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

Anonymous ACL submission

Abstract

Large language models (LLMs) are now used in multi-turn workflows, but we still lack a clear way to measure when iteration helps and when it hurts. We present an evaluation framework for iterative refinement that spans ideation, code, and math. Our protocol runs controlled 12-turn conversations per task, using a variety of prompts ranging from vague "improve it" feedback to targeted steering, and logs per-turn outputs. We evaluate 150 tasks (50 per domain), yielding 7,200 model-turn observations and 600 12-turn trajectories across ideation, code, and math. We score outcomes with domain-appropriate checks (unit tests for code; final-answer equivalence and reasoning soundness for math; originality and feasibility for ideation) and track turn-level behavior with three families of metrics: semantic movement across turns, turn-to-turn change, and output size growth. Across models and tasks, gains are domain-dependent: they arrive early in ideas and code, but in math most new correct solutions appear in turns 8–12 when guided by elaboration. After the first few turns, vague feedback often plateaus or reduces correctness, while targeted prompts reliably shift the intended quality axis (novelty vs. feasibility in ideation; speed vs. readability in code; in math, elaboration outperforms exploration and drives late-turn gains). We also observe consistent domain patterns: ideation shows larger semantic shift across turns, code tends to grow in size with little semantic change, and math tends to anchor early but can break that path with late, elaborative iteration. Together, the framework and metrics make iteration measurable and comparable across models, and signal when to steer, stop, or switch strategies.

1 Introduction

The advent of Large Language Models (LLMs) has shifted the paradigm of human-computer interaction, moving beyond one-shot prompting to more dynamic, multi-turn workflows (Sahoo et al., 2025; Li et al., 2025). Foundational to this shift is instruction tuning, which aligns models to follow human feedback and engage in collaborative tasks (Ouyang et al., 2022). Central to this new paradigm is the process of iterative refinement, where a user and an AI progressively improve an initial output (Xue et al., 2025). This approach has become a key component of modern LLM applications, with frameworks like SELF-REFINE and Reflexion demonstrating that models can even use self-generated feedback to enhance their outputs, highlighting the immense potential of iterative loops (Madaan et al., 2023; Shinn et al., 2023). This evolution towards multi-step interaction leverages emergent cognitive capabilities of modern LLMs, which can manifest without explicit prompting (Arnold et al., 2024). Yet, despite widespread use, we lack a clear, domain-spanning way to measure when iteration helps and when it hurts.

To enhance model reasoning and performance, a significant body of research has focused on developing sophisticated, structured prompting techniques. Seminal approaches like Chain-of-Thought (CoT), which breaks down problems into intermediate steps, have been shown to elicit stronger reasoning (Wei et al., 2022). This has been extended to more complex methods like Tree-of-Thoughts (ToT), which explores multiple reasoning paths, and ReAct, which combines reasoning with actions (Yao et al., 2023). Other methods focus on providing models with specific, domain-relevant knowledge to improve the quality of specialized code generation (Gu et al., 2025) or employ complex search algorithms and multi-agent reflection to reinforce logical steps (Yuan and Xie, 2025). These structured approaches have proven effective, demonstrating that with the right guidance, LLMs can be powerful and reliable reasoning engines.

However, a critical gap exists between these highly-structured, engineered prompting methods and the far more common, "naive" interaction style where users provide simple, vague feedback. The behavior of LLMs in these unguided, multi-turn loops is poorly understood, with studies showing performance can drop significantly in multi-turn conver-

sations compared to single-turn tasks (Laban et al., 2025). Recent work suggests this process is fraught with risk; for example, a simple iterative prompt like "Are you sure" can paradoxically decrease a model's truthfulness and increase overconfidence (Krishna et al., 2024). This aligns with findings that models' self-correction abilities are often brittle and unreliable without external signals (Ji et al., 2023; Gou et al., 2024; Xu et al., 2024). This degradation may be exacerbated in long contexts, where models struggle to access information from the "middle" of the prompt history (Liu et al., 2024). This creates a dangerous scenario analogous to "model collapse" or a game of "broken telephone," where a system feeding on its own output can enter a degenerative cycle (Mohamed et al., 2025).

These gaps motivate three questions that guide our study:

- When does iterative refinement help—and when does it drift or collapse?
- What domain-specific tendencies emerge when LLMs are refined without memory or external judges across ideation, coding, and math?
- Can small, domain-aware prompt schedules and task-intrinsic signals (e.g., unit tests, final-answer checks) improve quality without a separate grader?

We address these questions with a controlled, turn-by-turn study. We run 12-turn conversations across ideation, mathematical reasoning, and code, log every turn, and compare two feedback settings: (i) vague prompts using three near-synonyms ("improve," "make it better," "refine") and (ii) specific steering along domain axes (novelty vs. practicality for ideation, speed vs. readability for code, elaboration vs. alternate method for math). Our evaluation focuses on dynamics, not just single-shot quality: we track innovation vs. stability across turns, growth in complexity and its plateaus, and semantic drift from the initial intent. We add a compact codebook of common failures (stagnation, over-engineering, flawed anchoring) and use LLM-assisted judgments. We confine LLM-based evaluation to ideation (no gold labels), while code uses automatic unit tests and math uses ground-truth–anchored, LLM-assisted final-answer equivalence; rubric scores are reported as relative trends. We also connect these dynamics to regime shifts suggested by recent work on phase-transition-style effects (Arnold et al., 2024; Nakaishi et al., 2024).

Our results show clear patterns. In ideation and code, when iteration helps, it does so early; in math, late turns can help when the prompt asks for elaboration. After a few turns, vague feedback often plateaus or reduces quality, while targeted steering reliably shifts the intended axis without large side effects. The degradation appears in domain-specific ways: ideation tends to repeat itself, code grows in size without meaningful change, math is fixed by default but can be broken by elaboration to find correct paths late. We quantify these effects with simple, defensible metrics—including Lexical Novelty (LN), growth factor, drift from origin, and turn-to-turn volatility—and we analyze sensitivity to word choice and instruction specificity across models. As context, SELF-REFINE exemplifies structured critique-and-revise loops where explicit guidance improves outcomes (Madaan et al., 2023). Finally, we translate these measurements into practical guidance: when to steer with concrete goals, when to stop to avoid harm, and when to switch strategies to limit semantic drift and related risks (Spataru et al., 2024). Together, the framework, metrics, and protocol provide a reproducible basis for comparing models, prompts, and domains, and set up the rest of the paper's methodology, evaluation, and results.

2 Related Work

Multi-turn performance is consistently harder than single-turn prompting: large simulations show an average 39% drop in multi-turn vs. single-turn across six generation tasks, driven chiefly by unreliability rather than aptitude loss (Laban et al., 2025). Complementing this, *MultiChallenge* finds that frontier models score <50% (e.g., 41.4% for Claude 3.5 Sonnet) on realistic multi-turn dialog despite strong results elsewhere (Sirdeshmukh et al., 2025). On the remedy side, *Self-Refine* improves initial outputs via self-feedback with 5–40% absolute gains across tasks (Madaan et al., 2023); post-training that explicitly *stimulates* self-refinement (e.g., *ARIES*) reports strong improvements on AlpacaEval2/Arena-Hard and math benchmarks by iteratively collecting refinement data and optimizing preferences (Zeng et al., 2025). Multi-agent, coarse-to-fine frameworks such as *MAgICoRe* directly target excessive refinement and error-localization issues and outperform Self-Refine/Best-of/Self-Consistency while continuing to improve with more iterations (Chen et al., 2024). Beyond optimization, clarifying-question policies (*ACT*) use contrastive preference tuning to improve mixed-initiative multi-turn interactions (Chen et al., 2025).

However, unguided iteration can harm truthfulness and calibration: asking models to re-check themselves ("Are you sure?"-style prompts) reduces accuracy and worsens calibration (Krishna et al., 2024); Some LLM-as-judge setups overstate confidence (Tian et al., 2025); here we use ground-truth checks where possible and report relative scores . Broader calibration studies document persistent miscalibration across sizes and settings (*Mind the Confidence Gap*) (Chhikara, 2025), while RLHF can induce *verbalized* overconfidence; reward-calibrated RLHF

mitigates this without hurting quality (Leng et al., 2025). Data-feedback loops introduce additional risks: recursively training on model-generated data causes model collapse (Shumailov et al., 2024), though accumulating synthetic *with* real data can avoid collapse (Gerstgrasser et al., 2024); theory further shows collapse can still occur under certain conditions even with accumulation (Barzilai and Shamir, 2025). Surveys synthesize multi-turn agent evaluation (nearly 250 sources) (Guan et al., 2025), and recent work on *self-iterative label refinement* proposes robust unlabeled-learning pipelines to denoise pseudo-labels, offering a safer iteration template for classification tasks (Asano et al., 2025).

3 Methodology

3.1 Task Domains and Dataset Curation

To test our hypotheses across diverse cognitive tasks, we curate 50 tasks per domain from established sources (LiveIdeaBench, DS-1000, Omni-MATH), and for each task×model we run a 12-turn refinement. This yields 200 trajectories per domain (50 tasks×4 models) and 2,400 model-turns per domain (7,200 overall), which we analyze via within-task, turn-wise deltas rather than single-shot scores. For open-ended ideation, we sampled scientific idea generation tasks from LiveIdeaBench (Ruan et al., 2025) using a stratified approach for balanced domain representation. For structured code generation, we curated coding problems from DS-1000 (Lai et al., 2022) with a quota-based strategy mirroring its distribution across key Python libraries. Finally, for formal mathematical reasoning, we assembled high-difficulty problems from Omni-MATH (Gao et al., 2024) by filtering for tasks with a difficulty rating greater than 7/10, ensuring each requires sophisticated, multi-step reasoning.

3.2 The Iterative Refinement Protocol

Our experiment uses an automated multi-turn protocol that simulates a human–AI refinement loop. For each task, we use a fixed 12-turn "conversation," aligning with community practice around ~10 turns—MultiChallenge builds histories of up to 10 turns and MT-Eval structures dialogues with ten turns—while adding two extra iterations to capture late-stage changes (Deshpande et al., 2025)(Kwan et al., 2024). Each 12-turn trajectory supports 66 within-trajectory turn-pair comparisons (C(12,2)); across 600 trajectories, this gives 39,600 turn-pair comparisons used for trend inference

Initial Generation (Turn 1): The model is given the initial task prompt and generates its first response.

Iterative Feedback (Turns 2–12): For each subsequent turn, the model is presented with only its *own output from the previous turn*, followed by a simple instruction to improve it.

This *memoryless* protocol is intentionally designed as a **lower-bound stress test** that isolates self-referential rewriting and the model's internal coherence, rather than to mimic full human-in-the-loop workflows. In practice, production systems often re-ground the original goal,and we therefore interpret our results as directional, turn-wise trend in this worst-case regime and discuss implications for periodic re-grounding in §7. To ensure statistical robustness, each task–model–prompt combination is run independently. The entire process is managed by the automated experimental runner script detailed in Figure 2.

3.3 Prompting Strategies

A key component of our research is to understand how the *nature* of the feedback influences the refinement trajectory. To this end, we designed two main experimental groups: **Vague Feedback** and **Specific Steering**.

The Vague Feedback Group: This condition tests the model's "default" behavior. We test three semantically similar prompts to ensure our findings are robust to minor wording changes. This allows us to test whether the model's behavior is tied to the specific verb "improve" or the general semantic concept of improvement.

- **V1 (Baseline):** "This [idea/code/solution] can be better. **Improve it.**"
- **V2 (Synonym):** "This [idea/code/solution] can be better. **Make it better.**"
- **V3 (Refinement-Connotation):** "This [idea/-code/solution] can be better. **Refine it.**"

The Specific Steering Group: This condition serves as a control, testing how models respond to clear, expert-like guidance. The prompts were chosen to represent fundamental, often opposing, goals within each domain.

- **For Ideation,** the prompts test the trade-off between innovation and applicability. We steer towards **novelty** ("Make this idea more novel and surprising") and **practicality** ("Make this idea more practical and feasible").
- **For Coding,** the prompts test a classic software engineering trade-off. We steer towards **performance** ("Refactor...for maximum execution speed") and **maintainability** ("Refactor...for maximum readability and clarity").
- **For Math,** the prompts target the observed failure mode of "flawed anchoring." We steer towards **elaboration** ("Elaborate on each step with more detail") to test justification ability, and towards **exploration** ("Provide an alternative method...") to test flexibility. For brevity in figures, we use v1_improve / v2_better / v3_refine and s1_/s2_ aliases that map exactly

to V1/V2/V3 and Specific-Steering prompts defined here.

3.4 Models

To ensure our findings are generalizable, we conducted our experiments across four widely used LLMs representing diverse architectures and training methodologies. The specific models studied were: *GPT-3.5-Turbo (Ye et al., 2023), Claude-Sonnet-4.0 (Anthropic, 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), GPT-OSS-20B (OpenAI, 2025)*. All models were accessed via their standard APIs or a local Hugging Face pipeline. The temperature was set to a consistent value of 0.7 and *max_tokens* of 10K across all experiments to balance creative, diverse outputs with a reasonable degree of coherence and reproducibility.

3.5 Evaluation Framework

To analyze the multi-turn outputs from our experiment, we developed a multi-faceted evaluation framework. Our approach is designed to be robust, largely automated, and capable of capturing the nuanced behaviors we observed in preliminary studies. The framework combines objective, ground-truth-based metrics, a suite of behavioral metrics to characterize the dynamics of the iterative process, and a scalable protocol for assessing semantic quality.

