from 5.36 to 3.06 and feasibility from 5.97 to 3.92. Buzzword inflation captures the cost of unguided ideation. Cohort-mean buzzword count grows from 1.42 to 35.98 under `s1_novel`, compared with only 1.89 to 4.50 under `s2_practical`; vague schedules fall between these extremes. Overall, novelty-seeking and vague improvement can produce increasingly ornamental ideas, while practical steering keeps language grounded. The best ideation policy is therefore staged: first widen the space intentionally, then switch to feasibility and clarity grounding.

5 Discussion

Iteration is domain-conditioned. The same instruction "make it better" maps to different operations across domains. In ideation, it often broadens the concept or makes it sound more ambitious; in coding, it adds wrappers, edge cases, or scaffolding even when the original program was already correct; and in math, it tends to preserve and elaborate the existing proof path. These behaviors appear consistently across drift, volatility, novelty, growth, correctness, and judge-rated trajectories.

The best turn is often not the final turn. The largest actionable result is the discovery–retention gap. Coding best@12 exceeds final@12 by 42.5 pp, and math shows a 23.1 pp gap. When intrinsic verifiers exist, these gaps are directly exploitable: unit tests can checkpoint code, and answer-equivalence checks can preserve correct math answers. A system that always returns the latest refinement wastes correct intermediate outputs.

Steering should be staged, not repeated blindly. Our results argue against one monolithic refinement loop. Ideas need divergence followed by grounding; math needs conservative elaboration plus answer retention; and code needs early testing, rollback, or restart rather than rewriting. This suggests refinement policies that separate generation, verification, targeted revision, and stopping.

Capability threshold and the GPT-3.5 anomaly. GPT-3.5-Turbo shows an anomaly: it retains higher T12 coding correctness than larger models, yet its math correctness declines sharply. We hypothesize that this reflects the interaction between output budget and reasoning capacity: the shorter 4,096-token budget may limit coding bloat, while weaker reasoning makes repeated math elaboration unstable. We treat this as a domain-asymmetric capability-threshold effect, not evidence that shorter-output models are better refiners.

Memoryless refinement is a stress test, not a product prescription. Our setup deliberately removes re-grounding and external tools. This magnifies the risks of iterative prompting, but it also isolates the cost of ungrounded self-referential rewriting. A model that collapses under memoryless refinement may perform better when the original task is preserved, tools are used, and intermediate outputs are checkpointed. The broader conclusion is that iteration needs governance: explicit goals, domain-specific signals, and stopping conditions.

6 Conclusion

We studied iterative LLM prompting as a trajectory-level phenomenon across ideation, coding, and math. Across seven models, 450 tasks, five feedback techniques, and ∼14,700 12-turn trajectories, the central result is that another turn is not inherently better. Coding collapses under sustained refinement: correctness falls from 40.7% to 2.5%, and the best@12–final@12 gap is 42.5 pp. Math is mixed: GPT-OSS-20B gains 14.7 pp under elaboration, but several models drift away from correct answers. Ideation benefits from explicit steering but degrades under vague improvement through buzzword inflation and feasibility loss. The practical lesson is to replace open-ended refinement with controlled refinement policies. For code, test early and return the best passing turn. For math, preserve proof structure and checkpoint answer correctness. For ideas, separate novelty generation from practical grounding. More generally, iterative prompting should be paired with domain-aware steering, intrinsic verification, and stop/switch rules. With these controls, iteration becomes a useful process; without them, it can silently convert good intermediate outputs into worse final ones.

Limitations

Our study covers seven representative models, three domains, 450 tasks, and ∼176K turn-level observations, but it does not exhaust the space of models, prompts, or tasks. Larger frontier reasoning models, Mixture-of-Experts systems, additional 70B+ open-weight checkpoints, and broader task pools may reveal further regimes. The feedback set is intentionally compact: three vague prompts and two domain-specific prompts per domain. More systematic prompt search could identify stronger steering schedules or more robust stop policies.

Our evaluation also has measurement limitations. Coding and math have objective anchors, but ideation requires LLM-assisted judgment because no gold labels exist. We mitigate this by reporting relative, turn-wise trends, separating ideation claims from objective correctness claims, and validating the Gemini judge against Claude Sonnet 4 on a stratified 50% sample (Table 4). Turn-level agreement is moderate ($r = 0.524$, $\kappa = 0.392$), but trajectory-level directional agreement is substantially stronger, 66% overall and 92% for the paper's most policy-critical claim (all $p \approx 0$), supporting the main trajectory-level trends. A full human audit would further strengthen originality and clarity claims. Math answer equivalence is also imperfect, especially for complex symbolic answers, although our layered extractor reduces strict-format false negatives. Finally, the memoryless protocol is a lower-bound stress test rather than a faithful simulation of well-engineered chat systems. Production systems can re-ground the task, use tools, and preserve constraints; our results show why those safeguards matter.

References

Anthropic. 2025. System card: Claude opus 4 & claude sonnet 4. https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47.pdf. Accessed: YYYY-MM-DD.

Hikaru Asano, Tadashi Kozuno, and Yukino Baba. 2025. Self iterative label refinement via robust unlabeled learning. *Preprint*, arXiv:2502.12565.

Daniel Barzilai and Ohad Shamir. 2025. When models don't collapse: On the consistency of iterative mle. *Preprint*, arXiv:2505.19046.

Justin Chih-Yao Chen, Archiki Prasad, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. 2024. Magicore: Multi-agent, iterative, coarse-to-fine refinement for reasoning. *Preprint*, arXiv:2409.12147.

Maximillian Chen, Ruoxi Sun, Tomas Pfister, and Sercan O Arik. 2025. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. In *The Thirteenth International Conference on Learning Representations*.

Prateek Chhikara. 2025. Mind the confidence gap: Overconfidence, calibration, and distractor effects in large language models. *Preprint*, arXiv:2502.11028.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E. Primack, Summer Yue, and Chen Xing. 2025. MultiChallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702, Vienna, Austria. Association for Computational Linguistics.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *Preprint*, arXiv:2410.07985.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Tomasz Korbak, Henry Sleight, Rajashree Agrawal, John Hughes, Dhruv Bhandarkar Pai, Andrey Gromov, Dan Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. 2024. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. In *First Conference on Language Modeling*.

Google. 2025. Gemini 2.5 pro: Model card. Last updated June 27, 2025.

Google DeepMind. 2026. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4. Accessed: 2026-05-03.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Shengyue Guan, Haoyi Xiong, Jindong Wang, Jiang Bian, Bin Zhu, and Jian guang Lou. 2025. Evaluating llm-based agents for multi-turn conversations: A survey. *Preprint*, arXiv:2503.22458.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang,

Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38.

Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju. 2024. Understanding the effects of iterative prompting on truthfulness. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org. Event-place: Vienna, Austria.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. MT-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20153–20177, Miami, Florida, USA. Association for Computational Linguistics.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. LLMs Get Lost In Multi-Turn Conversation. *arXiv preprint*. ArXiv:2505.06120.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2022. Ds-1000: A natural and reliable benchmark for data science code generation. *Preprint*, arXiv:2211.11501.

Jixuan Leng, Chengsong Huang, Banghua Zhu, and Jiaxin Huang. 2025. Taming overconfidence in llms: Reward calibration in rlhf. *Preprint*, arXiv:2410.09724.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Youquan Li, Miao Zheng, Fan Yang, Guosheng Dong, Bin Cui, Weipeng Chen, Zenan Zhou, and Wentao Zhang. 2025a. FB-bench: A fine-grained multi-task benchmark for evaluating LLMs' responsiveness to human feedback. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9271–9291, Suzhou, China. Association for Computational Linguistics.

Yubo Li, Xiaobin Shen, Xinyu Yao, Xueying Ding, Yidi Miao, Ramayya Krishnan, and Rema Padman. 2025b. Beyond Single-Turn: A Survey on Multi-Turn Interactions with Large Language Models. *arXiv preprint*. ArXiv:2504.04717.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. SELF-REFINE: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc. Event-place: New Orleans, LA, USA.

Amr Mohamed, Mingmeng Geng, Michalis Vazirgiannis, and Guokan Shang. 2025. LLM as a Broken Telephone: Iterative Generation Distorts Information. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7493–7509, Vienna, Austria. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card. Technical report, OpenAI. Model card, version posted Aug 5, 2025.

Kai Ruan, Xuan Wang, Jixiang Hong, Peng Wang, Yang Liu, and Hao Sun. 2025. Liveideabench: Evaluating llms' divergent thinking for scientific idea generation with minimal context. *Preprint*, arXiv:2412.17596.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint*. ArXiv:2303.11366.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2024. AI Models Collapse When Trained on Recursively Generated Data. *Nature*, 631(8022):755–759.

Zailong Tian, Zhuoheng Han, Yanzhe Chen, Haozhe Xu, Xi Yang, Richeng Xuan, Houfeng Wang, and Lizi Liao. 2025. Overconfidence in llm-as-a-judge: Diagnosis and confidence-driven solution. *Preprint*, arXiv:2508.06225.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024. Pride and prejudice: LLM amplifies self-bias in self-refinement.

