Autonomous Research Pipelines: Optimal Design, Empirical Stress-Test, and the Correlation Problem

Anonymous

March 2026

Abstract

An open-source pipeline of 12 AI agents autonomously discovers a research problem, builds a theory, verifies it adversarially, and writes a publication-ready paper. Two complete runs (5 hours each) produced real paper drafts, generating rich performance data: 19 math audits (16 pipeline + 3 post-pipeline, all post-pipeline FAIL), 10 scorer evaluations (peak score 69/100), and 765 LLM-on-LLM error-detection experiments. A formal framework from inspection economics characterizes the design space. Three results emerge. First, optimal gate ordering depends on detection-rate-to-cost ratios, not intuitive sequencing; the pipeline gets this mostly right. Second, error correlation is the binding constraint: when the same model generates and evaluates, compound evaluation degrades predictably. A shared blind spot where 0/30 agents detected an error under structured evaluation provides direct evidence, though the same model detects it at 60% under an alternative protocol. Third, a larger model catches more errors (120b: 85% vs. 20b: 56%, $p < 0.001$), but if shared across roles, introduces correlation that erodes the gains. The framework yields concrete improvements to gate ordering, evaluator assignment, and stopping rules. The paper closes the loop: its improvement proposals feed back into the next pipeline version.

Keywords: autonomous research, AI agents, inspection economics, error correlation, pipeline design

JEL: O31, D83, C63

1 Introduction

Large language models can write proofs, search literatures, and draft papers. Can they do all of this autonomously, in sequence, producing a publication-ready research paper without human intervention? And if so, how should such a system be designed?

This paper makes three contributions.

First, the system. An open-source pipeline of 12 specialized AI agents autonomously discovers a research problem, generates candidate mechanisms, develops formal theory, audits the mathematics adversarially, checks novelty against the existing literature, stress-tests the results, writes a paper, and simulates peer review. Eight stages and six quality gates enforce standards at every transition. Anyone can fork the repository and run it.[1] Two complete runs, each approximately five hours, produced real paper drafts in finance theory. This is a practical contribution: the system exists and works.

Second, the economics of its design. The pipeline's architecture embeds implicit choices about agent ordering, evaluator assignment, and stopping rules. Are these choices optimal? What would optimal even mean? Drawing on inspection economics (Dorfman, 1943), optimal stopping theory, and information theory, a formal framework characterizes the design space. Six results emerge, the last two of which are new to the inspection economics literature:

(i) *Gate ordering.* Optimal sequencing depends on the ratio of each gate's detection rate to its cost, not on intuitive heuristics like "easy checks first." The pipeline gets this mostly right, but not entirely: the math audit should precede idea review in some configurations.

(ii) *Error correlation (ρ).* When the same model generates content and evaluates it, errors compound predictably. A shared blind spot where 0 of 30 agents detected an error provides direct evidence that $\rho > 0$. Compound evaluation under correlated errors degrades as $\rho + (1 - \rho)(1 - d)^n$ rather than $(1 - d)^n$, where d is the single-evaluator detection rate and n is the number of evaluators.

(iii) *Capability–independence frontier.* A larger model catches more errors but, if shared across generation and evaluation roles, introduces correlation. A threshold condition characterizes when splitting to a weaker but independent evaluator dominates. The data support capability over independence for current models (120b: 85% correct detection vs. 20b: 56%, $p < 0.001$), but the gap narrows for correlated errors.

[1] Available at `[redactedforreview]`.

(iv) *Escalation as optimal stopping.* Human review enters as a costly final gate. The framework characterizes when paying for human inspection dominates continued autonomous iteration, yielding an optimal escalation policy.

(v) *Goodhart distortion.* When generators adapt to evaluation gates, the error distribution shifts toward hard-to-catch types. This distortion compounds across stages: effective detection rates *decrease* through the pipeline even with constant nominal rates. This is not the same as correlation ($\rho > 0$); it is a distributional shift caused by the evaluation structure itself.