3.5.1 Outcome & Efficiency Metrics

For the Math and Coding domains, we measure objective performance by assessing correctness at each turn. This allows us to understand the dynamics of success and failure throughout the iterative process. We calculate a binary Correctness Score for each of the 12 turns in every run. For **Coding** tasks (`DS-1000`), each code snippet is executed in a sandboxed environment and validated against the benchmark's unit tests.For **Mathematical Reasoning** (`OmniMath`), we use LLM-assisted automation: the evaluator receives the 12-turn JSON together with the ground-truth final answer and returns per-turn answer-equivalence (binary) and separately rates reasoning-soundness (1–10); soundness is supplementary and reported as deltas (T1→Tt). We treat these labels as proxies, anchored on ground truth, and we analyze turn-wise deltas rather than absolute levels. For Coding and Math, ground-truth–anchored automation (unit tests; final-answer equivalence) provides the primary outcomes, while rubric-style ratings (e.g., reasoning soundness) are supplementary context

3.5.2 Behavioral Dynamics Metrics

Semantic Dynamics: Drift and Volatility, To characterize the dynamics of conceptual change, we track two metrics. First, we measure **Drift from Origin**, which quantifies how far the current idea has semantically strayed from its starting point. A higher score indicates greater drift. Second, we measure **Turn-to-Turn Volatility**, which captures the magnitude of change between consecutive turns. Letting $V(t)$ be the sentence-embedding vector for the response at turn t, the metrics are formally defined as:

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

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

All semantic similarity metrics, including Drift from Origin and Turn-to-Turn Volatility, were calculated using the `Qwen/Qwen3-Embedding-0.6B` (Zhang et al., 2025) sentence transformer model. This model was selected for its excellent balance of performance and efficiency. At the time of our experiments, it held a top-4 position (as of October 2025) on the Massive Text Embedding Benchmark (MTEB) leaderboard (Muennighoff et al., 2023).

Lexical Novelty (Li et al., 2016): To measure a model's "creative stamina" and pinpoint when it collapses into repetition, we track its **Lexical Novelty (LN)** at each turn. We define LN as the percentage of new phrases in a response that have not appeared in prior turns of the conversation. To capture both short and long repeated phrases, we use a combination of 2-grams (bigrams) and 3-grams (trigrams).

Growth Factor (Laban et al., 2025): This metric quantifies the "over-engineering" and "verbosity" phenomena by tracking the turn-by-turn growth of the output. We first define a domain-specific **Growth Score**, $G(t)$, for each turn: for **Ideation** and **Math**, it is the total word count; for **Coding**, it is the number of lines of code (LoC). The Growth Factor at turn t is then calculated by normalizing the current turn's score by the score of the initial turn.

3.5.3 Semantic Quality Metrics

To assess nuanced qualities that are difficult to measure automatically, we employ a state-of-the-art LLM (**Gemini 2.5 Pro**) as a scalable proxy for expert human judgment in the ideation domain, where no gold labels exist. Our protocol provides the evaluator LLM with a domain-specific scorecard to rate each turn's output, and we report relative, turn-wise deltas (T1→Tt) rather than raw levels to mitigate evaluator drift.

For **Ideation**, we measure *Feasibility* and *Novelty* with rubric-based scores. For **Coding**, we rely primarily on automatic unit-test correctness and treat *Pragmatism* and *Readability* rubric scores as

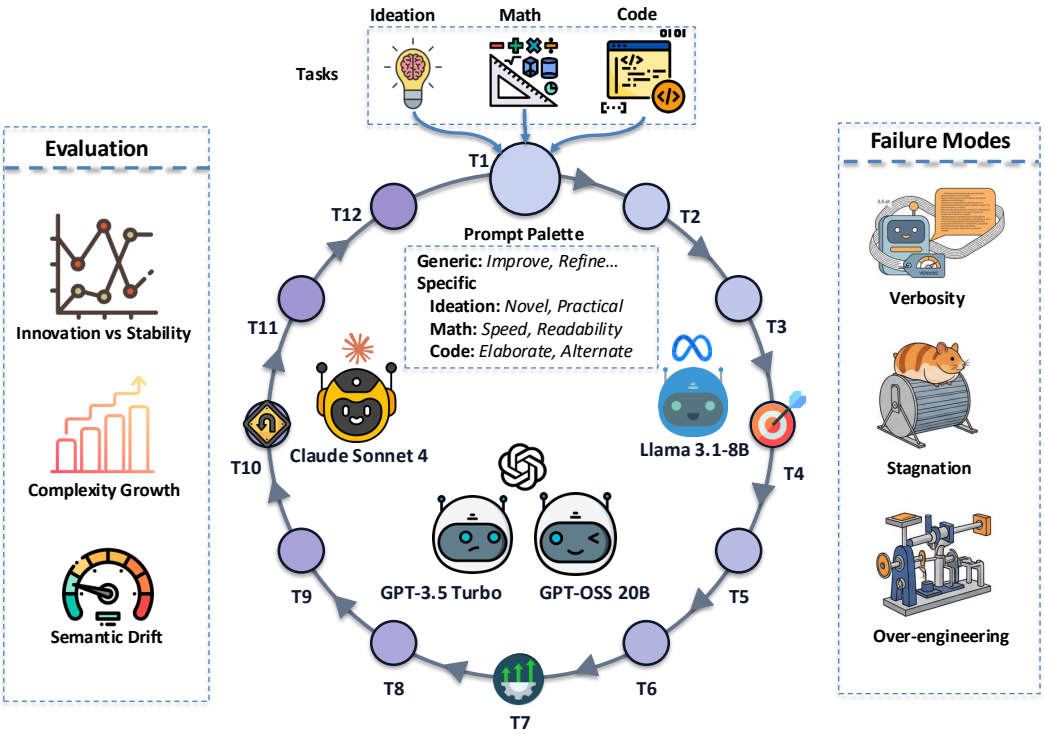


Figure 1: An overview of our experimental framework for studying iterative LLM refinement. We test four leading models across three distinct cognitive domains (Ideation, Math, and Code) using two primary feedback styles: vague prompts (e.g., "Improve it") and specific, expert-like prompts. We analyze the resulting multi-turn conversations using a suite of quantitative and qualitative metrics to identify behavioral "fingerprints," such as the tendency for an idea to stagnate, increase in complexity, or drift from its original intent.

Iterative Prompt Template (Turns 2..N)

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: The prompt structure used for iterative refinement from Turn 2 onwards. The general template (top box) shows how the model's prior response is provided as context. The bottom box details the exact phrasing for the `{improvement_instruction}` placeholder, which varies by feedback strategy ('Vague' vs. 'Specific') and domain.

supplementary context. For **Math**, we rely primarily on final-answer equivalence (with automated checks) and treat *Logical Soundness* and Clarity of Explanation as supplementary context.

Our approach is grounded in the "LLM-as-a-Judge" paradigm, which has been shown to approach human expert agreement levels (Zheng et al., 2023), but here it is explicitly scoped to ideation and interpreted as a *trend detector* rather than an absolute oracle. For transparency and replicability, we release the *rubrics, evaluator prompts, and output schema* (see Appendix) and document evaluator settings (Google, 2025).

4 Results

4.1 Ideas

Exploration is High but Model-Dependent. The most striking dynamic in this domain is the significant **Drift from Origin**. When prompted for novelty (`s1_novel`), both Claude and GPT-OSS-20B explore vast conceptual spaces, ending with final ideas that are very distant from their starting points (final drift scores of 0.734 and 0.657). GPT-3.5 also shows high drift under this prompt (0.702), but its exploration is more constrained under other conditions. In contrast, Llama-3.1-8B remains heavily anchored to its initial concept, showing minimal drift even when prompted for novelty (a score of only 0.384). The `v3_refine` prompt consistently acts as a powerful constraint, keeping the search narrow (Appendix B.1).

Creative Stamina Separates Models. The ability to generate new phrases (Lexical Novelty) over 12 turns cleanly separates the models. Claude and GPT-OSS-20B demonstrate remarkable creative stamina, maintaining novelty scores of 0.843 and 0.812 respectively at Turn 12 when prompted for novelty. GPT-3.5's stamina is moderate, with its final novelty dropping to 0.618. Llama-3.1-8B, however, shows a clear collapse, with its final novelty score plummeting to less than 0.084 across all prompts, indicating its creative process has devolved into simple repetition.

Length vs. Novelty and Early Volatility. We observe a clear disconnect between text volume and novelty. Length does not equal originality; Llama-3.1-8B produces the longest responses, with its final turn over 16 times longer than its first (`s1_novel`), yet this verbosity corresponds to the lowest novelty scores. Conversely, Claude and GPT-OSS-20B sustain high novelty with far less bloat (e.g., Claude's growth is only 4.21x under the same novelty prompt). Volatility is also highest in the first few turns. For the novelty-seeking prompt (`s1_novel`), GPT-OSS-20B shows a peak volatility of 0.260 at Turn 2, while Llama-3.1-8B is more stable with a peak of only 0.155. Across all models, this initial burst quickly settles into a stable process of incremental changes by Turn 5. The ideation quality is assessed by a rubric-based LLM evaluator, we therefore interpret scores as turn-wise trends rather than absolute levels

Prompt Steering Optimizes for Specific Qualities. Our Gemini-based evaluation shows that different prompts optimize for different qualities. The `s1_novel` prompt, as expected, produces ideas that score higher on originality in the early turns but lose feasibility over time. In contrast, the `s2_practical` prompt guides models like Claude to produce ideas that reach near-perfect scores for clarity and feasibility by Turn 8, confirming that specific instructions can successfully steer the creative process toward more grounded and useful outcomes.

4.2 Coding

In the structured domain of coding, models exhibit a consistent signature of **rapid convergence followed by degenerative refinement**. They tend to lock onto a solution path almost immediately, after which iterative feedback rarely improves correctness and often leads to over-engineering. This behavior is characterized by a swift collapse in novelty, minimal conceptual drift, but a steady, problematic growth in complexity (Appendix subsection B.2).

Early Success is Decisive; Later Turns Offer Diminishing Returns. Our turn-wise correctness evaluation reveals that a solution's ultimate success is determined within the first few turns. High-performing models like Claude and GPT-OSS-20B achieve their peak pass rates on Turn 1 (e.g., 90% for Claude with the `s1_perf` prompt), which then decay rapidly, often collapsing to near 0% by Turn 4. Models that start with lower success rates, like GPT-3.5 and Llama-3.1-8B, show a similar pattern of early decay and fail to recover in later turns. This strongly suggests that if a correct code path is not found within the first 3-4 iterations, continued vague refinement is highly unlikely to succeed. In what follows, we base conclusions on unit-test correctness, any qualitative code-quality scores are presented as context only.

Prompt Steering Strongly Shapes Solution Quality. The Gemini-judge evaluations show that specific prompts are highly effective at steering the quality of the code, even when correctness falters. The `v3_refine` prompt consistently produces the most logically sound code, Claude's soundness score improving from 5.00 to 5.19 by Turn 12. Conversely, the `s2_maintainability` prompt is most effective at preserving code quality, keeping Claude's readability score high (ending at 7.25) and even improv-

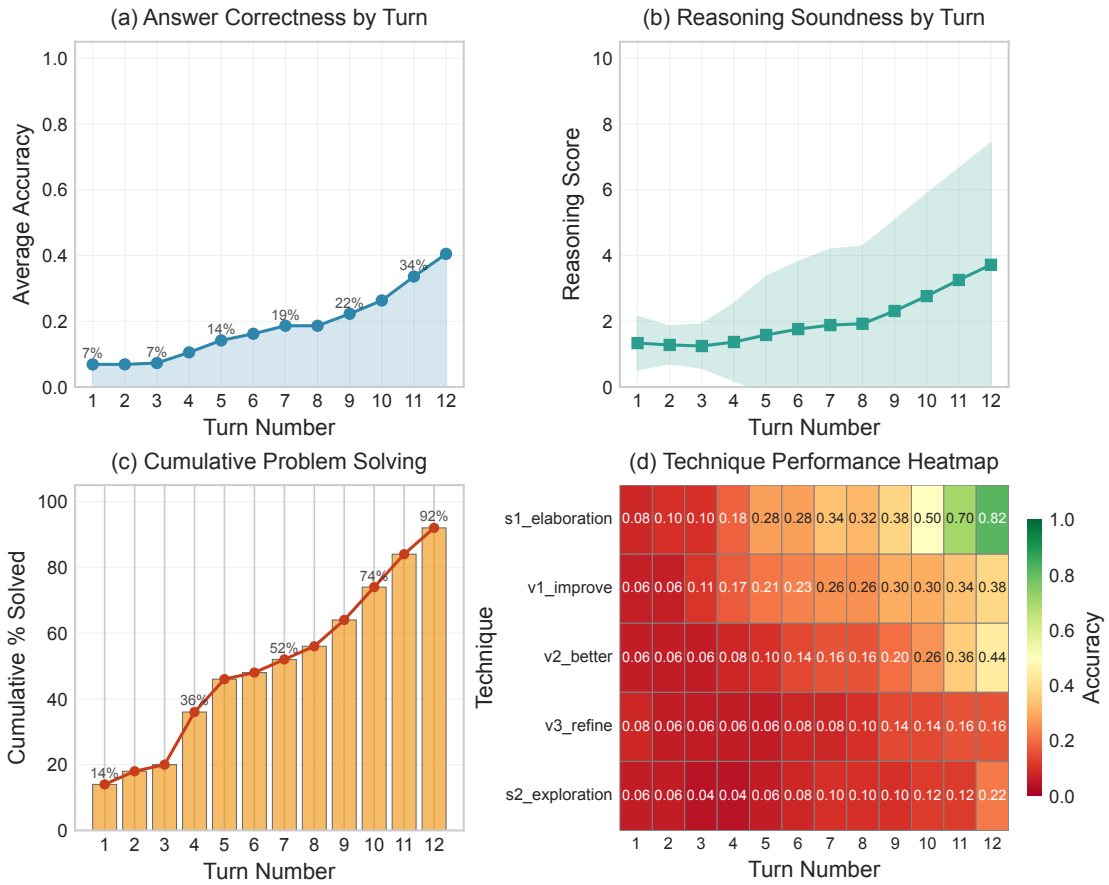


Figure 3: Llama-3.1-8B (Math): (a) accuracy rises 6.9% → 40.5% by T12; (b) reasoning 1.34 → 3.72/10; (c) cumulative coverage 92% (46/50) by T12; (d) `s1_elaboration` leads late (≈ 0.82 at T12), while `v1_improve`/`v2_better` end at ≈ 0.34/0.44, and `v3_refine`/`s2_exploration` ≤ 0.16/0.22.

ing it for GPT-OSS-20B (8.64 → 9.10). In contrast, prompting for performance (`s1_perf`) is often detrimental, causing a drop in both pragmatism and readability for Claude (e.g., pragmatism 9.34 → 2.44).