In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15474–15492, Bangkok, Thailand. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*. arXiv.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *Preprint*, arXiv:2303.10420.

Yongcheng Zeng, Xinyu Cui, Xuanfa Jin, Guoqing Liu, Zexu Sun, Dong Li, Ning Yang, Jianye Hao, Haifeng Zhang, and Jun Wang. 2025. Evolving llms' self-refinement capability via iterative preference optimization. *Preprint*, arXiv:2502.05605.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Datasets and Benchmarks Track (NeurIPS 2023).

Appendix

This appendix provides supplementary figures, tables, and per-domain detail for the main results. Figures cover outcome and discovery–retention behavior (Figures 3–7), feedback strategy comparisons (Figures 8–11), behavioral dynamics (Figures 12–15), per-domain turn-level heatmaps (Figures 16–18), and model-wise and technique-wise turn curves (Figures 19–24). We also report per-domain summary statistics, a best-of-N comparison against iterative refinement (Table 3), and Gemini–Claude judge agreement for ideation scoring (Table 4). All three domains; ideation, coding, and math, are represented throughout.

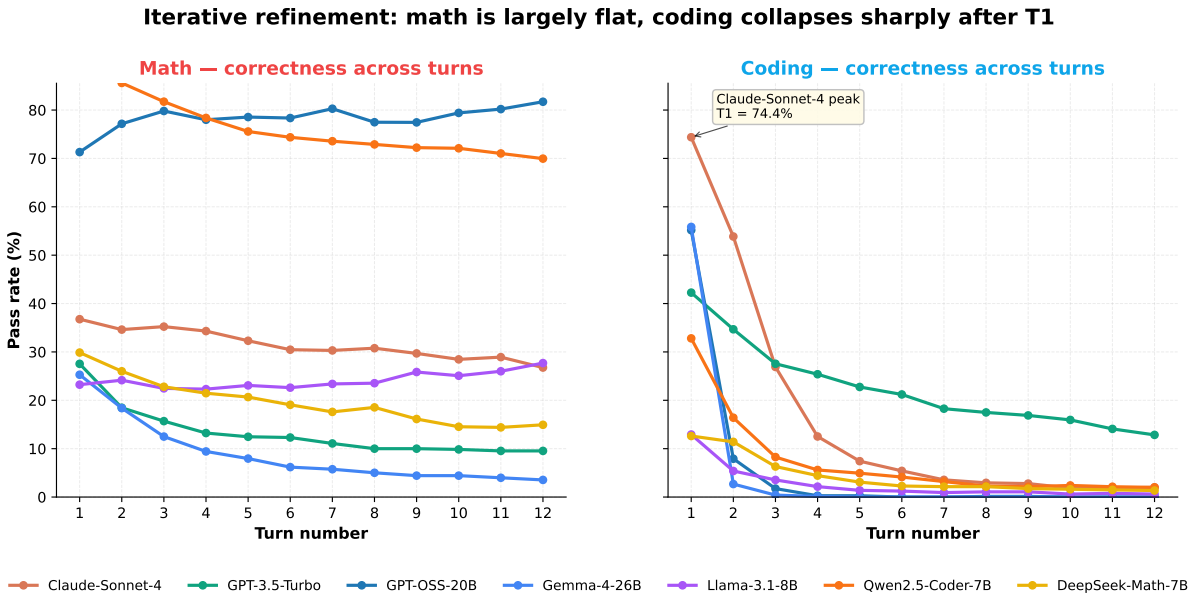

Figure 3: Turn-wise correctness across all models for math (left) and coding (right). Math shows model-dependent behavior, with GPT-OSS-20B improving under some schedules and other models declining modestly; coding collapses from 40.7% at T1 to 2.5% at T12.

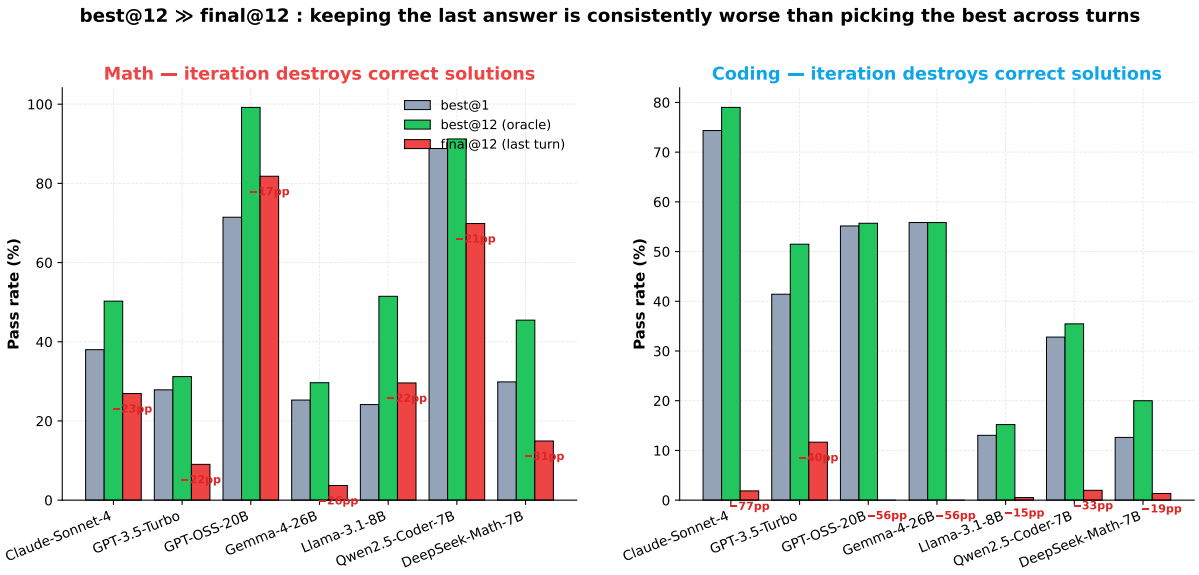

Figure 4: best@1, best@12, and final@12 grouped per (model, domain). best@12 represents the best observed turn in a trajectory, while final@12 represents the answer returned after all refinement turns. Red `-N` pp annotations show the pass rate lost by running to T12 instead of stopping at the best observed turn, up to −59.7 pp in coding for GPT-OSS-20B under `v3_refine` and −44.7 pp in math for Claude under `s1_elaboration`.

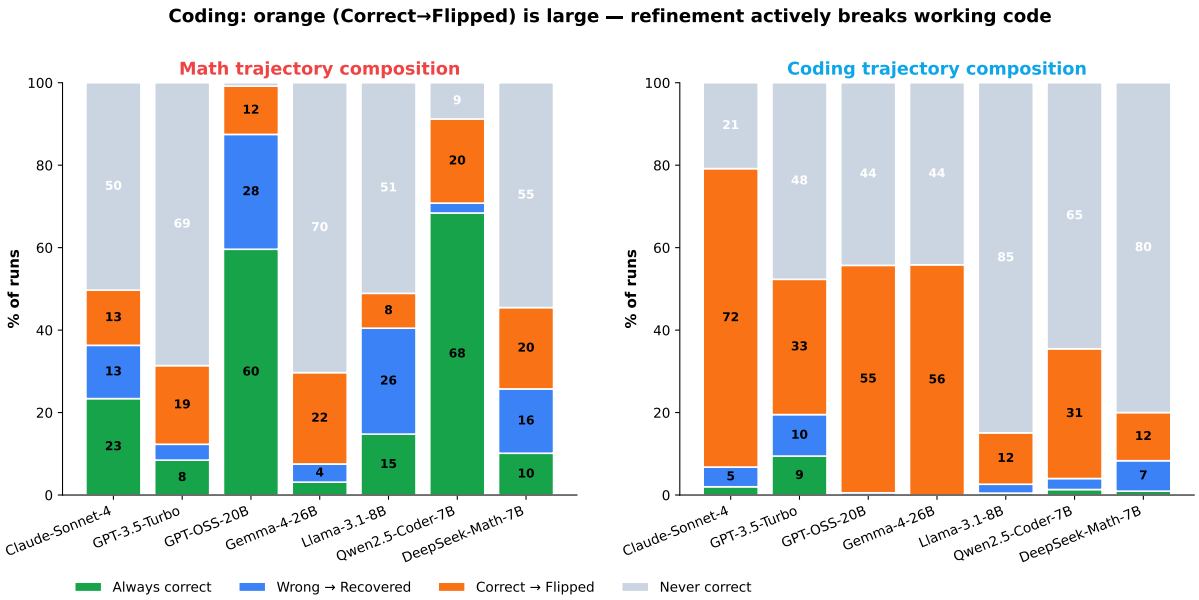

Figure 5: Trajectory category composition per (model, domain): always-correct / wrong-then-recovered / correct-then-flipped / never-correct. Coding is dominated by "correct-then-flipped" for Claude (72.4%) and "never-correct" for the open-weight baselines (Llama-3.1-8B 84.9%, DeepSeek-Math-7B 80.0%). Math is more mixed, with GPT-OSS-20B at 59.6% "always-correct" and 27.9% "wrong-then-recovered" (87.5% combined ending correct).