(vi) *Optimal pipeline depth.* Adding stages reduces undetected errors but destroys information (rejected drafts lose correct insights entangled with incorrect ones) and costs more. A closed-form expression for the optimal number of stages K^* shows the current pipeline (6 gates) is slightly deeper than optimal, and that correlation makes pipelines optimally shallower.

Third, empirical stress-test and improvement path. Two full autonomous runs provide ground-truth performance data: 19 math audits (10 pass, 6 fail in-pipeline; plus 3 post-pipeline audits, all fail), 10 scorer evaluations (peak score 69/100, never reaching the 75 threshold), 25 ideas generated with an 8% survival rate to final papers, and 765 LLM-on-LLM error-detection experiments. Confronting the formal framework with observed gate pass/fail rates, error types, and costs reveals where the theory fits and where it misses. Concrete design improvements follow: specific changes to gate ordering, evaluator assignment, and stopping rules that the framework predicts should help. The paper is itself an iteration; it identifies what version 2 of the pipeline should change.

The meta-angle. This paper was partially written using the system it analyzes. The AI workflow description required by the conference is the paper itself. The improvement proposals feed back into the next pipeline version, closing the loop.

Related work. This paper connects four literatures. (i) *Autonomous AI for science.* Lu et al. (2024) build "The AI Scientist" and propose benchmarks for automated research; Si et al. (2024) show LLMs can generate novel research ideas rated comparably to human experts. Our contribution differs: rather than asking whether AI *can* do research, we ask how the research pipeline should be *designed*, providing a formal framework for optimal architecture. (ii) *LLM-as-judge.* Oh et al. (2024) study LLM-on-LLM evaluation empirically but do not model the correlation structure that arises when the same model generates and evaluates.

Our framework formalizes this correlation and derives its consequences. (iii) *Inspection economics.* Dorfman (1943) originated the theory of optimal group testing; Squeglia (2008) extends it to acceptance sampling. We import these ideas into AI pipeline design, where "inspection" is evaluation and "defects" are errors in generated content. (iv) *Ensemble diversity.* Dietterich (2000) shows that ensemble accuracy depends on classifier diversity as much as individual accuracy. Our capability–independence frontier (Proposition 3) formalizes an analogous trade-off for evaluator assignment in sequential pipelines, with the added complication that generator–evaluator correlation creates a floor on compound evaluation that ensemble methods typically do not face.

The remainder proceeds as follows. Section 2 describes the system. Section 3 develops the formal framework. Section 4 presents the empirical evidence. Section 5 translates the framework into concrete design changes. Section 6 assesses limitations. Section 7 concludes.

2 The Pipeline

2.1 Architecture Overview

The pipeline transforms an empty directory into a publication-ready paper through eight sequential stages, each separated by a quality gate. Figure 1 illustrates the flow.

2.2 Agents

Twelve specialized agents perform distinct roles. Table 1 summarizes each agent's function, evaluation scope, and model assignment.

Three features of this design deserve attention.

The pipeline mirrors academic peer review. The stage structure parallels the standard academic workflow: problem identification (literature review), idea development (research meetings), formal modeling (writing), internal review (seminars), and external evaluation (journal submission). The difference: every step is automated and every transition is gated. The pipeline compresses what typically takes months into hours, while preserving the adversarial evaluation structure that makes academic publishing work.

Separation of generation and evaluation. The pipeline never asks an agent to evaluate its own output in the same context. The theory generator writes; the math auditor checks. The paper writer drafts; the referee reviews from a cold context with no knowledge of the development process. This separation is a design choice motivated by the correlation problem formalized in Section 3.