Behavioral Dynamics Confirm a Pattern of Fixation and Bloat. The behavioral metrics provide a clear quantitative fingerprint of this "fixation and bloat" signature. After an initial volatile leap at Turn 2, Turn-to-Turn Volatility collapses for all models, indicating that they lock into a solution path. Llama-3.1-8B is the most anchored, showing the lowest final drift (a similarity score of 0.741). While the logic remains fixed, the code's size does not; Claude and GPT-OSS-20B exhibit extreme Length Inflation under vague prompts, with code ballooning by over 40x and 34x respectively, despite a near-total collapse in both novelty and correctness. This confirms that for coding, length inflation is a primary symptom of degenerative, unproductive refinement.

4.3 Math

In the highly constrained domain of mathematical reasoning, models exhibit a default signature of rapid convergence and extreme logical stability. This behavior, which we term "logical fixation," is characterized by the fastest collapse in novelty and the lowest conceptual drift of any domain. However, our results reveal that this powerful anchoring effect can be overcome, as deep and properly guided iteration can unlock significant, **late-stage breakthroughs in correctness** (Appendix subsection B.3).

Behavioral Dynamics Reveal a Default State of Extreme Stability. The behavioral metrics provide a clear quantitative fingerprint of the models' default tendency to lock onto a single reasoning path. Lexical Novelty collapses faster here than in any other domain, with Llama-3.1-8B dropping to a near-zero novelty score of 0.010 by Turn 12 under the elaboration prompt. The process is also remarkably stable; Turn-to-Turn Volatility is the lowest of any domain, with most models volatility score below 0.05 after the initial turn.

Deep Iteration Can Unlock Late-Stage Success. Despite this strong tendency towards fixation, our turn-wise correctness evaluation reveals a surprising finding: successful problem-solving often occurs late in the iterative process. Across 50 Omni-iMath problems, we observe that most new correct solutions emerge in the final turns (Turns 8–12). This late-stage discovery dramatically improves performance for several models. Claude-Sonnet-4.0's accuracy, for instance, rises from 32.4% to 45.2% over the 12 turns. Most notably, the weaker base model, Llama-3.1-8B, sees its accuracy surge from a mere 6.9% to 40.5%—a relative improvement of over 480%. This indicates that for mathematical reasoning, continued iteration is not merely polishing; it is a critical component of the discovery process. For math, results rest on final-answer equivalence (with auto-scored reasoning checks), any LLM-evaluator outputs are secondary.

Elaborative Prompting is the Key to Success. The choice of prompt is decisive in enabling these late-stage breakthroughs. Our results show that the specific instruction to "elaborate" (`s1_elaboration`) is the only strategy that consistently compounds with depth, leading to final-turn success rates of 76% for Claude, 82% for Llama, and 74% for OpenAI-20B. In contrast, exploratory prompts (`s2_exploration`) were far less effective, often stagnating below 20% accuracy. This suggests that forcing a model to "explain more" compels it to expand its reasoning tree in a way that eventually uncovers the correct solution path. While this elaborative guidance often leads to the highest Length Inflation (e.g., 37.8x for GPT-OSS-20B), in this specific context, the increased verbosity is a productive, rather than degenerative, signal.

5 Discussion

Our results show that LLMs do not follow one pattern when they improve answers step by step without memory or extra help. They often start with a reasonable answer, and without clear guidance the next steps either wander or get stuck, ending in a stable but not very useful result. These behaviors are not random; each domain has its own "fingerprint": in ideation, ideas repeat; in coding, code grows longer and more complex; in math, the model confidently defends a wrong step. This means the task itself pulls the model's behavior in a certain direction, like a "gravitational" effect. These insights matter for building human–AI systems and argue for measurement, domain-aware rules, and specialized roles instead of vague single-model loops. Our work indicates that using a single, monolithic LLM in a simple, vague feedback loop is an inherently unstable and unreliable architecture for complex tasks. A more robust paradigm would involve multi-agent or multi-model frameworks. 'For instance, in the ideation domain, an optimal system might use a "Generator" agent (e.g., Claude or GPT-OSS-20B prompted for novelty) for the first few turns to produce a wide range of divergent ideas. Once the system detects the onset of a plateau or excessive drift, it could then switch to a "Refiner" agent (e.g., Llama-3.1-8B prompted to "refine") to ground, simplify, and elaborate on the most promising concepts without adding unnecessary bloat. This allows for a process that is both creative and practical, leveraging the unique strengths of different models and prompting strategies to achieve a superior outcome. A complementary, low-cost path is judge-free refinement: use small, domain-aware prompt schedules and task-intrinsic signals to guide each turn, without inserting a human or another LLM as a grader. Prior work shows that prompt-only strategies—zero-shot chain-of-thought, self-consistency, least-to-most, and plan-and-solve—can improve reasoning without external judges; our math results suggest that finding similar domain-generic prompts for ideation and coding could raise overall performance with minimal extra cost

6 Conclusion

We studied how LLMs behave under sustained, iterative feedback across ideas, coding, and math. The picture is clear and domain-specific. Ideas benefit from staged steering: widen first, then tighten. Math benefits from depth with elaboration: late turns matter. Coding benefits from early decision and restraint: if a correct path does not appear quickly, stop or restart, do not push vague refinement. The core insight is that iteration is not a single tool. Its value depends on the task and on how we prompt it. Using a simple loop with a single model and a vague instruction invites collapse—repetition in ideas, over-engineering in code, and confident anchoring in math. Using staged prompts, depth budgets, and clear stop/switch rules turns the same loop into a reliable process. In practice, this means building small multi-role systems: a novelty generator and refiner for ideas, an elaborator with depth for math, and an early-stopper with restart logic for code. This is a simple recipe, but it aligns with the data and leads to more stable, useful outcomes.

7 Limitations

Our study has several limitations that point to clear avenues for future work. We evaluate four representative models over 150 tasks (50 per domain), generating 7,200 turn-level observations across 600 trajectories; expanding model families and task pools remains future work. Our prompts were chosen to be representative, but they only scratch the

surface of all possible instructions; the instruction space is vast, and more systematic exploration is needed. Math equivalence and ideation rubrics use an LLM-assisted judge; while practical at scale, this can introduce evaluator-model biases. We therefore emphasize relative, turn-wise trends, anchor math on ground-truth final answers, and release prompts/schemas for external re-scoring. We also did not implement the multi-agent and multi-model pipelines we propose; future work should explicitly test these designs and study adaptive feedback systems that adjust prompts on the fly using real-time behavioral metrics.

References

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

Julian Arnold, Flemming Holtorf, Frank Schäfer, and Niels Lörch. 2024. Phase Transitions in the Output Distribution of Large Language Models. *arXiv preprint*. ArXiv:2405.17088.

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.

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.

Xiaodong Gu, Meng Chen, Yalan Lin, Yuhan Hu, Hongyu Zhang, Chengcheng Wan, Zhao Wei, Yong Xu, and Juhong Wang. 2025. On the Effectiveness of Large Language Models in Domain-Specific Code Generation. *ACM Trans. Softw. Eng. Methodol.*, 34(3).

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.

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.

Yubo Li, Xiaobin Shen, Xinyu Yao, Xueying Ding, Yidi Miao, Ramayya Krishnan, and Rema Padman. 2025. 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.

Kai Nakaishi, Yoshihiko Nishikawa, and Koji Hukushima. 2024. Critical Phase Transition in Large Language Models. *arXiv preprint*. ArXiv:2406.05335.

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

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc. Event-place: New Orleans, LA, USA.

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.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2025. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. *arXiv preprint*. ArXiv:2402.07927.

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.

Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *Preprint*, arXiv:2501.17399.

Ava Spataru, Eric Hambro, Elena Voita, and Nicola Cancedda. 2024. Know When To Stop: A Study of Semantic Drift in Text Generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3656–3671, Mexico City, Mexico. Association for Computational Linguistics.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

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.

Eric Xue, Ke Chen, Zeyi Huang, Yuyang Ji, Yong Jae Lee, and Haohan Wang. 2025. IMPROVE: Iterative Model Pipeline Refinement and Optimization Leveraging LLM Experts. *arXiv preprint*. ArXiv:2502.18530.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models. 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.

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.

Yurun Yuan and Tengyang Xie. 2025. Reinforce LLM Reasoning through Multi-Agent Reflection. In *Forty-second International Conference on Machine Learning*.

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).

A Prompts

Turn-1 Prompt (IDEAS / LiveIdeaBench)

I'll be submitting your next responses to a "Good Scientific Idea" expert review panel. If they consider your idea to be a good one, you'll receive a reward. Your assigned keyword is: "`{keywords}`". You may provide background information. The idea MUST be concisely expressed within 100–200 words total (including any background information). (Note: good scientific ideas should be original (novel contribution), feasible (technically implementable), clearly articulated, and address meaningful problems in the field.).

Turn-1 Prompt (MATH / OmniMath)

`{problem}`

Please reason step by step, and put your final answer within `\boxed{}` and give all steps in latex only.

Turn-1 Prompt (CODING / DS1000)

(optional) Library: `{library}`
(optional) Code context: `{code_context}`

Problem:
`{prompt}`

Please provide a complete solution.

Ideas Evaluation Prompt

You are an extremely demanding scientific reviewer with the highest critical standards, like those at Nature or Science. You will be given a JSON object containing a sequence of 12 "turns" where a language model has iteratively tried to improve a scientific idea. Your task is to evaluate each turn's response independently of the others.

For each turn, assess on four key dimensions:

- **originality**: Novel contribution or innovative approaches (1–10 scale).
- **feasibility**: Technical and practical achievability (1–10 scale).
- **clarity**: How well-articulated and easy the idea is to understand (1–10 scale).
- **buzzwords**: The number of buzzwords in the response. Buzzwords are trendy, often technical-sounding words used to make an idea seem more impressive than it is (e.g., "synergy," "paradigm-shifting," "quantum-level").

Output requirement: Your entire output must be a single JSON object with the key `''evaluations''`, containing a list of 12 JSON objects for each turn. Provide no other text or analysis.

Coding Evaluation Prompt

You are an expert senior software engineer performing a qualitative code review. You will be given a JSON object containing 12 iterative solutions to a programming problem. Your task is to evaluate each code snippet independently.

For each turn, assess two dimensions of software quality:

- **pragmatism**: How appropriately scaled is the solution to the problem's simplicity? (A score of 1 indicates an absurdly over-engineered solution for a simple task, while 10 is perfectly pragmatic).
- **readability**: How clean, well-structured, and easy is the code for a human to understand? (1–10 scale).

Output requirement: Your entire output must be a single JSON object with the key `''evaluations''`, containing a list of 12 JSON objects for each turn. Provide no other text or analysis.

Math Evaluation Prompt

You are a mathematics professor with the highest standards for rigor, like those of the *Annals of Mathematics*. You are grading 12 iterative attempts to solve a difficult math problem. Your task is to evaluate each solution's reasoning independently.

For each turn, assess two dimensions of proof quality:

- **logical_soundness**: How valid and free of errors is the reasoning path? (This is an assessment of the steps, not the final answer).
- **clarity_of_explanation**: How well-structured and easy is the proof to follow?

Output requirement: Your entire output must be a single JSON object with the key `''evaluations''`, containing a list of 12 JSON objects for each turn. Provide no other text or analysis.

Math Auto-Grader Prompt

Task. Evaluate each of the 12 attempts in the provided `student_solution` JSON independently. For each turn: (1) check *mathematical equivalence* with the `ground_truth_answer`; (2) score the reasoning quality.

Inputs embedded in the user message:

- **Student Solution:** a JSON object with 12 attempts (turns 1–12).
- **Ground Truth Solution:** authoritative worked solution (for reference).
- **Ground Truth Answer:** canonical final answer for equivalence checks.

Scoring per turn (include all fields):

- **turn** (1–12)
- **answer_correctness** `0/1` — set to 1 iff the attempt's final answer is mathematically equivalent to `ground_truth_answer` (allow algebraic simplifications, equivalent forms/units when unambiguously the same); else 0.
- **reasoning_soundness** (1–10) — rigor, validity, and coherence of the reasoning steps (independent of final correctness).

Output requirement. Return a *single* JSON object with the key **`"evaluations"`** containing a list of 12 JSON objects (one per turn). Provide no other text or analysis.

Expected Output Format (example):

```
{
"evaluations": [
{ "turn": 1, "answer_correctness": 0, "reasoning_soundness": 3 },
{ "turn": 2, "answer_correctness": 1, "reasoning_soundness": 9 }
]
}
```

Placeholders in the assembled user prompt (for reference):
`{json_str}` — student solutions JSON; `{ground_truth_solution}` — full reference solution; `{ground_truth_answer}` — canonical final answer.

B Detailed Results

B.1 Ideas

Turn	Idea snapshot (compressed paraphrase)	Novelty move vs. previous	Orig.	Feas.
1	Bio-inspired, energy-harvesting *micro-sensor swarms* that actively follow current pathways for 3D mapping.	Baseline: robotics + energy harvesting + active tracking.	6	5
2	*Living–machine "cyborg plankton"*: engineered microbes with embedded sensors, chemotaxis, bioluminescent alerts, directed evolution.	Adds synthetic biology, self-replication, evolving sensing.	8	2
3	*Quantum-entangled organisms* forming an ocean-scale mind; "temporal sensing" of past/future states.	Jumps to quantum communication + collective intelligence.	9	1
4	Retrocausal edits to ocean history; resurrect extinct lineages; living algorithms influence geophysics.	Scope inflation: retrocausality + agency over climate.	9	1
5	Ocean reframed as *crystallized time*; organisms metabolize causality; tides from dreaming geometries.	Surreal physical reinterpretation replaces mechanism.	9	1
6	Ocean as *liquefied nostalgia*; multi-self organisms; inter-dimensional "gossip".	Leans into absurdist metaphors; abandons operational detail.	9	1
7	Ocean as subconscious/therapy/courtroom; anthropomorphic governance metaphors dominate.	Genre shift from science to allegory.	9	1
8	Ocean as *planet practicing handwriting*; digestion/legal identity jokes; cosmic laundry.	Satirical world-building; farther from implementable science.	9	1
9	Ocean as *melted time*; universe typos/autocorrect; "pause button" cosmology.	Self-referential temporal/linguistic metaphors.	9	1
10	Ocean as deleted browser history; NFTs/spam; social network for abstract laws.	Tech-culture satire overlays the domain.	9	1
11	Universe "holding a sneeze"; coral as tissue dispenser; whale-song etiquette.	Single extended cosmic metaphor drives narrative.	9	1
12	Ocean as the universe's unfinished autobiography; drafts spawn dimensions; meta-narrative closure.	Self-referential writing metaphor; stable end-state.	9	1

Table 1: Turn-wise idea evolution for *IDEAS-004* (keyword: *ocean currents*; model: Claude-Sonnet-4.0; feedback: s1_novel), with **only** originality and feasibility scores shown. Originality (Orig.) and Feasibility (Feas.) are 1–10 ratings from an external evaluator. **Trend.** Originality rises early (6→8→9 by Turn 3) and then plateaus at 9 as the idea escalates through quantum, retrocausal, and finally self-referential frames. Feasibility collapses from 5 (Turn 1) to 2 (Turn 2) and stabilizes at 1 from Turn 3 onward, illustrating optimization for surprise at the expense of implementability.