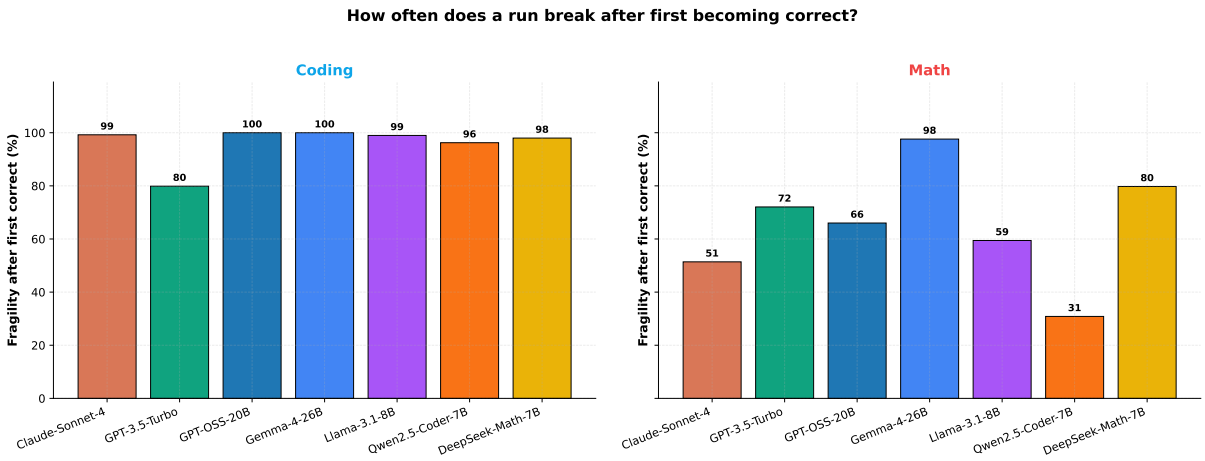

Figure 6: Fragility-after-first-correct: among tasks ever solved, the fraction still correct at T12. Coding fragility is near zero for every model, meaning correct solutions are rarely retained once overwritten. Math fragility varies more widely: GPT-OSS-20B retains 87% of correct solutions, while Claude under `s1_elaboration` retains only 25%.

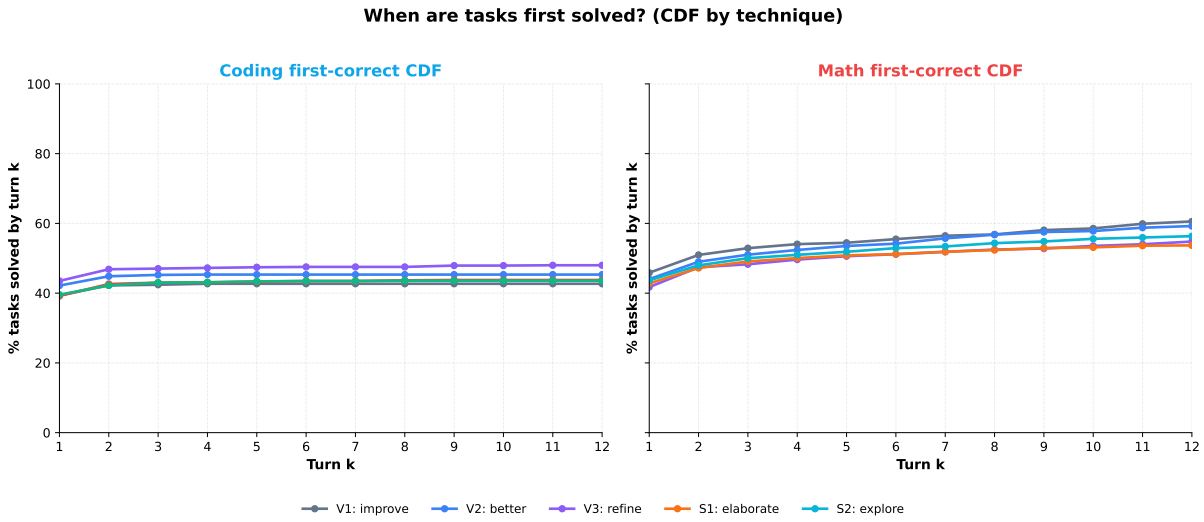

Figure 7: CDF of the first turn at which each task is solved, per domain. In coding, >95% of all eventual correct answers appear by T3; in math, the curve continues to rise through later turns. This asymmetry motivates different stopping policies: early checkpointing in coding and more cautious continued iteration in math.

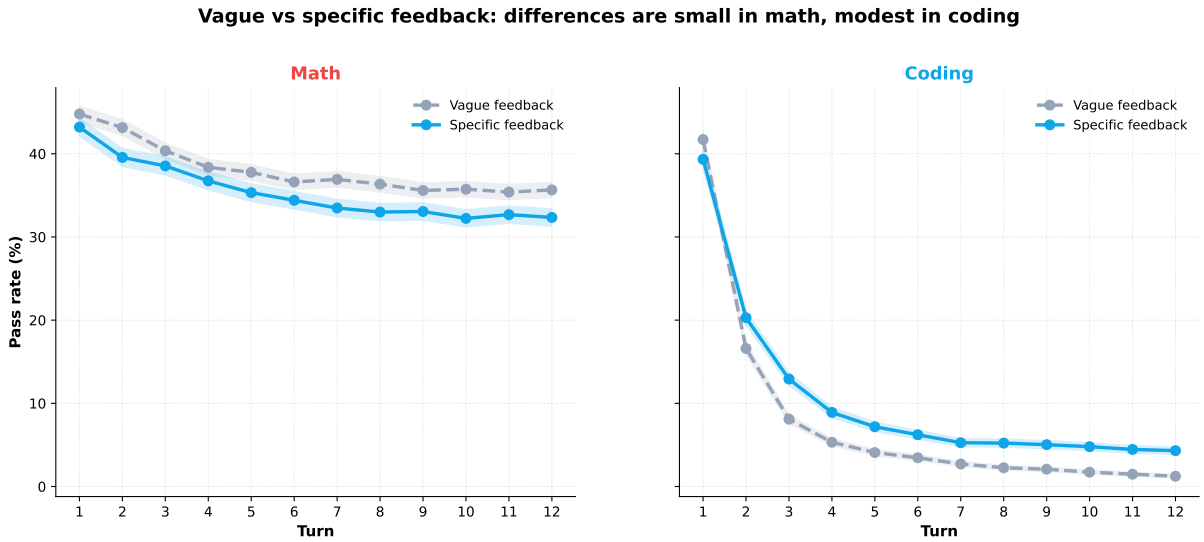

Figure 8: Vague (V1/V2/V3) vs. specific (S1/S2) feedback aggregated by domain. In ideation, specific prompts improve the targeted judge-rated dimension more strongly than vague prompts. In coding, specific schedules retain a small but consistent correctness margin. In math, the depth gain is concentrated in `s1_elaboration` for the strongest math model.

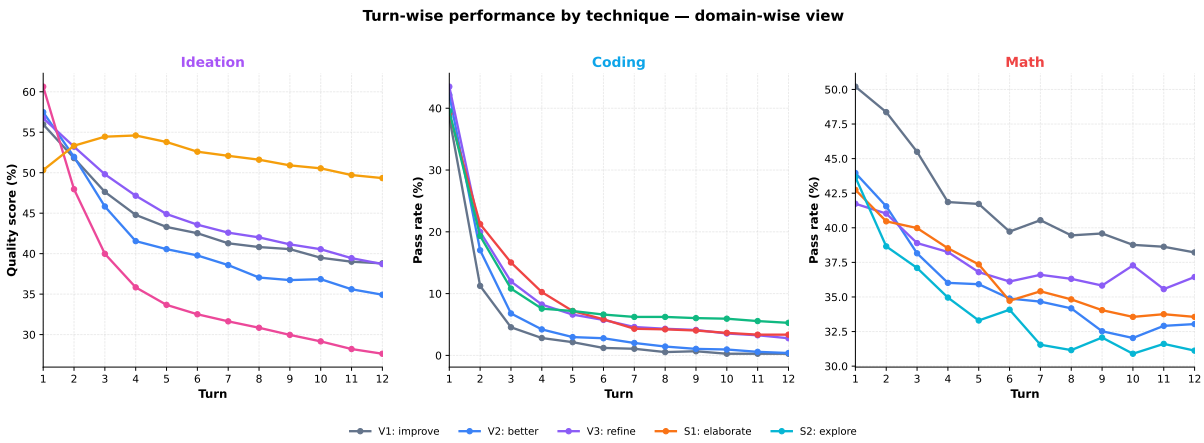

Figure 9: Turn-wise performance by technique—domain-wise view covering all three domains side by side. Ideation uses Gemini-judge composite quality (%); coding and math use unit-test/answer-equivalence pass rate. The divergence between specific and vague techniques is most pronounced in ideation and most uniform in coding.