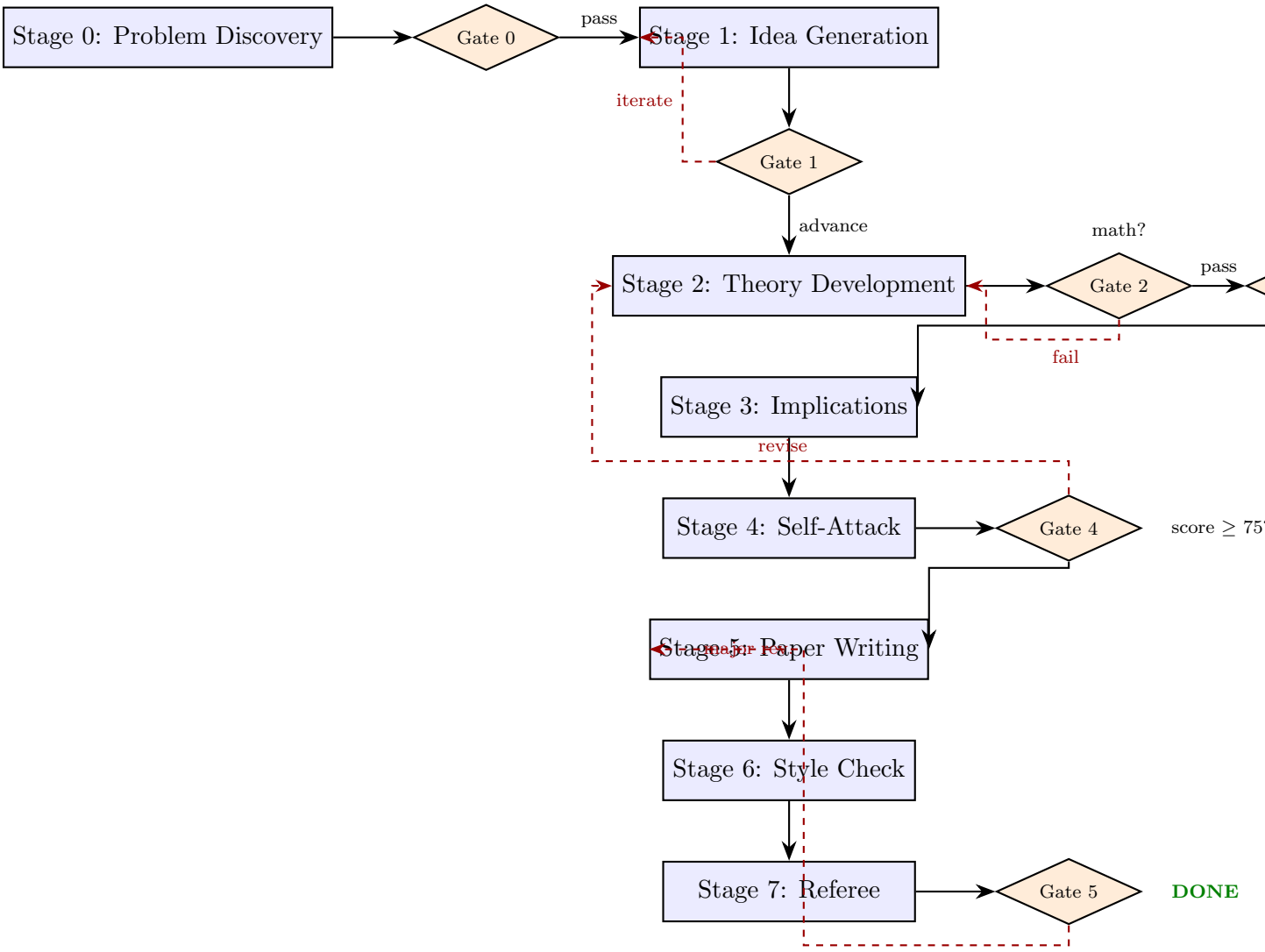

Figure 1: Pipeline architecture. Solid arrows show forward progression; dashed arrows show feedback loops. Six gates enforce quality at each transition. Escalation rules (Table 2) prevent infinite loops.