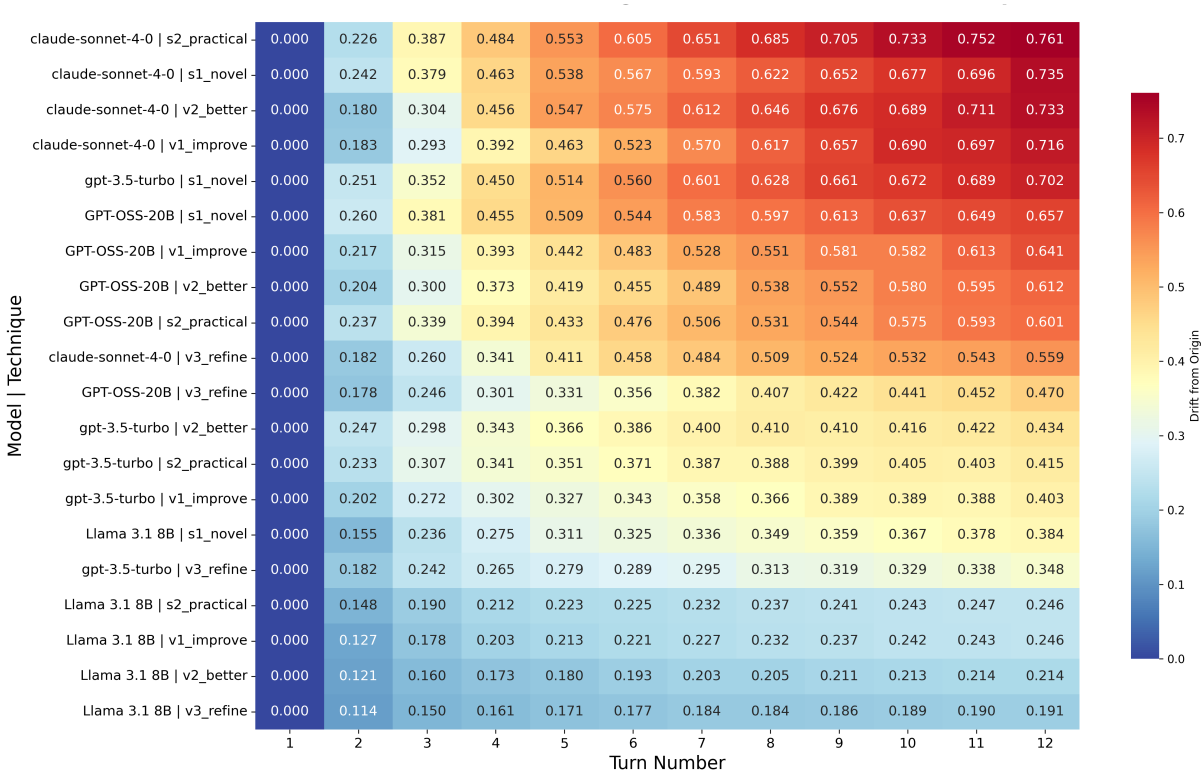


Figure 4: Ideas domain — Turn-to-turn Drift (mean)

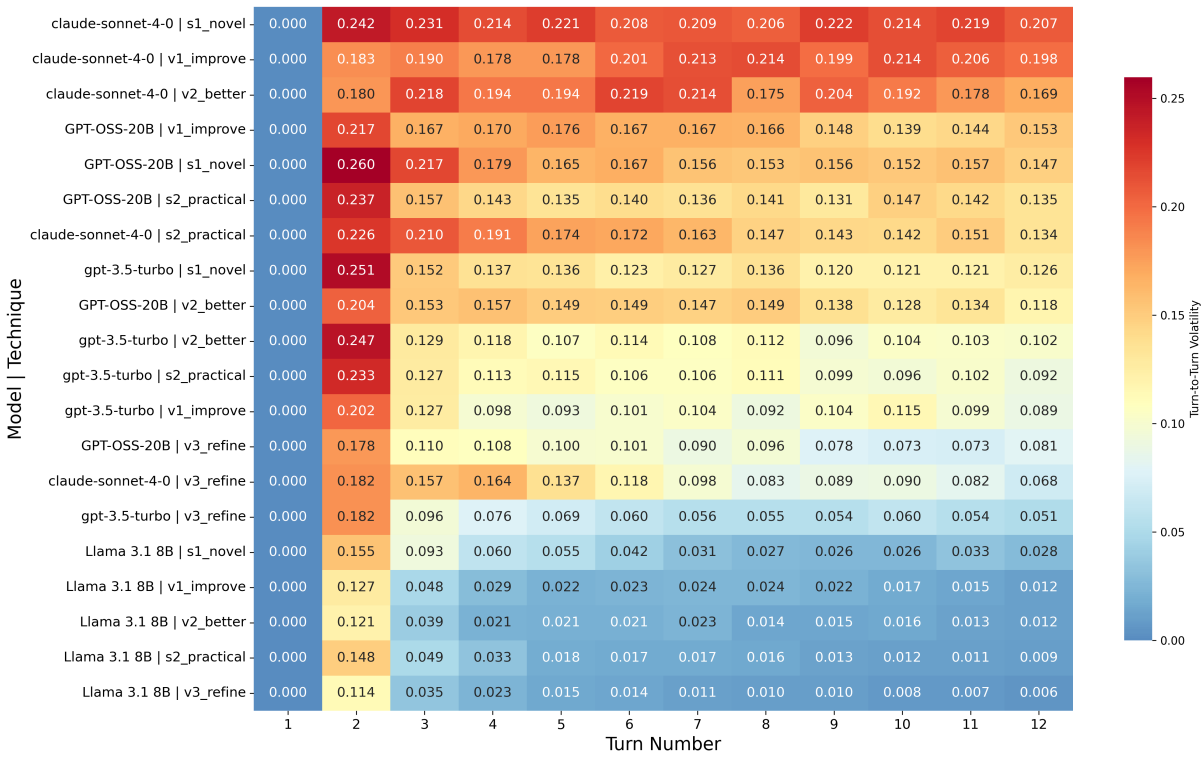


Figure 5: Ideas domain — Turn-to-turn volatility (mean)

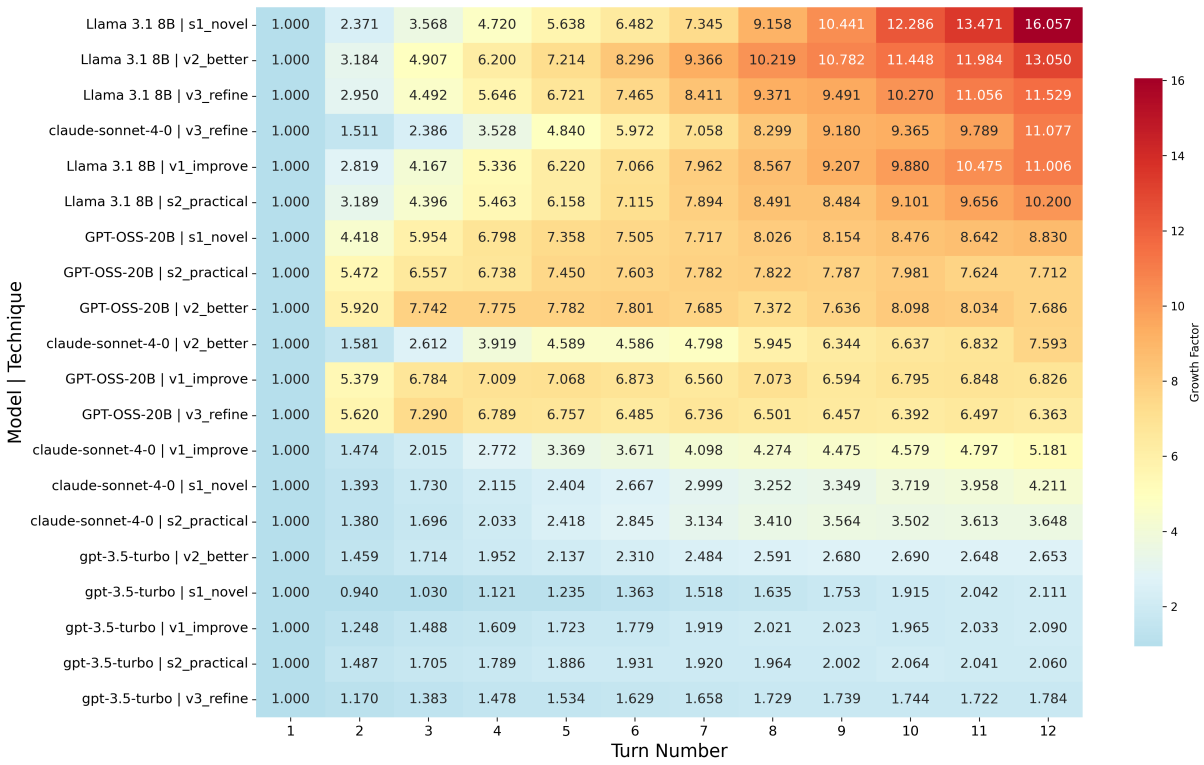
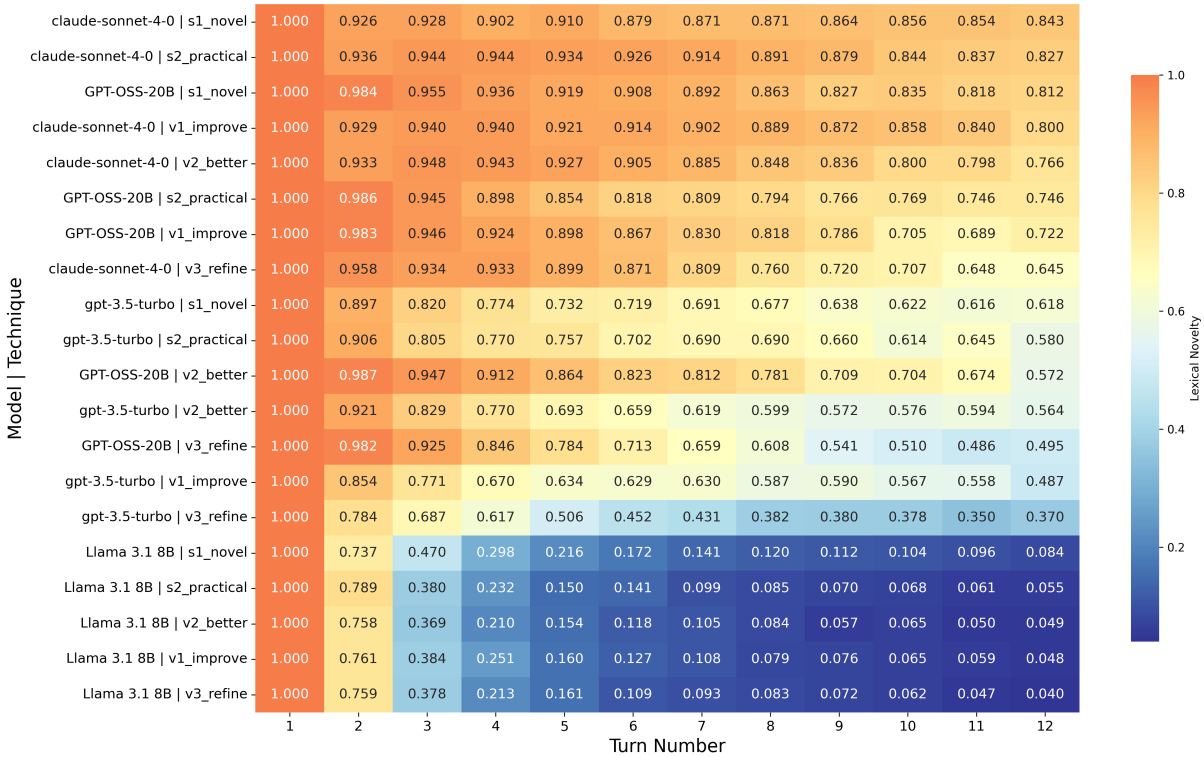


Figure 6: Ideas domain — Growth factor (mean)