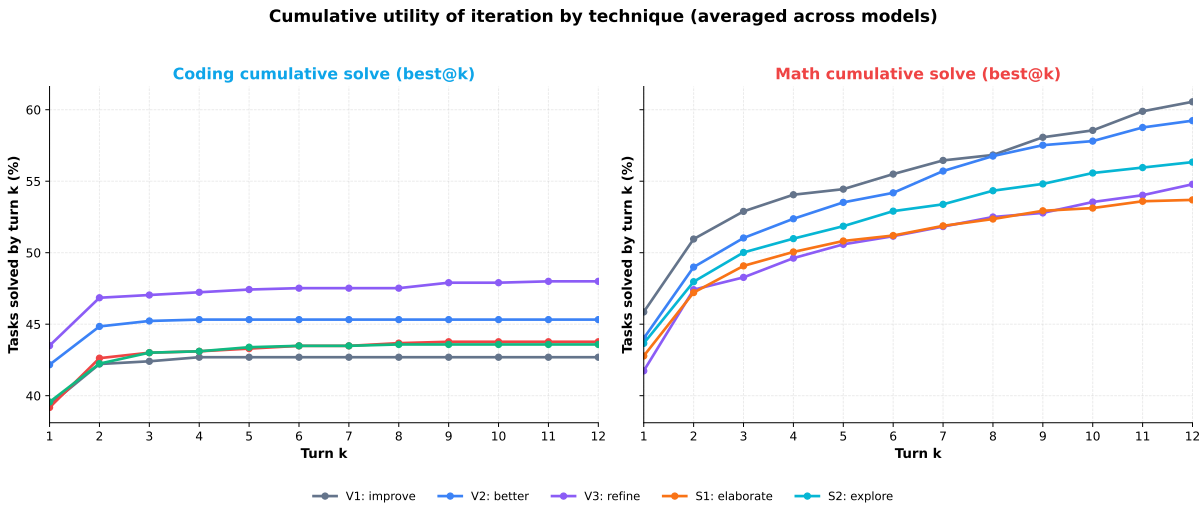

Figure 10: Cumulative solve rate by technique (best@k as a function of k). In coding, all techniques plateau by T3–T4, indicating that late turns add few new correct solutions. In math, `s1_elaboration` for GPT-OSS-20B continues rising through later turns.

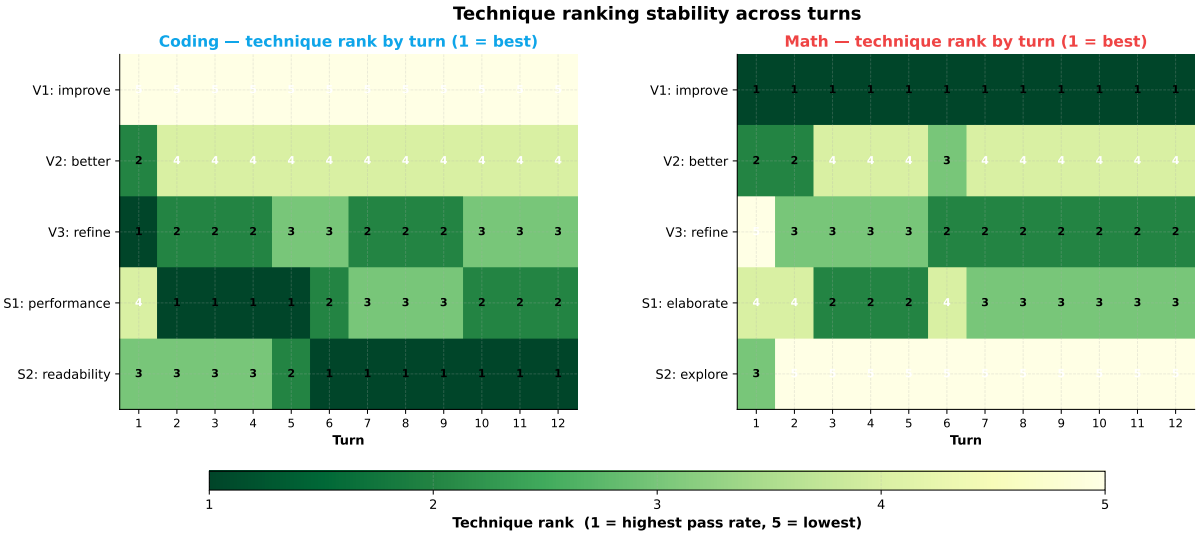

Figure 11: Per-(model, domain) ranking of techniques by best@12. `s1_elaboration` is strongest in math for GPT-OSS-20B and Qwen2.5-Coder-7B; `s2_maintainability` is the safest coding schedule for Claude and GPT-OSS-20B; in ideation, `s1_novel` and `s2_practical` rank highly for frontier models because they directly target the evaluated quality axes.

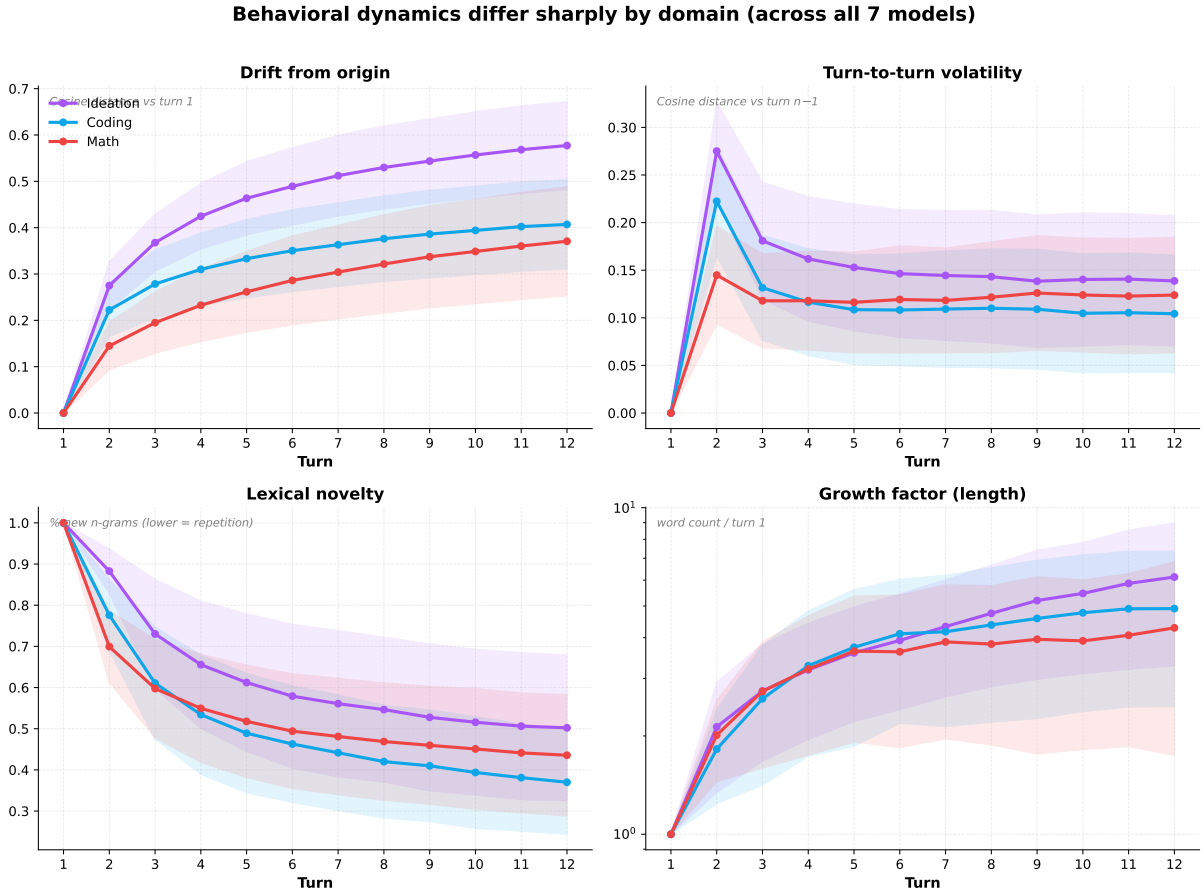

Figure 12: **Behavioral dynamics across all three domains.** Drift and volatility use `Qwen3-Embedding-0.6B` cosine distance; lexical novelty is the share of new bi-/tri-grams; growth is normalized word count or lines of code. Ideation drifts most (T12 drift 0.534), coding grows most (T12 growth 9.9×), and math anchors hardest (T12 drift 0.375).