Table 1: Agent roster

Agent	Stage	Model	Role	Evaluation Scope
Literature Scout	0	Sonnet	Problem discovery	Web search for gaps, puzzles, open questions
Idea Generator	1	Opus	Mechanism brainstorming	3–5 candidate mechanisms per round
Idea Reviewer	1 (Gate)	Opus	Idea ranking and selection	Novelty, tractability, importance, clarity
Theory Generator	2	Opus	Formal model development	Full proofs, comparative statics, intuition
Math Auditor	2 (Gate)	Opus	Adversarial math verification	Re-derives every equation from scratch
Novelty Checker	2 (Gate)	Sonnet	Literature deduplication	10+ targeted web searches per theory
Self-Attacker	4	Opus	Weakness identification	Six attack vectors, severity ranking
Scorer	4 (Gate)	Opus	Holistic quality assessment	5 hard requirements + 5 scored dimensions
Paper Writer	5	Opus	LaTeX paper production	Full paper with citations and appendix
Style	6	Sonnet	Style enforcement	44 rules (voice, filler, precision)
Referee	7	Opus	Simulated peer review	Cold read, no prior context
Scribe	All	Sonnet	Background documentation	Decisions, dead ends, process narrative

Adversarial evaluation. The math auditor re-derives every equation from scratch rather than checking the presented proof. The self-attacker explicitly searches for paper-killing flaws across six attack vectors. The scorer checks five hard binary requirements before computing a weighted score. These agents are instructed to find errors, not to confirm quality.

Escalation with bounded recursion. Feedback loops connect later stages back to earlier ones, but each loop has a hard recursion limit (Table 2). After exhausting retries, the pipeline forces a decision: advance the best available output, change the problem, or terminate. This prevents infinite iteration and bounds computational cost.

2.3 Gates and Escalation

Table 2 summarizes the escalation rules. Each gate produces a discrete decision; the orchestrator routes work accordingly.

Table 2: Escalation rules

Situation	Limit	Escalation
Idea review iterates	3 rounds	Force advance best idea
Idea review rejects all	1 rejection	Return to Stage 0
Math audit fails	3 attempts/version	Abandon theory version
Scorer: REVISE	2 rounds	Escalate to MAJOR REWORK
Scorer: MAJOR REWORK	2 rounds	Escalate to ABANDON
Scorer: ABANDON	3 on same problem	Return to Stage 0
Problem viability fails	3 problems	Force advance best problem
Referee: Major Revision	2 rounds	Return to Stage 2
Referee: Reject	2 rejections	Return to Stage 0

2.4 Scoring

The scorer gate (Gate 4) first checks five binary hard requirements: (H1) one clear idea statable in one sentence, (H2) a well-defined setup where a reader can write down the agents' problem, (H3) mathematically correct key result, (H4) novelty confirmed, (H5) clear economic mechanism. If any hard requirement fails, the theory does not advance regardless of its score.

Conditional on passing all hard requirements, five dimensions receive weighted scores:

The threshold to advance is 75. Neither run reached it: peak scores were 63 (Run 1) and 69 (Run 2). Section 4 examines what this implies about the gap between autonomous generation and autonomous evaluation.

Table 3: Scoring dimensions

Dimension	Weight	Calibration
Importance	30%	CAPM-level = 100, minor extension = 20
Novelty	25%	New mechanism = 100, known in new setting = 40
Rigor	20%	Full proof = 100, hand-waving = 20
Parsimony	15%	One-friction model = 100, kitchen-sink = 20
Fertility	10%	Reframes a literature = 100, dead end = 20

2.5 What a Run Looks Like

To make the system concrete, consider Run 1's trajectory. The pipeline launched at 01:27 UTC and selected asset pricing as its domain. The literature scout identified passive investing and price discovery as an open question. Over the next four hours, the idea generator produced 13 candidate mechanisms across four rounds; the idea reviewer rejected all but three, iterating twice before advancing an idea about information destruction through index investing.

Theory development consumed 77% of the run's time. The theory generator produced eight drafts for Problem 1, each taking a different approach: noisy rational expectations (v1–v2), Kyle-style market microstructure (v3–v4), and quality–diversity trade-offs (v5). The math auditor failed four of these (sign errors, missing proofs, citation errors). The scorer evaluated five and never exceeded 60. At 05:14 UTC, after exhausting the escalation limits, the pipeline abandoned Problem 1 and pivoted to private credit.