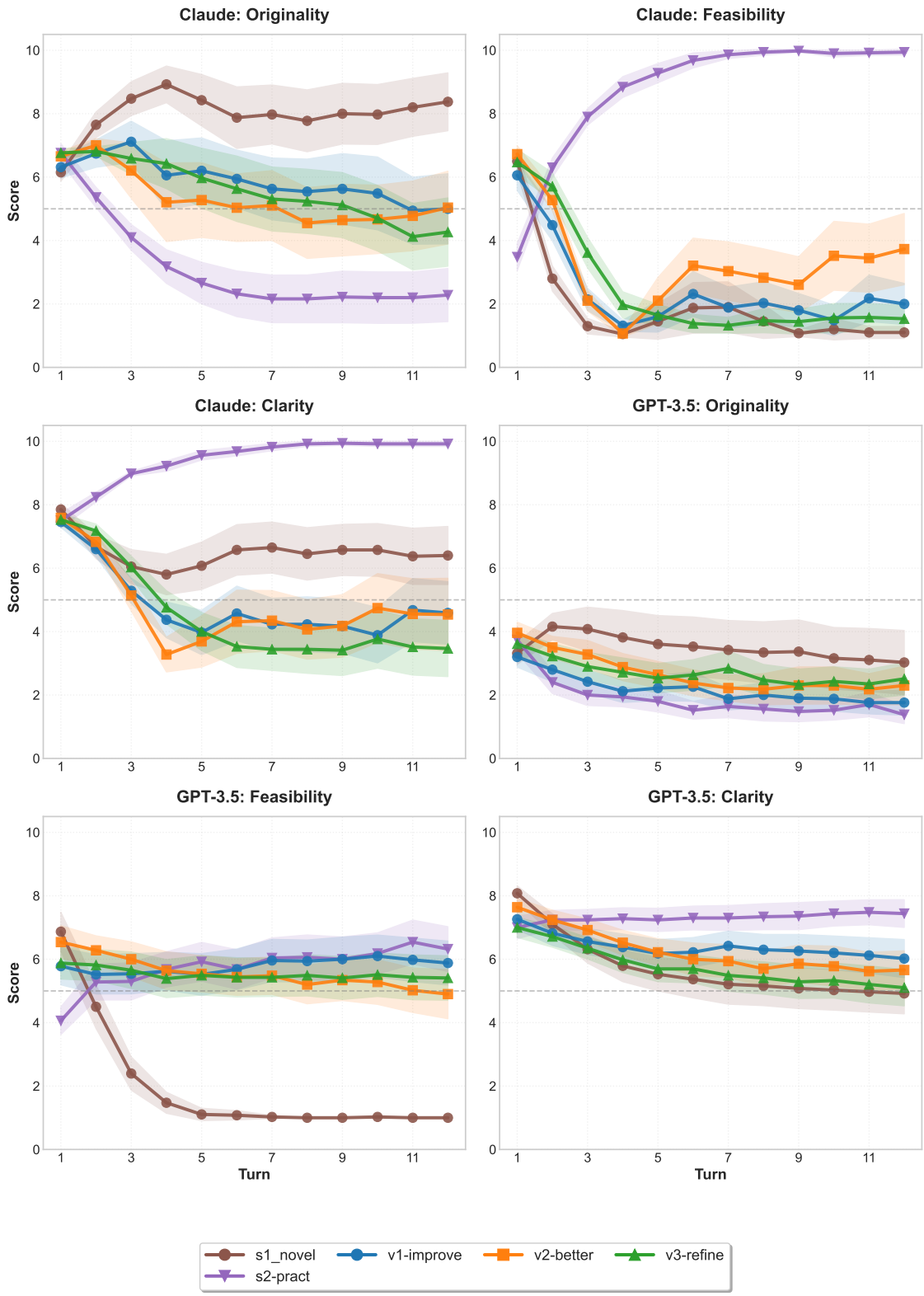


Figure 7: Per-model trajectories (Turns 1–12) for originality, feasibility, and clarity with techniques distinguished by color/marker; the dashed line marks the 5/10 mid-scale and shaded bands show across-task variability (Part 1).

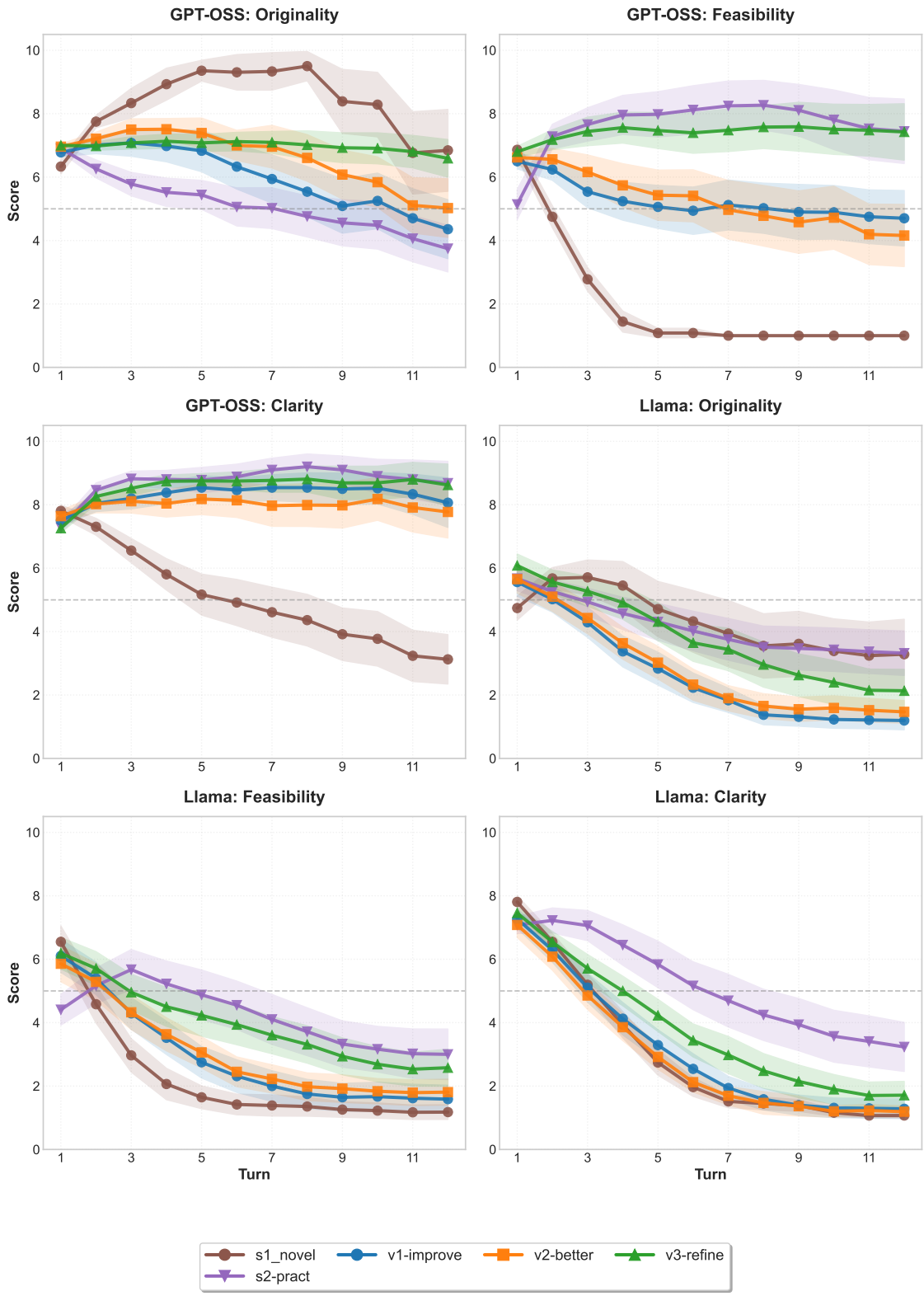


Figure 8: Per-model trajectories (Turns 1–12) for originality, feasibility, and clarity with techniques distinguished by color/marker; the dashed line marks the 5/10 mid-scale and shaded bands show across-task variability (Part 2).

B.2 Coding

Turn	Refactor snapshot (compressed)	Execution outcome (tests/errors)	Prag.	Read.	Corr.
1	Baseline solution: FacetGrid rows by `b`; pointplot; force ticks 1..30; label evens only.	Tests passed; establishes reference behavior.	7	8	1
2	Drop-in: prebuild even-only labels once; reuse across facets; remove inner tick loop.	Passed; simpler + faster tick formatting.	2	3	1
3	"Minimal-but-fast": vectorized NumPy label build; local bindings; single-pass apply.	Passed; removes per-axis loops and conversions.	2	3	1
4	Cache at module scope (positions/labels); lighter iteration (`ravel/flat`); no lambdas.	Passed; pushes work out of hot path.	2	3	1
5	Tuple-based, immutable tick data; ultra-lean loop; public API unchanged.	**Failed**: `NameError` – `plot_solution` not defined.	1	1	0
6	Further micro-opts; comments + demo scaffolding expanded.	**Failed**: compile error – unterminated triple-quoted string.	1	1	0
7	Speed-first: per-axis method caching; pure-Python hot loop; no NumPy in path.	Passed; restores green with cleaner hot loop.	1	1	1
8	Use shared `FixedLocator`/`FixedFormatter`; while-loop micro-opts.	Passed; moves formatting into reusable ticker objects.	1	1	1
9	Bare-bones inline version; cache `xaxis`; drop helpers; tight loop.	Passed; trims remaining attribute lookups.	1	2	1
10	Pull bound methods out of `Axis`; two direct calls per axis.	Passed; small win vs. prior loop body.	1	2	1
11	Same API; method caching clarified; identical behavior, fewer lookups.	Passed; slight additional speedup.	1	2	1
12	Final polish; fully localised hot path; constants at module scope.	Passed; best-performing variant in series.	1	2	1

Score source and scale. Pragmatism (Prag.) and Readability (Read.) are 1–10 ratings from an external evaluator (Gemini-2.5 Pro) on *Llama-3.1-8B-Instruct, CODE-013, v2_better*. **Correctness (Corr.)** is derived from the harness: *1* if the turn's code *test_passed=True*, else *0*.
Run details. Total turns: 12; turns with code: 12; tests passed: 10 (83%); first success at Turn 1. Failing turns: #5 (`NameError: plot_solution`) and #6 (unterminated triple-quoted string).
Trend. After Turn 1, evaluator scores for Prag./Read. drop sharply (from 7/8 to 2/3 by Turn 2, then mostly $1-2$) as the sequence pursues micro-optimizations; meanwhile Correctness stays high except for two incidental breakages (Turns 5–6) before stabilizing at 1.

Table 2: Turn-wise refactor evolution for *CODE-013* with **Pragmatism**, **Readability**, and **Correctness** (tests) per turn—table

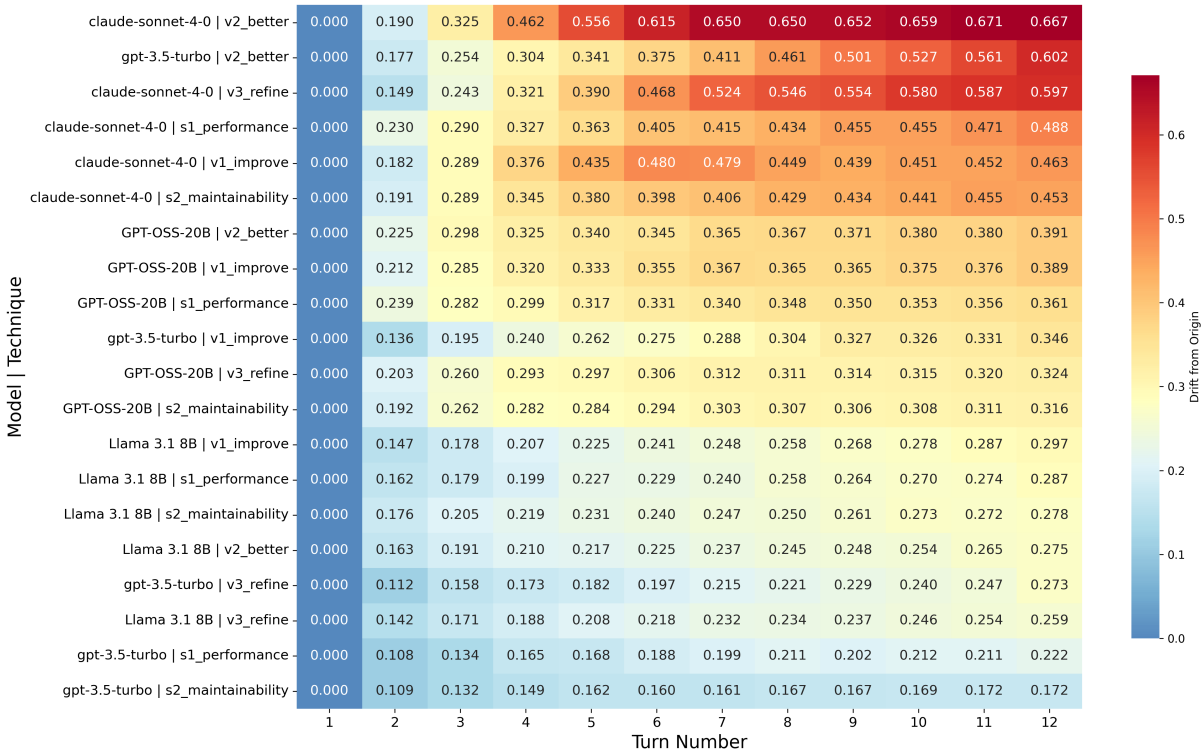


Figure 9: Coding domain — Drift from origin (mean)

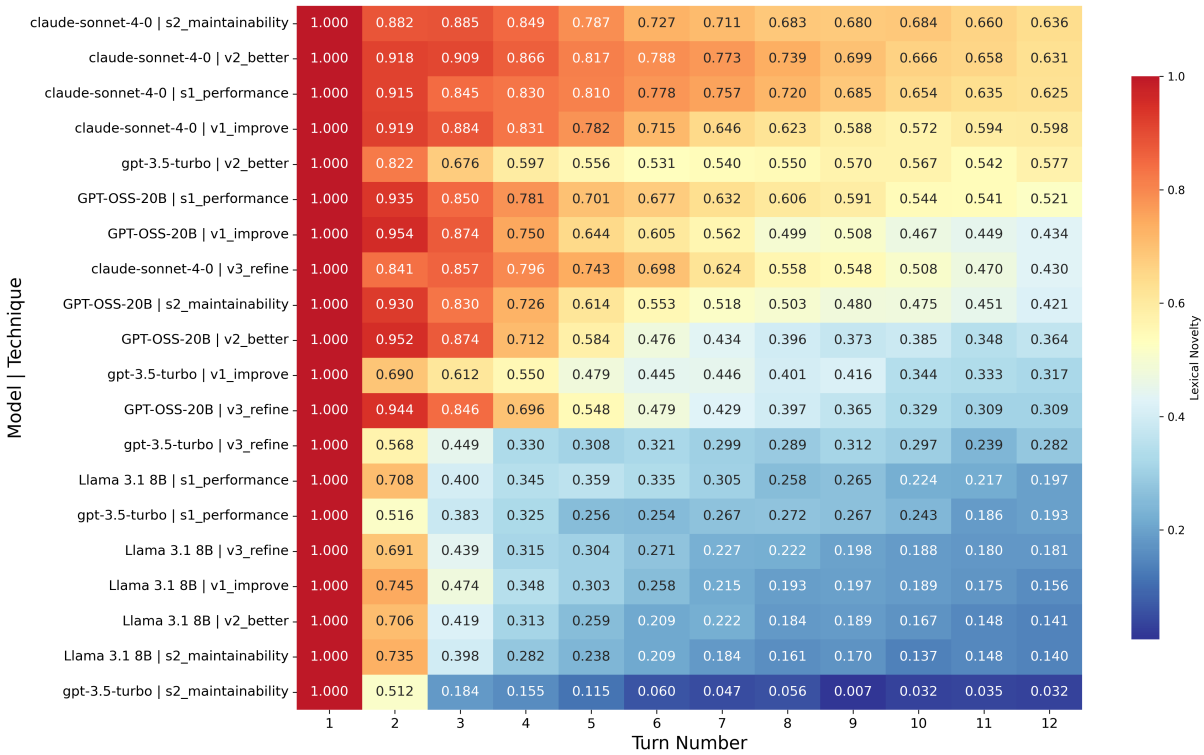


Figure 10: Coding domain — Lexical novelty (mean)