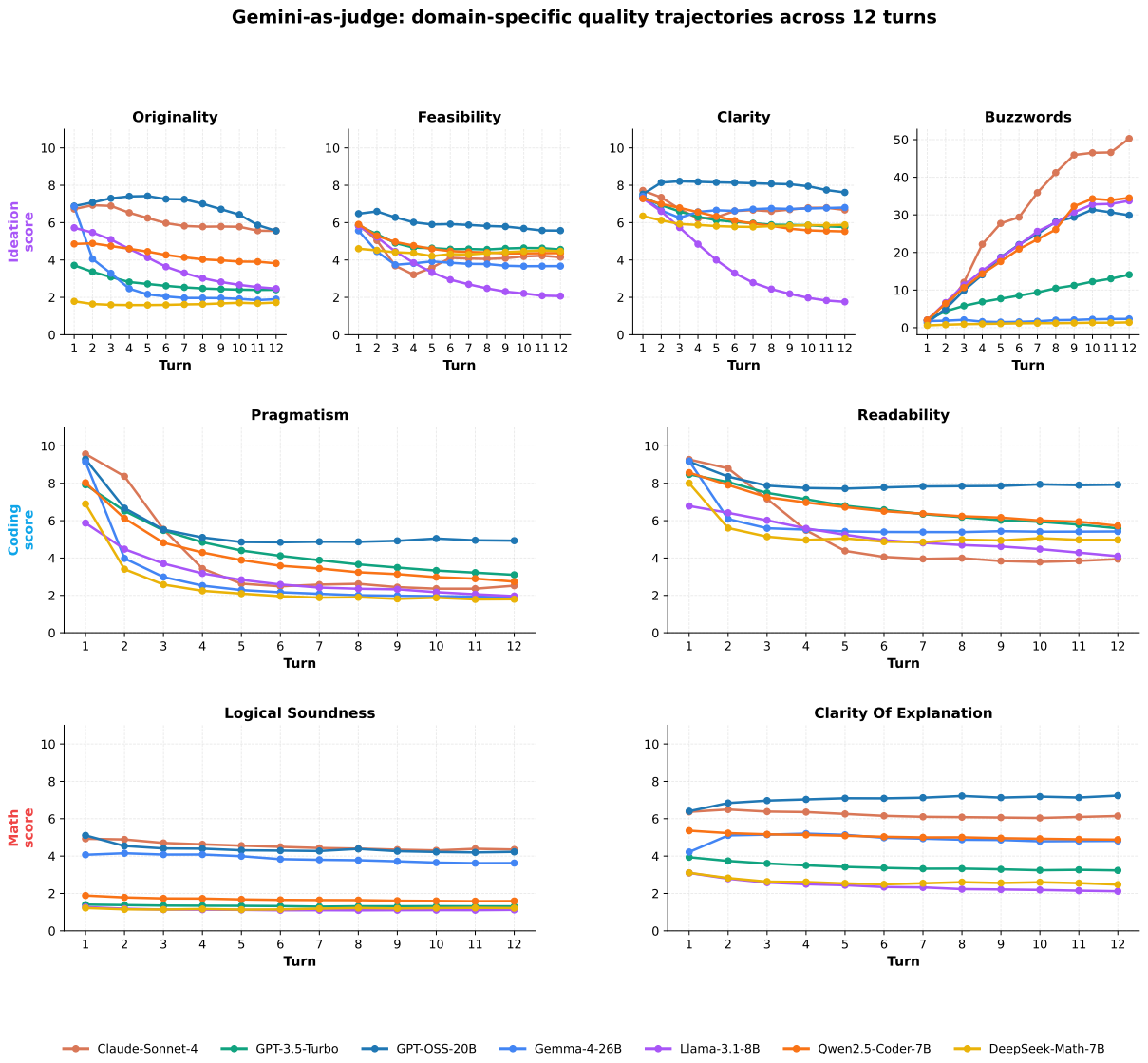

Figure 13: Gemini-2.5-Pro turn-by-turn rubric scores across all (model, technique) cells. **Ideas**: originality / feasibility / clarity / buzzword count. **Coding**: pragmatism / readability. **Math**: logical soundness / clarity of explanation. Specific steering produces sharper directional change in judge-rated ideation dimensions; coding rubric scores erode alongside correctness; math rubric scores remain more stable for stronger models.

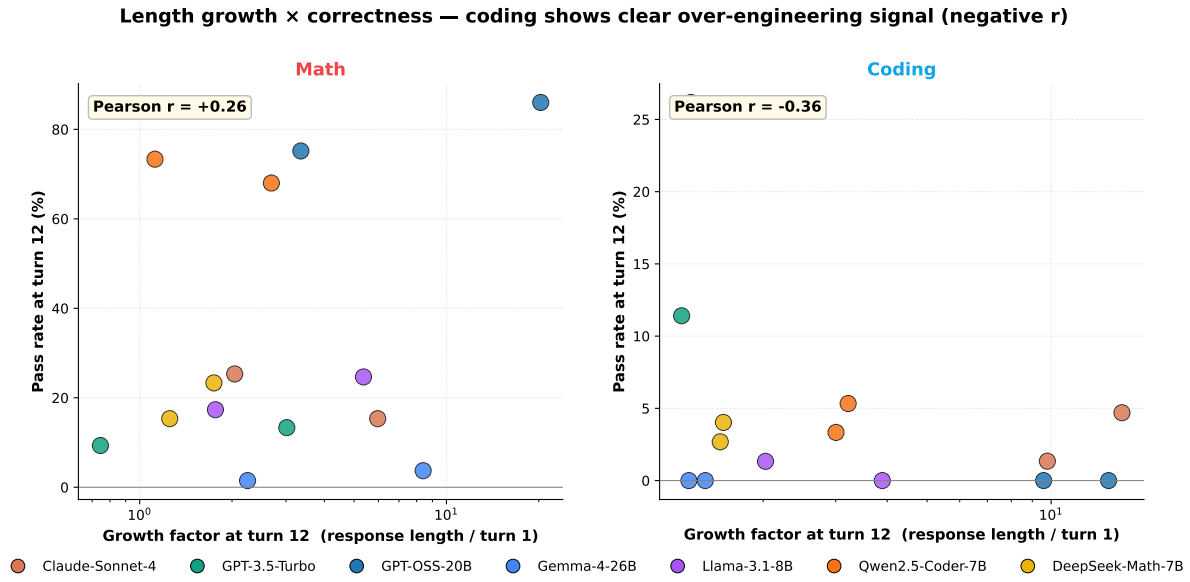

Figure 14: Growth factor × pass rate at T12. Each marker is a (model, technique) cell. Pearson r on log-x: **coding** $r = -0.34$ and **math** $r = +0.08$. This pattern supports the "length growth = over-engineering" interpretation in coding, while showing that length growth alone does not explain math outcomes.

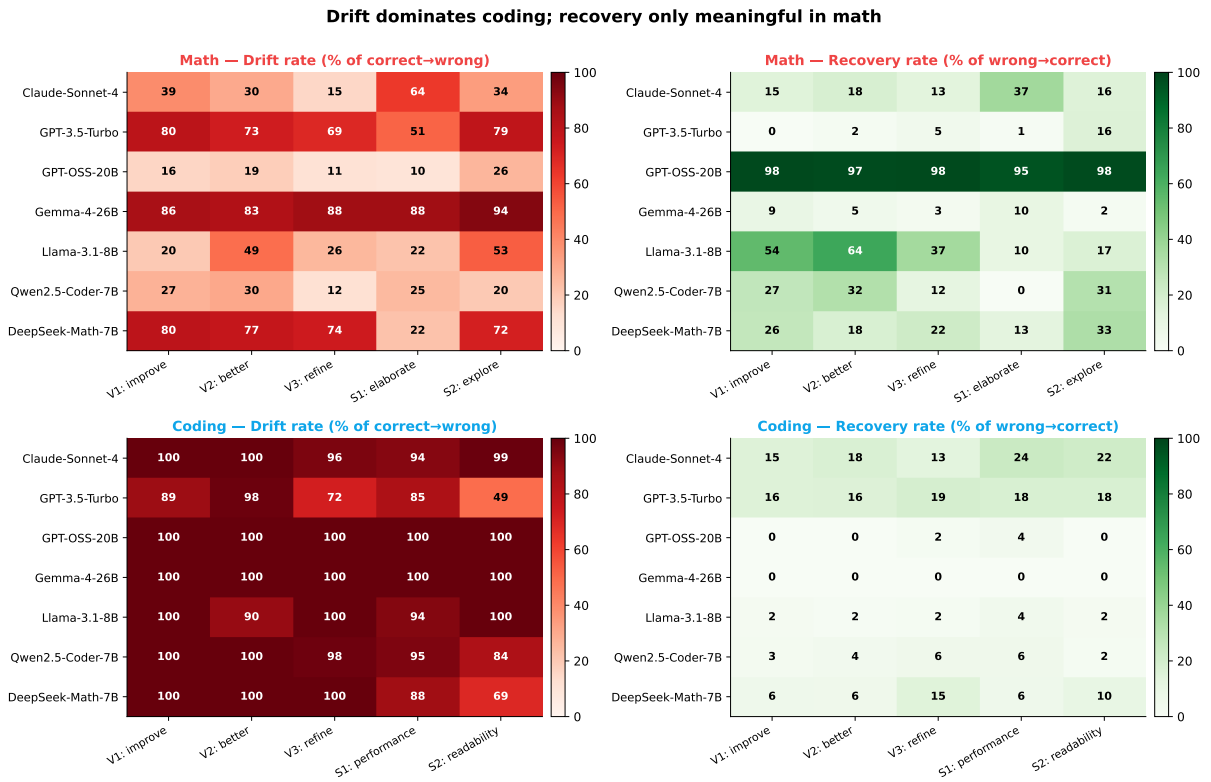

Figure 15: Drift (red) and recovery (green) rates per (model, technique), with stacked rows for math and coding. Coding drift saturates above 90% in nearly every cell, while math recovery is high mainly for GPT-OSS-20B (95–97%) and Llama-3.1-8B under `v2_better` (63.7%).