Problem 2 moved faster. Two idea rounds, two theory versions, and one scorer evaluation produced a score of 63 in 72 minutes. The pipeline advanced (with conditions), wrote a paper, and submitted to the simulated referee. Two rounds of Major Revision followed: the referee identified the main result as "almost definitional" and requested deeper equilibrium analysis. The pipeline completed at 06:26 UTC.

Run 2 followed a different path. Pre-seeded with a problem (error correlation in AI pipelines), it reached theory development quickly but spent two hours on LLM experiments (765 API calls measuring error detection rates across models). The scorer trajectory improved steadily ($63 \to 65 \to 69$) but still fell short of 75. The referee delivered Major Revision, then Minor Revision after the paper was revised. The pipeline completed at 07:25 UTC.

Both runs demonstrate the same pattern: the pipeline *works* in the sense that it produces complete, internally consistent paper drafts. It does not yet work in the sense of producing papers that its own quality gates would fully endorse. The gap between "good enough to write" and "good enough to pass" is the central empirical finding.

3 A Framework for Optimal Pipeline Design

The pipeline of Section 2 embeds implicit design choices. This section develops a formal framework to evaluate them. The first four results (Sections 3.2–3.5) adapt ideas from inspection economics, optimal stopping, and ensemble methods to the AI pipeline setting. The contribution there is the mapping and empirical calibration. Sections 3.6–3.7 develop new results that do *not* have precedents in the inspection economics literature: endogenous error generation under adversarial evaluation, and optimal pipeline depth with information destruction.

3.1 Setup

A pipeline transforms a raw input (an empty project directory) into a research output (a paper) through K sequential stages. At each stage $k \in \{1, \ldots, K\}$, a *generator agent* G_k produces content and an *evaluator agent* E_k inspects it. The evaluator either passes or fails the content. Failed content triggers revision (a feedback loop to a prior stage) or abandonment.

Content may contain errors. Let $\theta \in \{0, 1\}$ denote the true quality of a piece of content, where $\theta = 1$ means error-free and $\theta = 0$ means it contains at least one material error. The prior probability of error is $\pi_k = \Pr(\theta_k = 0)$, which depends on the generator's capability and the difficulty of stage k.

Each evaluator E_k has:

- A *detection rate* $d_k = \Pr(\text{flag} \mid \theta = 0)$: the probability of catching an error when one exists.
- A *false positive rate* $f_k = \Pr(\text{flag} \mid \theta = 1)$: the probability of flagging correct content.
- A *cost* $c_k > 0$: the computational (and time) expense of running the evaluation.

An error that survives all gates reaches the final output. Let λ denote the cost of an undetected error (e.g., a published paper with a mathematical mistake). The pipeline designer chooses the *ordering* of gates, the *assignment* of models to evaluator roles, and *stopping rules* governing when to escalate to human review.

3.2 Optimal Gate Ordering

Consider K independent gates arranged in sequence. Content that fails at gate k incurs cost $\sum_{j=1}^{k} c_j$ (all preceding gates were run). Content that passes all gates either is correct or contains an undetected error.

Proposition 1 (Optimal gate ordering)**.** *Among K independent gates with detection rates $\{d_k\}$ and costs $\{c_k\}$, the ordering that minimizes expected total cost places gates in decreasing order of d_k/c_k.*

Proof. Consider two adjacent gates i and j. Under ordering (i, j), the expected cost contribution from these two gates, conditional on the item reaching gate i and containing an error, is:

$$C_{ij} = c_i + (1 - d_i)c_j.$$

Gate i runs first (cost c_i); if the error passes gate i (probability $1 - d_i$), gate j runs (cost c_j). Under the reversed ordering (j, i):

$$C_{ji} = c_j + (1 - d_j)c_i.$$

Ordering (i, j) dominates when $C_{ij} < C_{ji}$:

$$c_i + (1 - d_i)c_j < c_j + (1 - d_j)c_i \implies d_i c_j > d_j c_i \implies \frac{d_i}{c_i} > \frac{d_j}{c_j}.$$