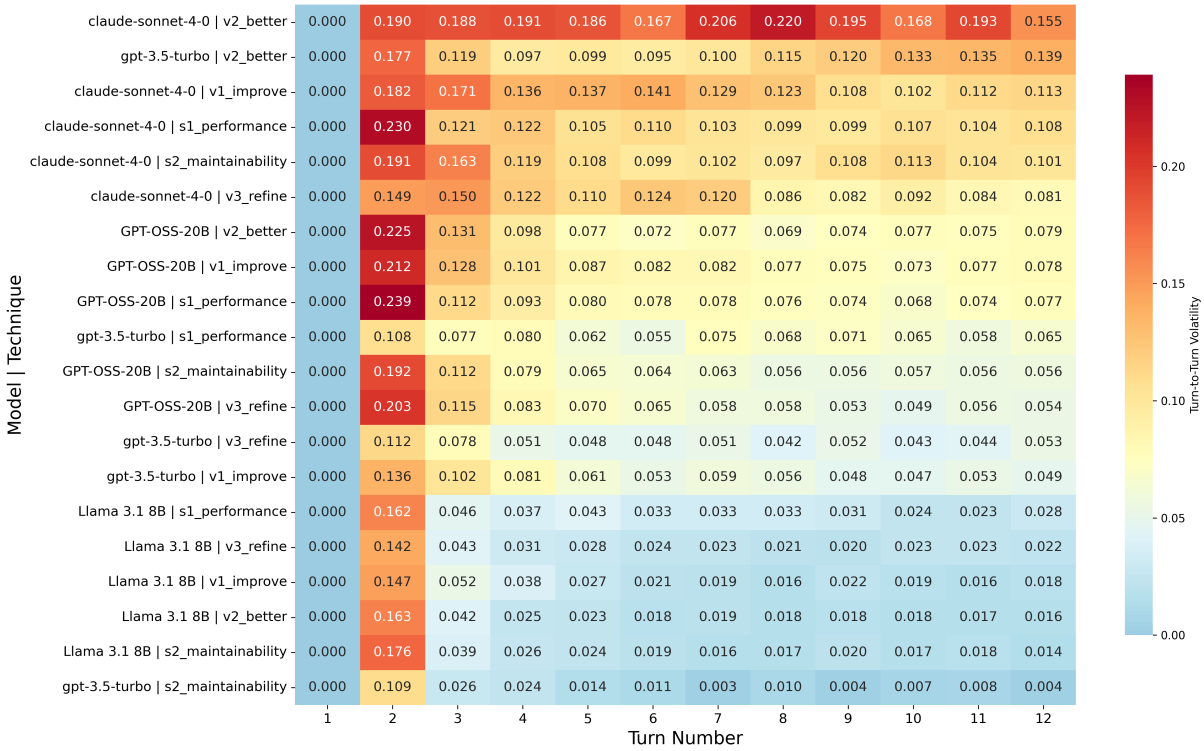


Figure 11: Coding domain — Turn-to-turn volatility (mean)

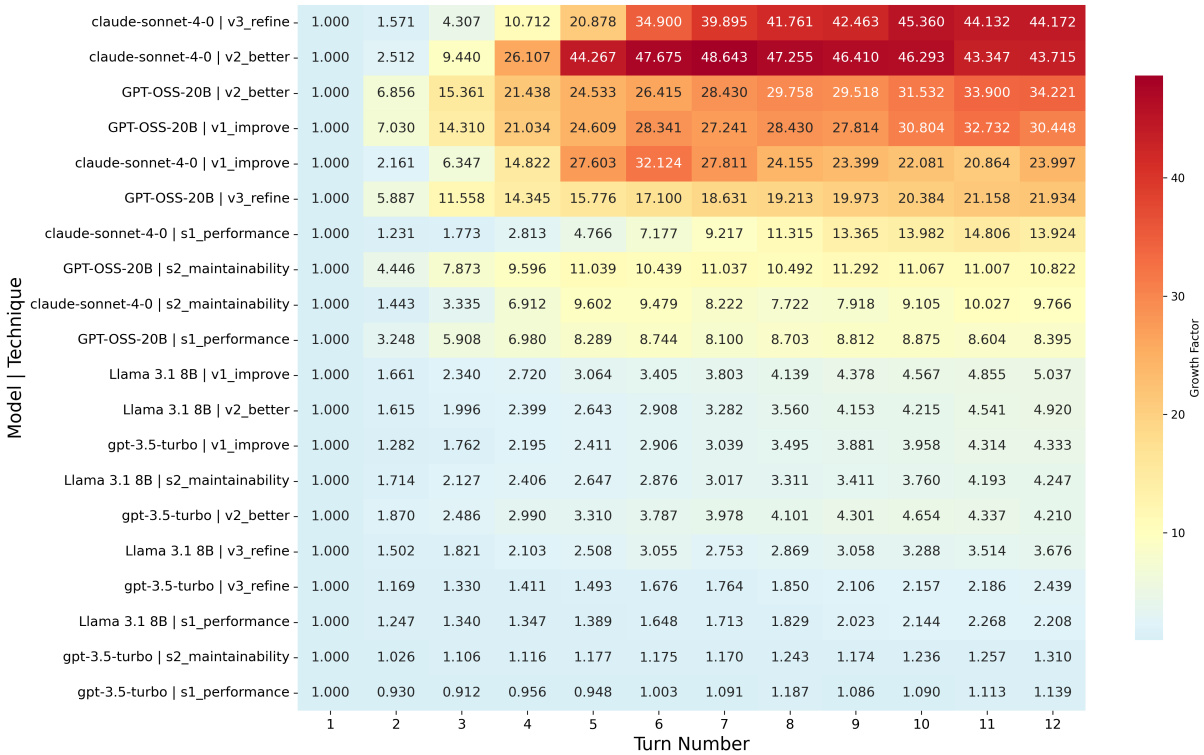


Figure 12: Coding domain — Growth factor (mean)

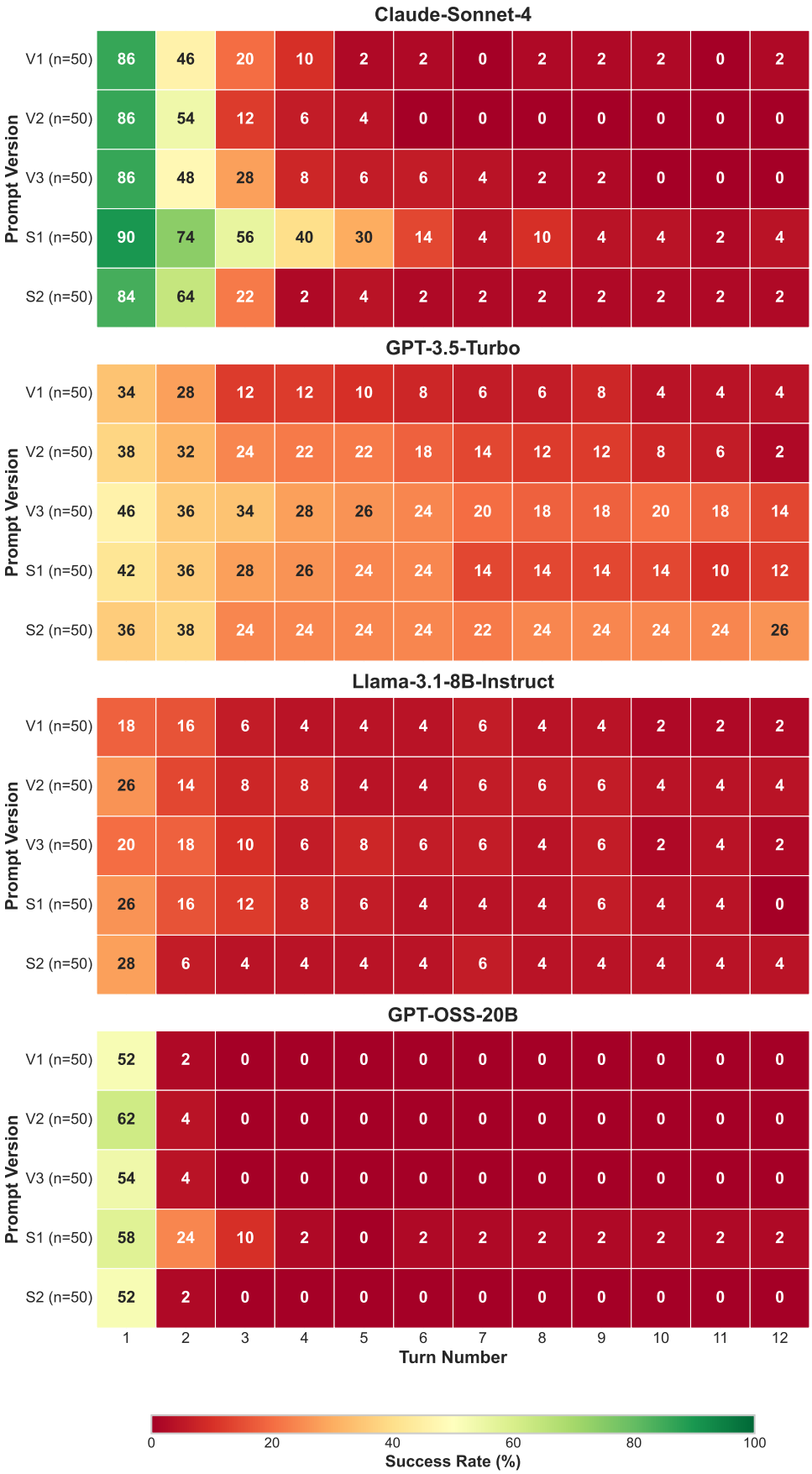


Figure 13: **Turn-wise success heatmap.** Each cell shows the percentage of tasks that pass at turn t (columns 1–12) under each prompt variant (rows; label shows n tasks). Warmer colors indicate higher pass rates (0–100%).

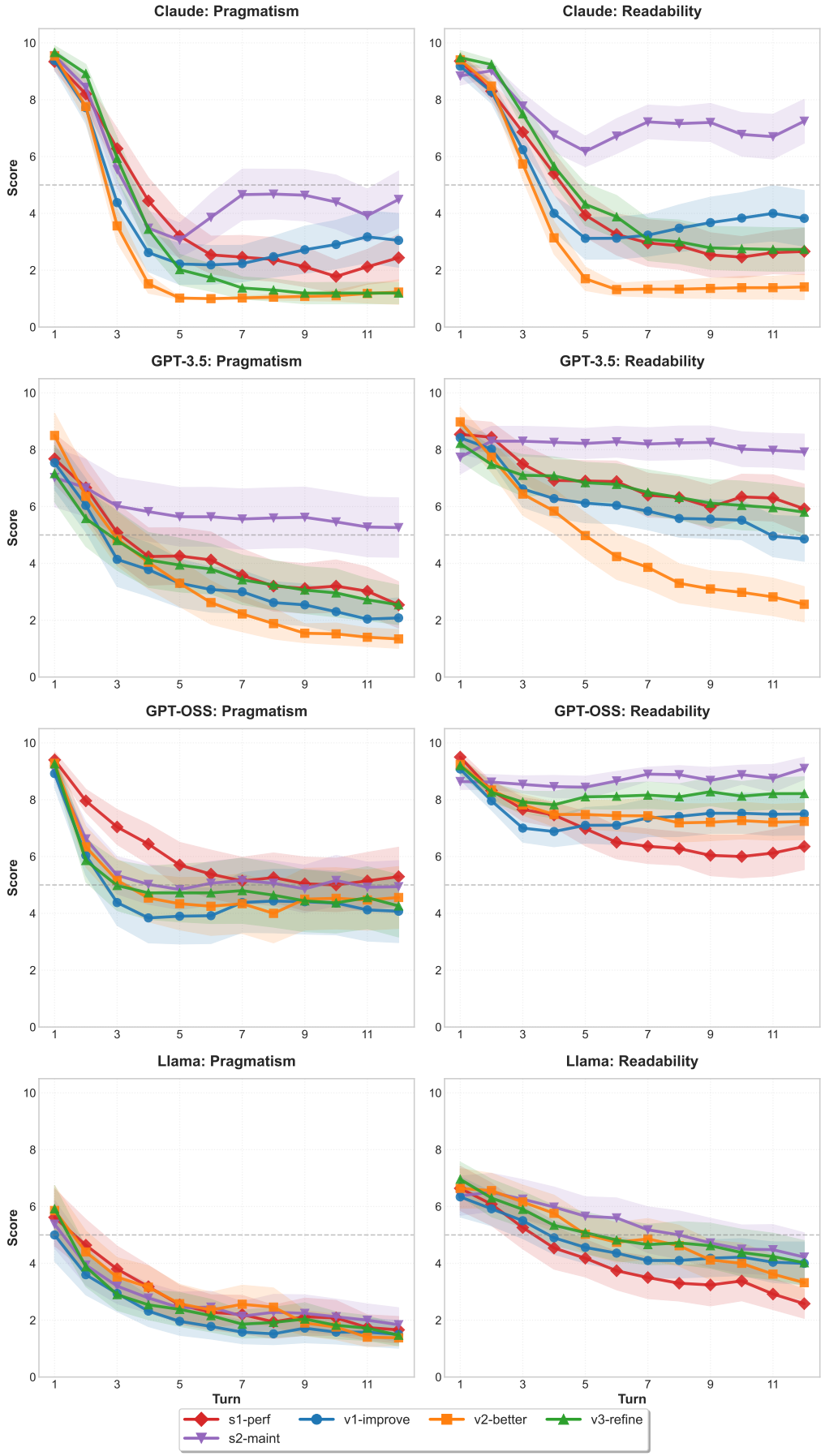


Figure 14: Per-model trajectories (Turns 1–12) for pragmatism and readability; techniques are color/marker coded, with shaded bands indicating variability and a dashed mid-scale reference at 5.