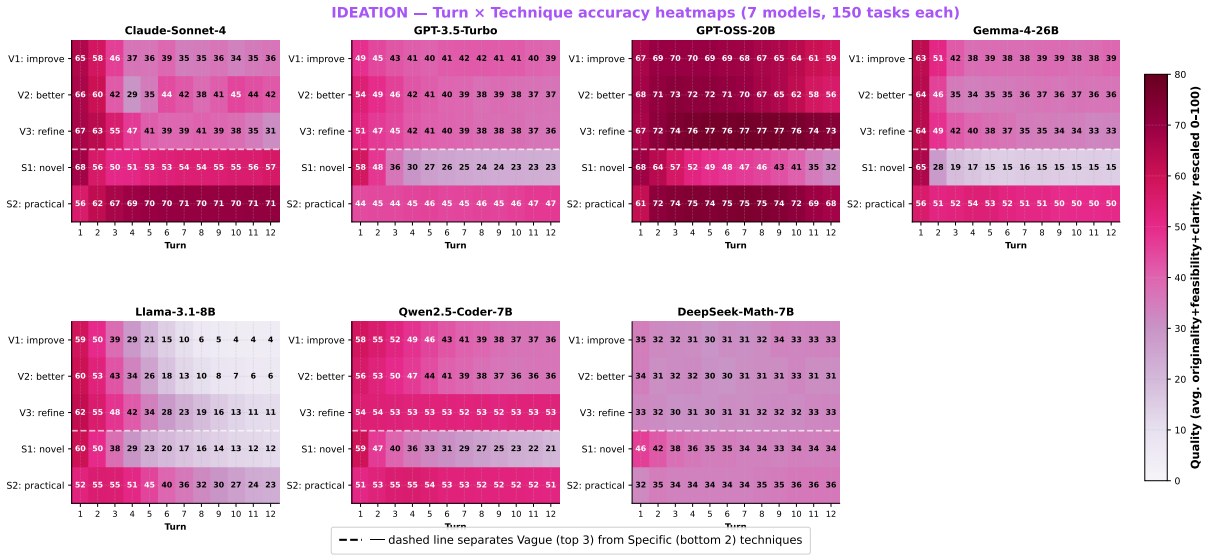

Figure 16: Per-(model, technique, turn) Gemini-judge-rated quality heatmap for **ideation**. Novelty-seeking prompts (`s1_novel`) drive sustained upward trends in judge-rated originality for frontier models; vague prompts produce a brief early rise followed by plateau or decline. Feasibility-seeking prompts (`s2_practical`) show consistent improvement in judge-rated feasibility across turns.

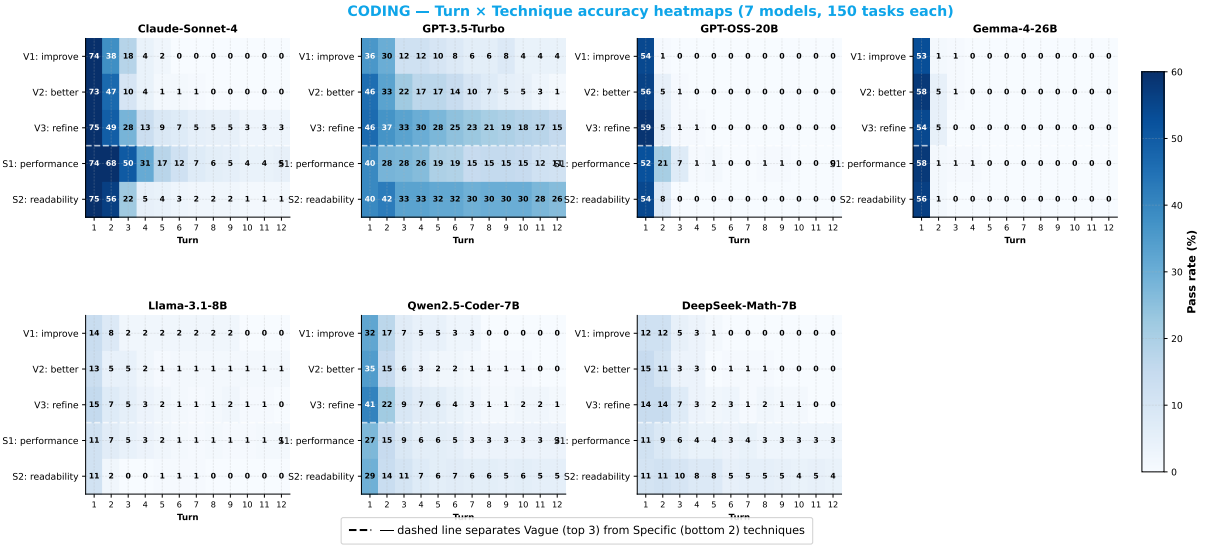

Figure 17: Per-(model, technique, turn) correctness heatmap for **coding**. Each cell shows unit-test pass rate at each of the 12 turns. The universal collapse from bright (high correctness) columns at T1 to dark (near-zero) columns by T4–T6 visualizes the degenerative refinement pattern described in the main Results section.

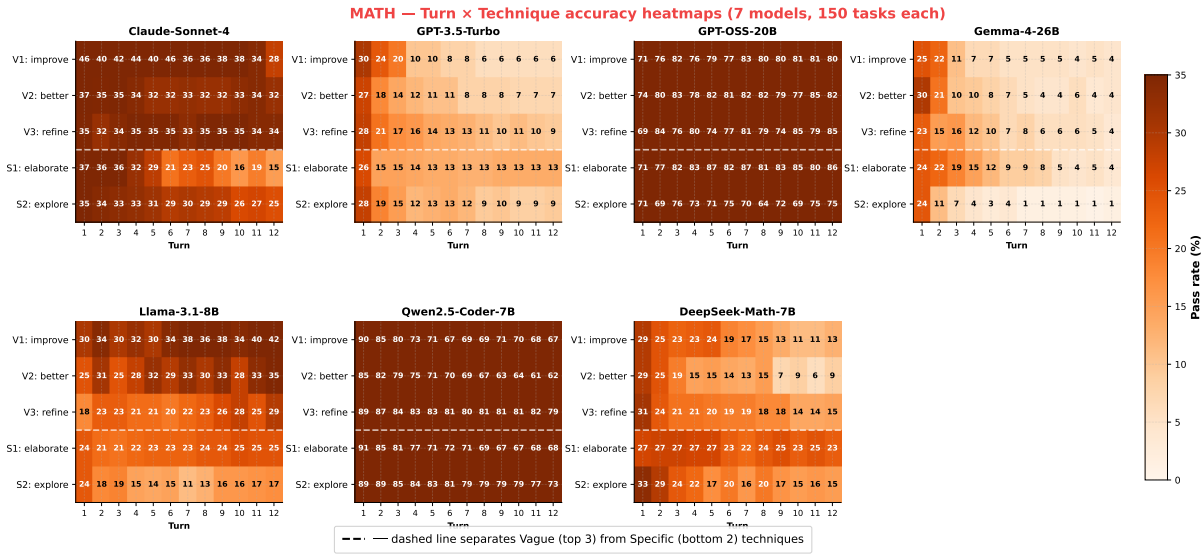

Figure 18: Per-(model, technique, turn) correctness heatmap for **math**. GPT-OSS-20B maintains bright columns throughout, indicating sustained high correctness across turns; other models show mixed patterns of anchoring, modest decline, or recovery. The contrast with coding is stark: math permits late-turn improvement for some strong model–prompt combinations.

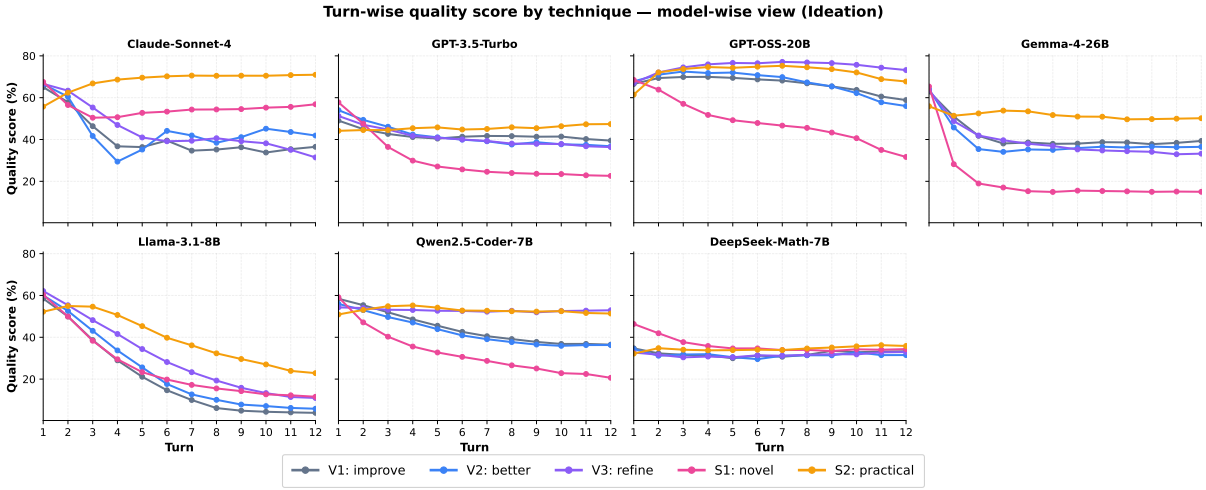

Figure 19: Model-wise view for **ideation**: each subplot shows one model's trajectory under all five feedback techniques. Frontier models (Claude, Gemma, GPT-OSS-20B) show clear technique separation; smaller models show less differentiation. Quality is measured by Gemini-judge composite score and should be interpreted as a relative trajectory signal.

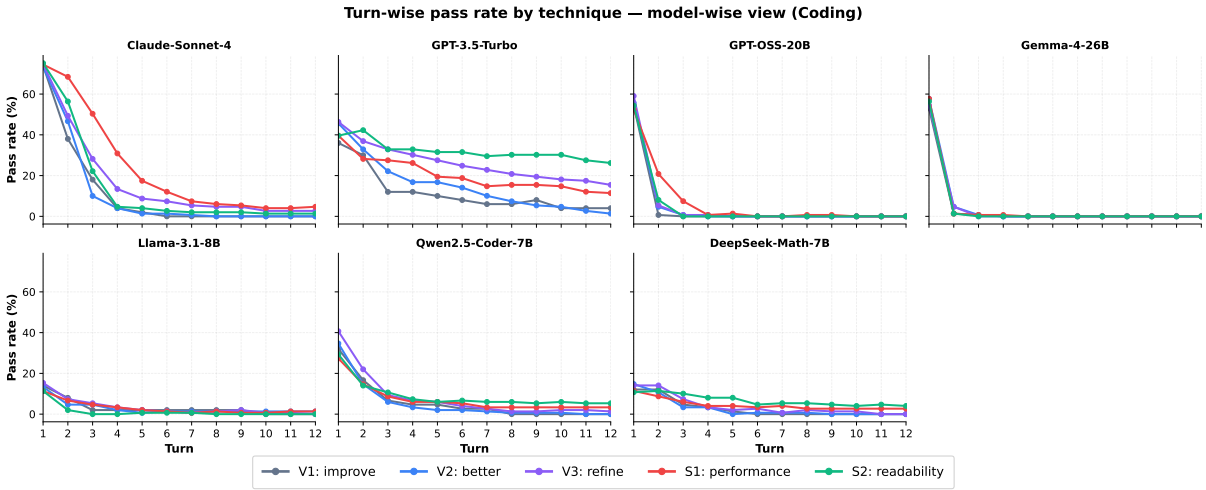

Figure 20: Model-wise view for **coding**: each subplot shows one model's pass-rate trajectory under all five feedback techniques. The collapse pattern is visible across models, though the rate and severity of decline vary.

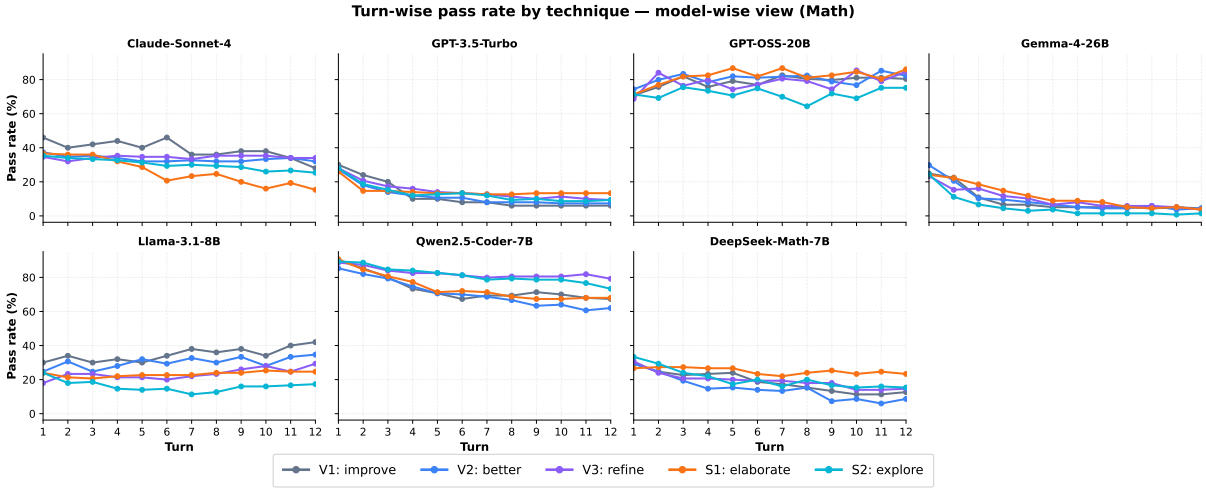

Figure 21: Model-wise view for **math**: each subplot shows one model's pass-rate trajectory under all five feedback techniques. GPT-OSS-20B shows upward trajectories across most techniques; other models show technique-dependent stability, decline, or limited recovery.

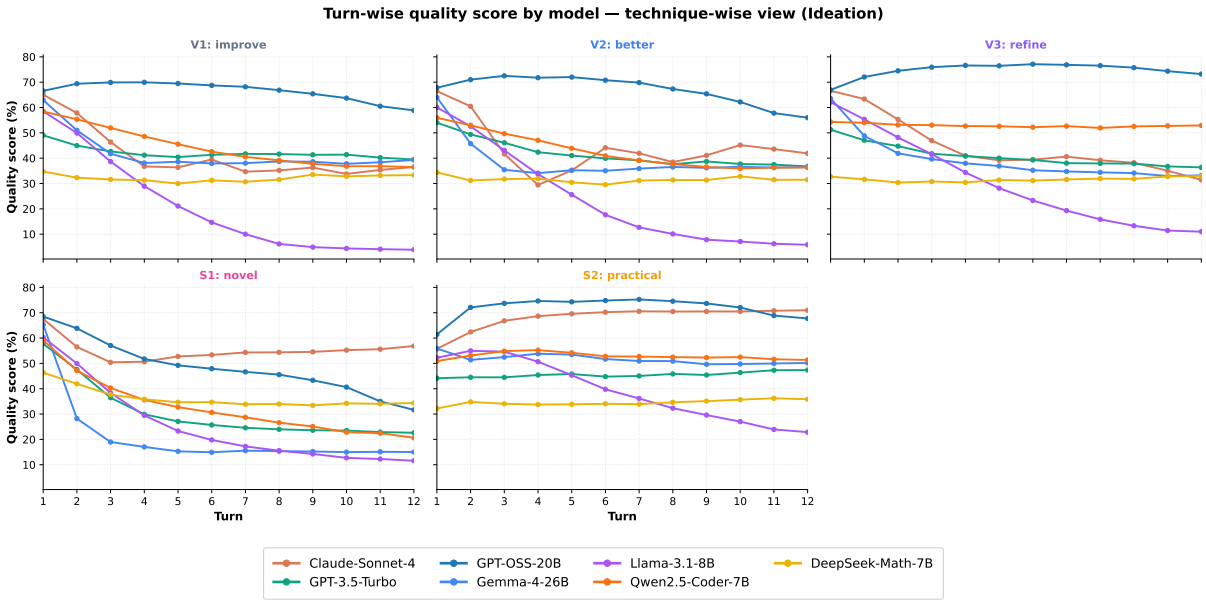

Figure 22: Technique-wise view for **ideation**: each subplot shows one technique with lines for all models. `s1_novel` and `s2_practical` reveal the widest inter-model differences, while vague prompts produce more uniform and flatter trajectories.