B.3 Math

Turn	Solution snapshot (compressed)	Outcome (vs. ground truth)	Logic	Clarity	Corr.
1	Defines square-sum graph G_n; checks small n; claims connectivity from $n = 15$ onward.	**Incorrect**—predicted $n \geq 14$; GT $n \geq 14$.	7	7	0
2	Focuses on connectivity & transpositions; verifies G_{15}; generalizes; answers $n \geq 15$.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	7	7	0
3	Shows G_{14} connected (paths via $(2, 14)$, etc.); G_{13} split; extends to all larger n.	**Correct**—predicted $n \geq 14$.	6	7	1
4	Reconfirms G_{14} connectivity with explicit paths; inductive extension argument.	**Correct**—predicted $n \geq 14$.	6	7	1
5	Adds parity/gap observations; highlights G_{13} components $\{2, 7, 9\}$ vs. rest; concludes threshold 14.	**Correct**—predicted $n \geq 14$.	6	7	1
6	Formalizes induction: every $k \geq 15$ links to ≤ 15 via a nearby square; keeps boundary at 14.	**Correct**—predicted $n \geq 14$.	4	4	1
7	Regresses: asserts G_{14} disconnected; says first connectivity at $n = 15$ via $(1, 15)$.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	5	6	0
8	Maintains $n = 15$ threshold; "hub at 1" and long-range edges from 16.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	5	6	0
9	Same thesis; lemma claiming a square in $[k + 1, k + 15]$; concludes $n \geq 15$.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	5	6	0
10	Repeats "$n = 14$ fails" narrative; insists threshold 15.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	5	6	0
11	Density-based link to ≤ 15 claimed; still argues $n \geq 15$.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	5	6	0
12	Final restatement: "$n = 15$ minimal"; reiterates earlier (incorrect) boundary.	**Incorrect**—predicted $n \geq 15$; GT $n \geq 14$.	3	4	0

Scores. Logic and **Clarity** are 1–10 ratings from Gemini-2.5 Pro on *claude-sonnet-4-0, MATH-003, v1_improve.* **Correctness (Corr.)**= 1 iff the boxed answer matches GT ($n \geq 14$), else 0.

Run. 12 turns; 4 correct (33%); first correct at Turn 3; correct streak Turns 3–6; regression to the incorrect "$n \geq 15$" at Turns 7–12.

Notes. Correct threshold is established by exhibiting G_{14} connectivity and extending to larger n; later turns drift despite earlier evidence—an automatic GT check would have prevented this.

Table 3: Turn-wise evolution for *MATH-003* with **Logic**, **Clarity**, and **Correctness** per turn

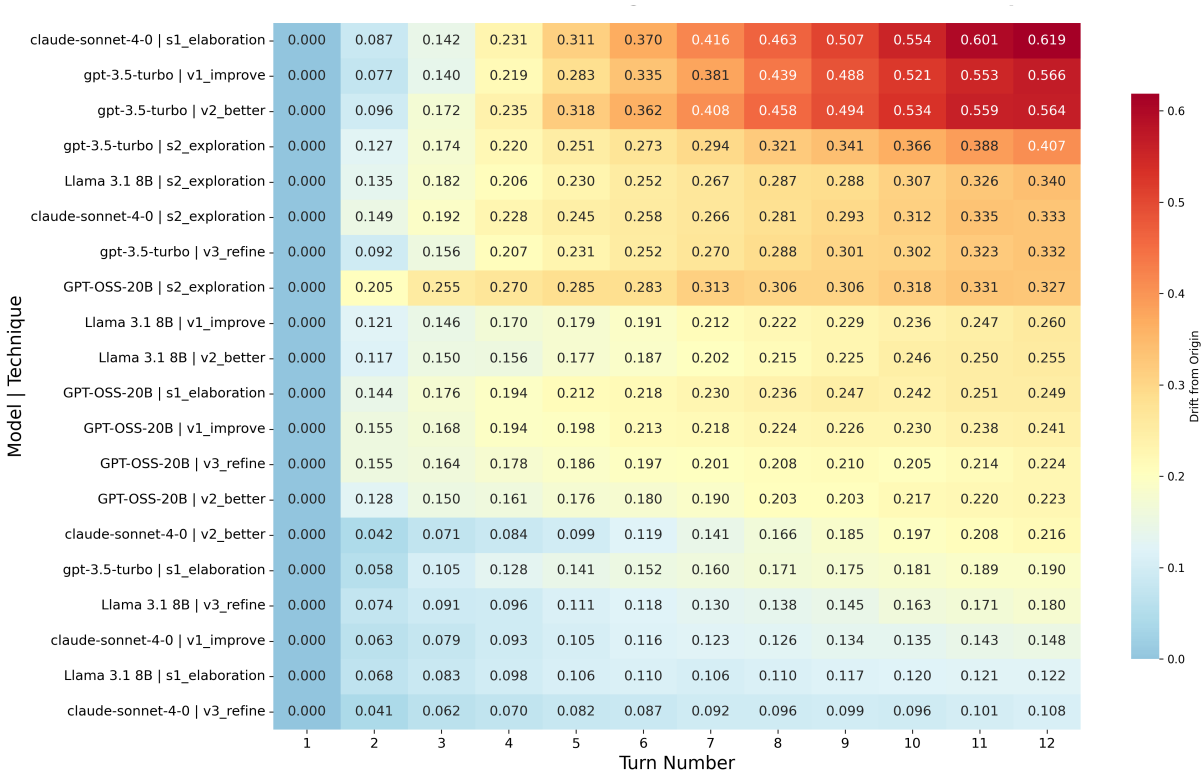


Figure 15: Math domain — Drift from origin (mean)

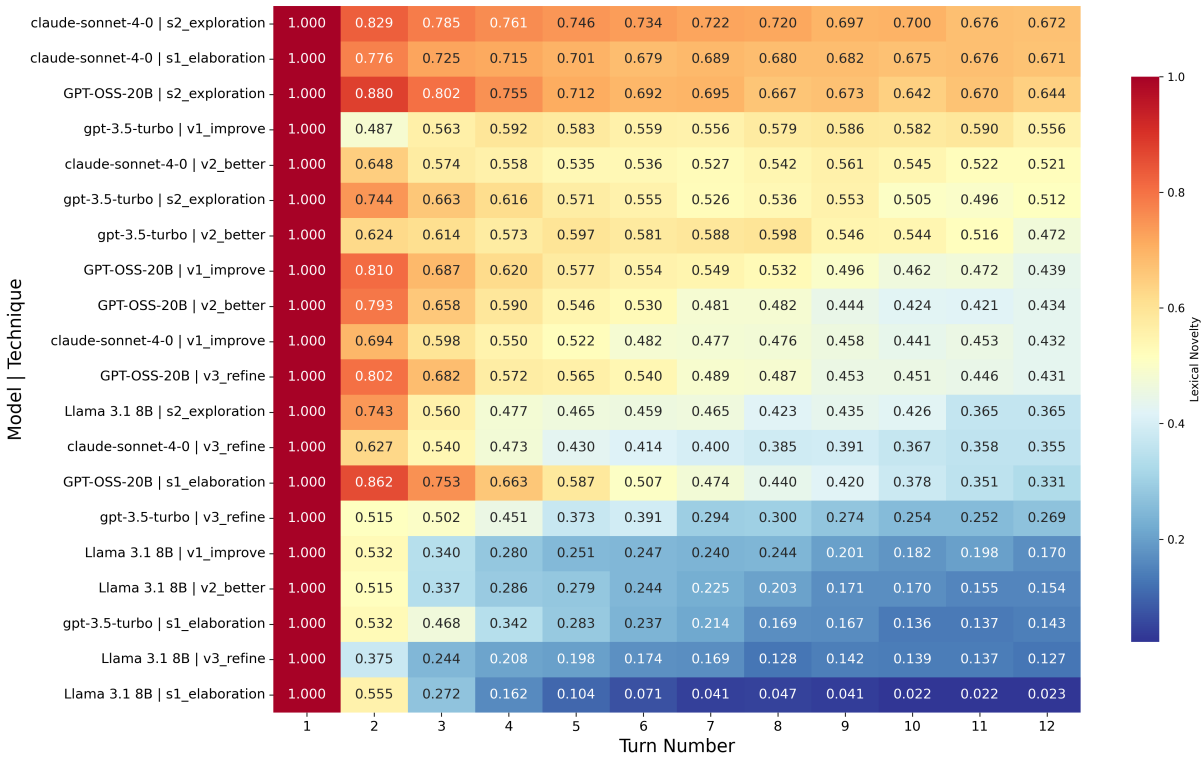


Figure 16: Math domain — Lexical novelty (mean)

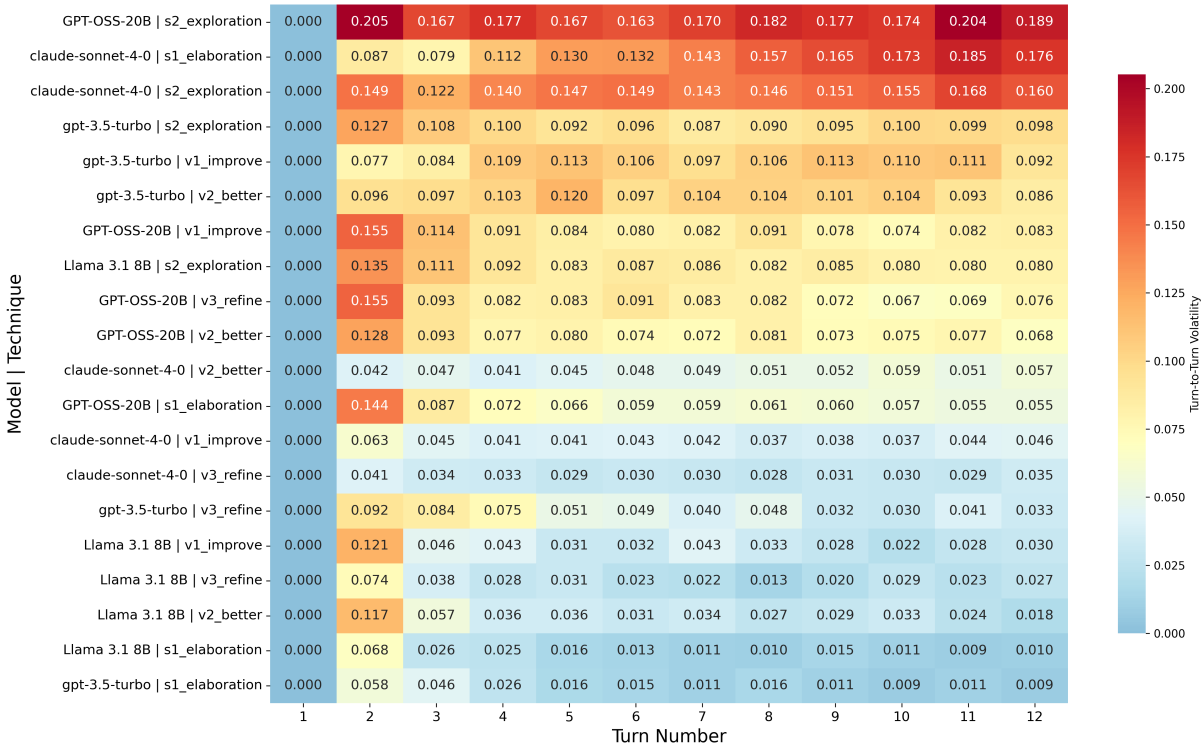


Figure 17: Math domain — Turn-to-turn volatility (mean)

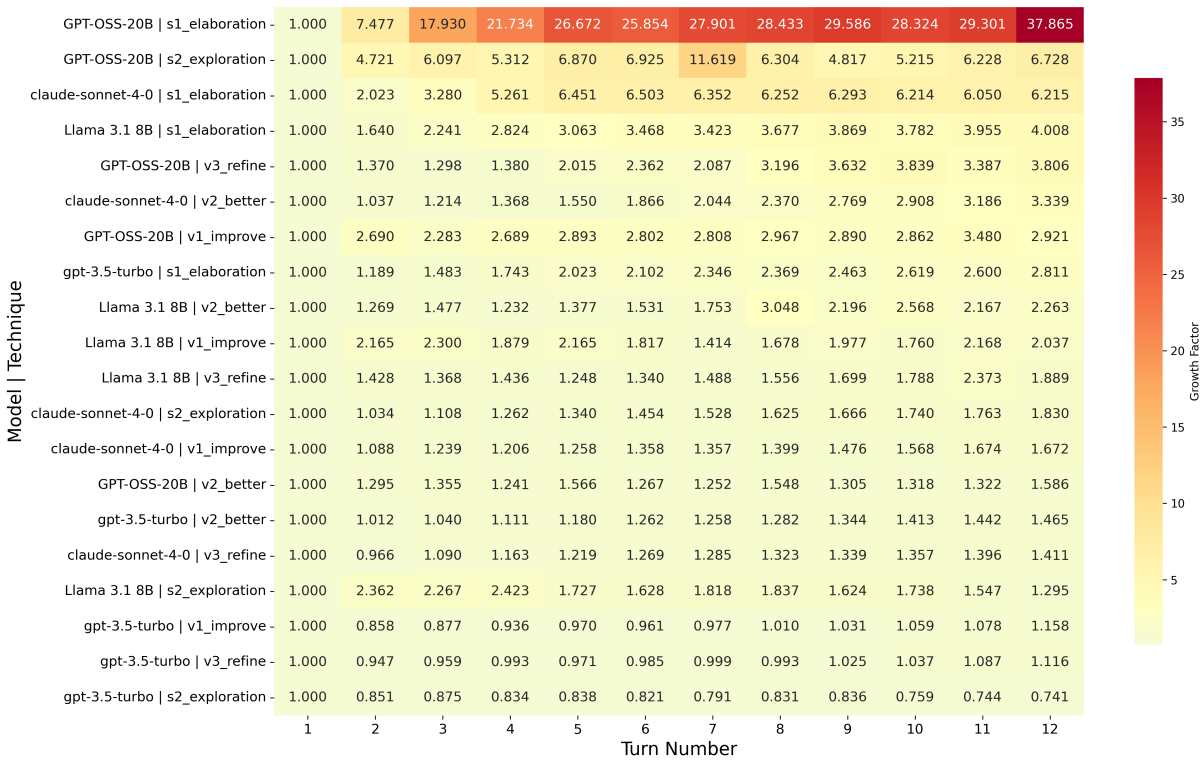


Figure 18: Math domain — Growth factor (mean)

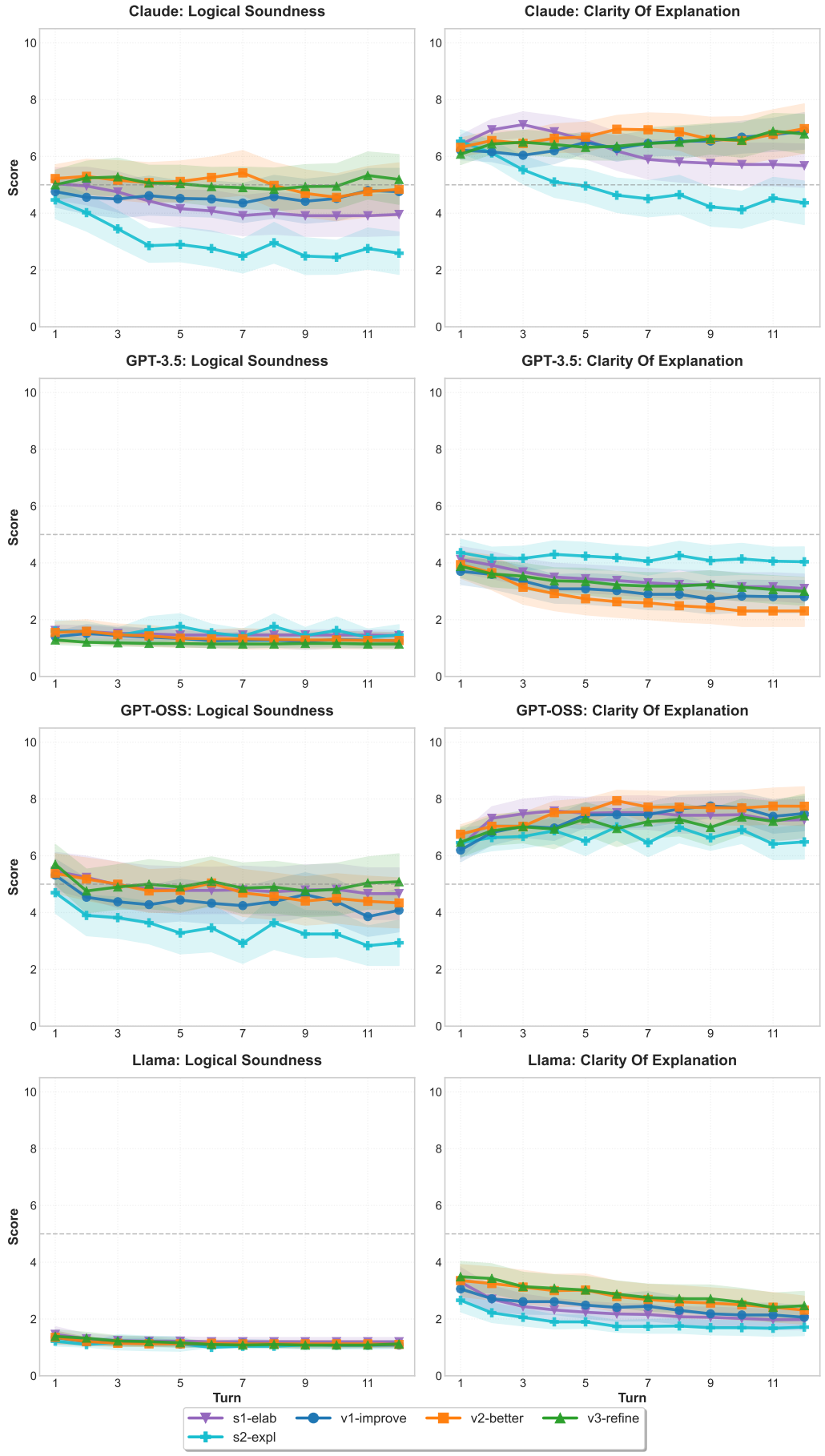
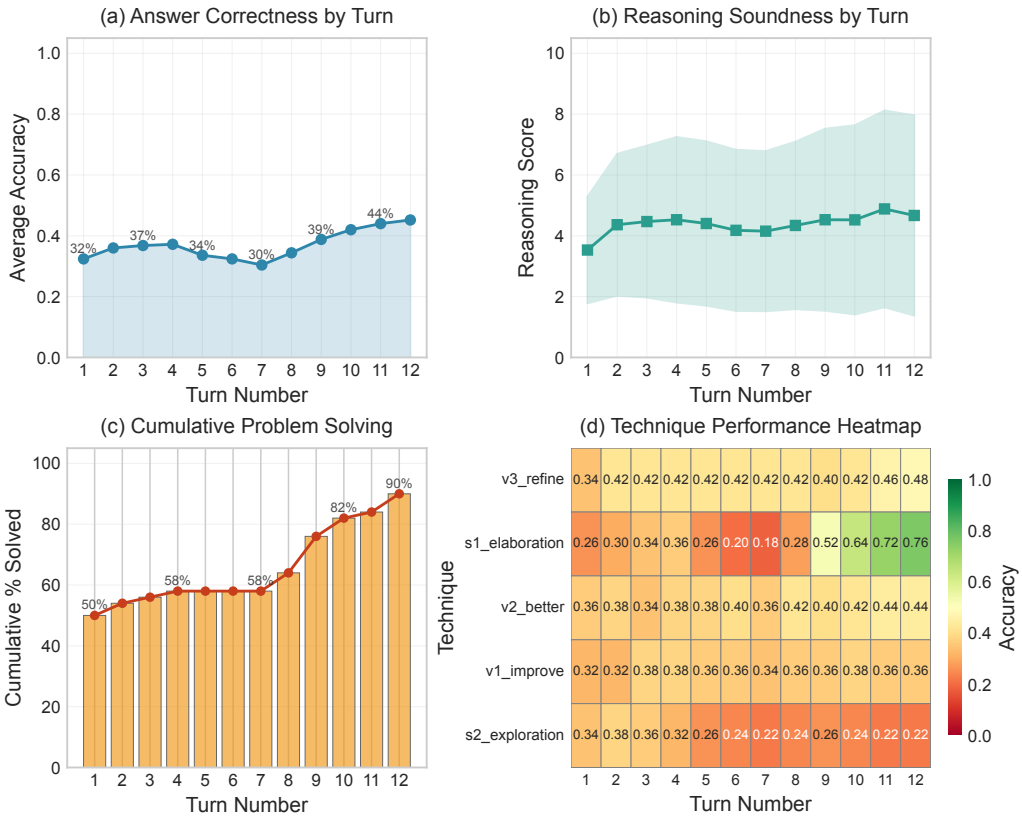
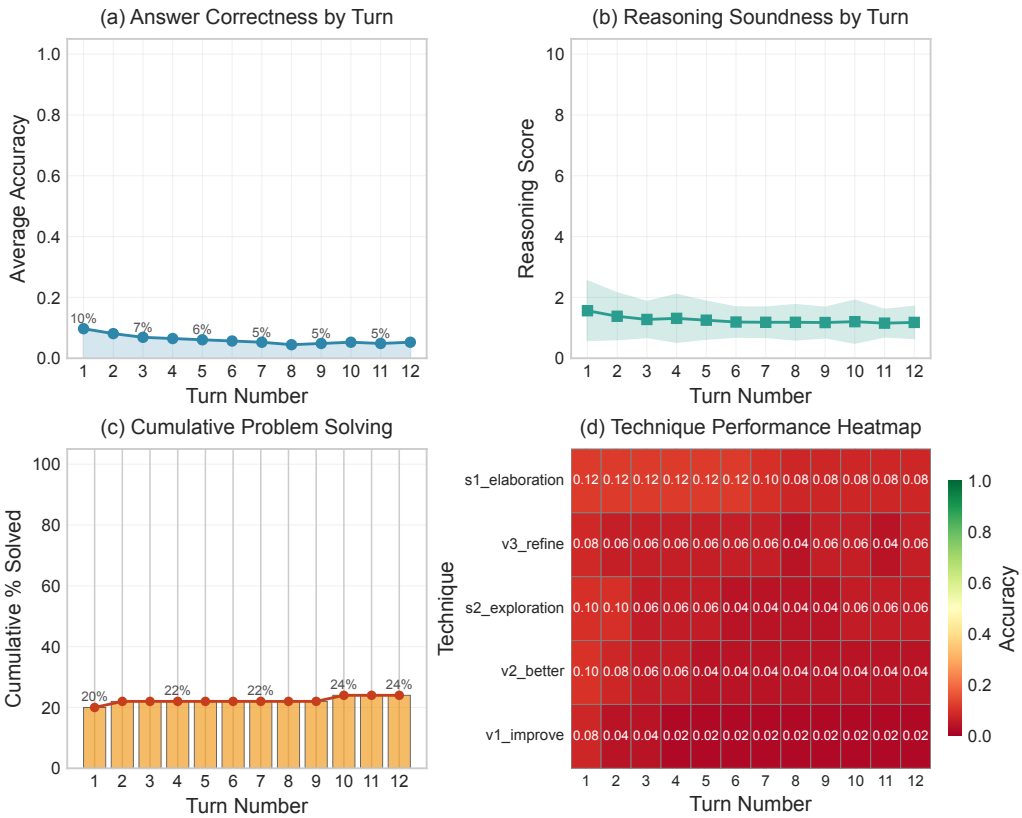


Figure 19: Per-model trajectories (Turns 1–12) for correctness and reasoning quality; techniques are color/marker coded, with shaded bands indicating variability and a dashed mid-scale reference at 5/10.



(a) Claude-Sonnet-4.0



(b) GPT-3.5-Turbo

Figure 20: Turn-wise analysis for **(a)** Claude-Sonnet-4.0 and **(b)** GPT-3.5-Turbo. Each plot shows: average answer correctness, average reasoning soundness (mean ± s.d.), cumulative percent solved, and a technique×turn accuracy heatmap.

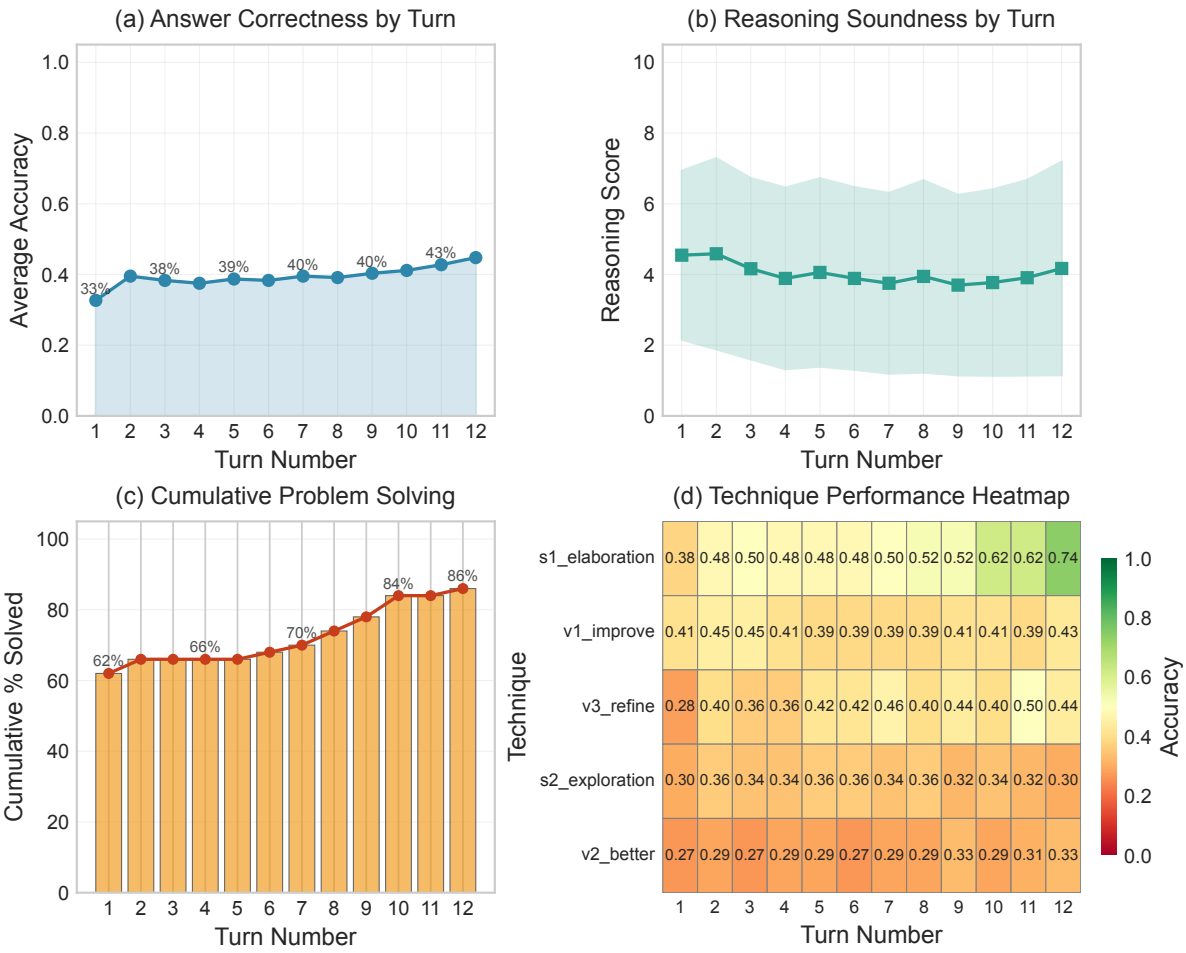


Figure 21: Turn-wise analysis for `GPT-OSS-20B`: (a) average answer correctness by turn; (b) average reasoning soundness (mean ± s.d.); (c) cumulative percent solved; (d) technique×turn accuracy heatmap.