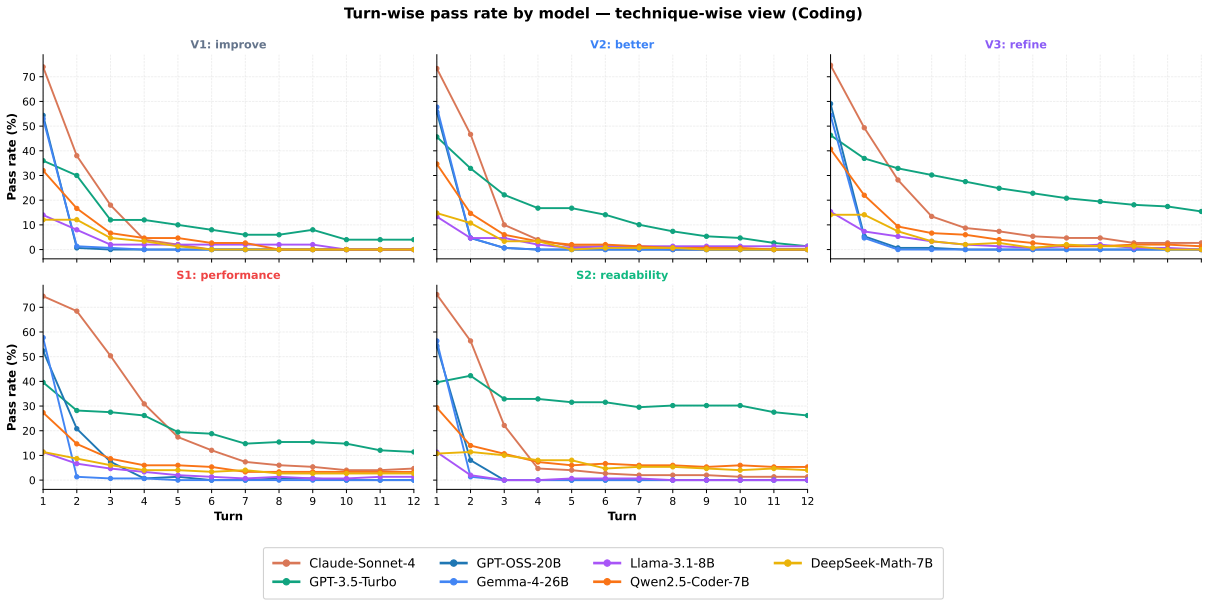

Figure 23: Technique-wise view for **coding**: each subplot shows one technique with lines for all models. All techniques show substantial degradation over turns; `s2_maintainability` on GPT-3.5-Turbo is the main partial exception.

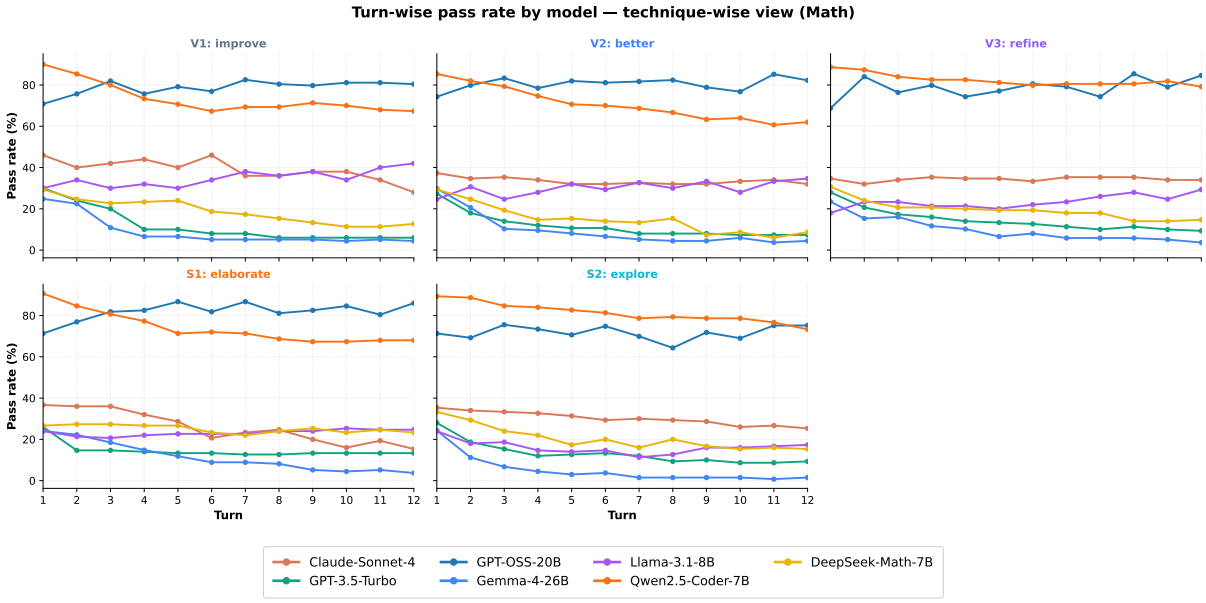

Figure 24: Technique-wise view for **math**: each subplot shows one technique with lines for all models. GPT-OSS-20B is consistently strongest; other models show technique-dependent variation in stability or decline.

Per-domain summary statistics. Ideation. Cohort-mean buzzword count under `s1_novel` increases from 1.42 at T1 to 35.98 at T12 (Claude individually: 1.57→25.62). The sharpest judge-rated clarity decline occurs for Llama-3.1-8B under `v2_better` (7.18→1.29). The clearest targeted-steering pattern appears for Claude under `s2_practical`, where judge-rated feasibility rises from 3.54 to 9.96 and clarity from 7.62 to 9.93, while originality falls from 6.89 to 2.27 as the expected trade-off. These results support the main finding that targeted prompts can control ideation trajectories, whereas vague improvement language often produces rhetorical inflation.

Coding. Every completed open-weight (model, technique) coding cell has best@12 $\geq$ 13% but final@12 $\leq$ 6%; Claude is also $\leq$ 5% final@12 on every technique. The main exception is GPT-3.5-Turbo under `s2_maintainability`, which retains $>$ 20% correctness at T12 (final@12 = 26.2%). Drift_rate is above 70% in nearly all completed coding cells, and many cells reach 100%, meaning every task that was correct at least once is wrong by T12. The lowest drift_rate cell is GPT-3.5-Turbo under `s2_maintainability` at 49.2%.

Math. GPT-OSS-20B is the clearest case of sustained late-turn improvement in math: best@12 is at least 98.6% on every reported technique, and final@12 is at least 74.8%. Because GPT-OSS-20B also starts from a high T1 baseline, its recovery-rate values should be interpreted alongside T1 accuracy rather than as standalone evidence of self-correction from failure. Claude under `s1_elaboration` illustrates a large discovery–retention gap (best@12=60.0%, final@12=15.3%, drift_rate=63.6%). Llama-3.1-8B shows a large wrong-at-T1 to correct-at-T12 effect under `v2_better` (recovery_rate=63.7%).

Best-of-N vs. iterative refinement (Table 3). A natural alternative to iterative refinement is best-of-N sampling: draw N independent responses and return the best according to the same task-level scorer. We run this baseline for GPT-3.5-Turbo ($N = 12$, temperature 0.7, same tasks as the main experiment) on coding and math, using the same unit-test and answer-equivalence scorers. Parallel sampling outperforms the final turn of the memoryless refinement loop in both domains. For coding, pass@12 reaches 28.9% compared with iterative@12 at 13.0% (+15.9 pp). For math, pass@1 alone (14.7%) exceeds iterative@12 (9.4%), while pass@12 reaches 26.0%. These results suggest that, at least for GPT-3.5-Turbo, repeated self-editing is less effective than independent resampling under the same sampling budget.

Domain	pass@1	pass@12	iter@12
Coding	13.4%	28.9%	13.0%
Math	14.7%	26.0%	9.4%

Table 3: Best-of-N baseline for GPT-3.5-Turbo (N=12, temp. 0.7). **pass@1**: single independent sample. **pass@12**: best result over 12 independent samples under the task-level scorer. **iter@12**: turn-12 output of the memoryless refinement loop.

Gemini judge reliability (Table 4). We assess judge sensitivity by re-scoring a stratified 50% sample (2,949 trajectories, 35,108 turn-level observations) with Claude Sonnet 4 using the same rubric as Gemini-2.5-Pro. At the turn level, agreement is moderate: Spearman $\rho = 0.583$ and linear-weighted $\kappa = 0.392$, which is expected for subjective creative-evaluation dimensions. At the trajectory level, the judges agree on the direction of change in 66% of trajectories, and agreement is strongest for the paper's most important ideation trend: feasibility decline under `s1_novel`, where directional agreement reaches 92% ($n = 609$, $p \approx 0$; $\Delta\rho = 0.667$). Gemini also shows a small systematic offset, scoring originality and clarity +1.07/+1.42 points higher on average. We therefore interpret ideation scores as relative trajectory signals rather than absolute quality labels.

	Turn-level			Δ(T12−T1)	
Dim.	r	ρ	κ	$\Delta\rho$	Dir
Originality	0.541	0.550	0.386	0.493	69%
Feasibility	0.649	0.663	0.503	0.667	69%
Clarity	0.481	0.435	0.292	0.478	57%
Buzzwords	0.423	0.683	0.387	0.642	69%
Mean	**0.524**	**0.583**	**0.392**	**0.570**	**66%**

Table 4: Inter-annotator agreement (Gemini-2.5-Pro vs. Claude Sonnet 4; 2,949 trajectories, 35,108 observations). **Turn-level**: per-turn score correlations (r Pearson, ρ Spearman, κ linear-weighted). **Δ(T12−T1)**: $\Delta\rho$ is Spearman on change scores; **Dir** = directional agreement (% where both judges agree on sign of change; all $p \approx 0$ vs. 50% chance). Under `s1_novel`, feasibility Dir reaches **92%** (n=609), supporting the paper's main ideation steering claim.