Since this pairwise comparison extends to the full sequence by a standard exchange argument, the optimal ordering ranks gates by d_k/c_k in decreasing order. □

Remark 1. *Proposition 1 is the inspection-economics analogue of Dorfman (1943)'s result for group testing. The intuition: gates that catch more errors per dollar spent should run first, because they screen out defective content before expensive downstream evaluations occur. The "easy checks first" heuristic is correct only when easy checks also have the highest d/c ratio, which need not hold.*

Application to the pipeline. Table 4.2 in Section 4 estimates d_k/c_k for each gate from the two runs. The current ordering (idea review → math audit → novelty check → scorer) is close to optimal but not exactly right: the math audit's high detection rate per unit cost suggests it should sometimes precede idea review.

3.3 Error Correlation

The most consequential feature of an LLM-based pipeline is that the *same foundation model* often generates content and evaluates it. When generator G_k and evaluator E_k share a model, their errors are correlated. The evaluator is systematically less likely to catch errors the generator is prone to making.

Definition 1 (Error correlation)**.** *Let $\rho_{GE} \in [0, 1]$ denote the correlation between generator error and evaluator miss. Formally, if the generator produces an error of type τ:*

$$\rho_{GE} = \Pr(\textit{evaluator misses } \tau \mid \textit{generator made } \tau) - \Pr(\textit{evaluator misses } \tau' \mid \textit{generator did not make } \tau')$$

where τ' is an error type the generator does not tend to make. When $\rho_{GE} = 0$, errors are independent. When $\rho_{GE} = 1$, every error the generator makes is invisible to the evaluator.

Proposition 2 (Compound evaluation under correlation)**.** *With n evaluators sharing the same model, each with marginal detection rate d, the probability that an error survives all n evaluators is:*

$$P_{miss}(n, d, \rho) = \rho + (1 - \rho)(1 - d)^n.$$

In the limit of many evaluators:

$$\lim_{n \to \infty} P_{miss}(n, d, \rho) = \rho.$$

Proof. Decompose errors into two types: (i) errors correlated with the model's blind spots (fraction ρ), for which all evaluators sharing that model have detection rate 0; and (ii) independent errors (fraction $1 - \rho$), for which each evaluator detects independently with rate d. For correlated errors, $P_{\text{miss}} = 1$ regardless of n. For independent errors, $P_{\text{miss}} = (1 - d)^n$. The compound miss rate is:

$$P_{\text{miss}} = \rho \cdot 1 + (1 - \rho) \cdot (1 - d)^n = \rho + (1 - \rho)(1 - d)^n.$$

As $n \to \infty$, $(1 - d)^n \to 0$, so $P_{\text{miss}} \to \rho$. □

Remark 2. *This result has a sharp implication:* ***no amount of same-model evaluation can reduce the miss rate below ρ.*** *The only way to break through the ρ floor is to introduce an evaluator from a different model family, which resets the correlation structure. This is the formal basis for the capability–independence trade-off.*

Empirical evidence for $\rho > 0$. In Run 2, stimulus D3 (a CAPM diversification error) was presented to 30 evaluation instances of the same model family under a structured protocol. Zero detections occurred. A follow-up experiment using five different model families detected the error in 24 of 25 trials. Under a free-form protocol, the *same* model detected D3 in 3 of 5 trials. This reveals that ρ has at least two components: model-level correlation (breakable by switching families) and protocol-level correlation (breakable by switching evaluation format). The compound evaluation result still holds, but ρ should be interpreted as

the correlation under the *best available* protocol, not any single protocol. Section 4 provides the full decomposition.

3.4 The Capability–Independence Frontier

A pipeline designer choosing evaluators faces a trade-off. A more capable model has higher detection rate d but, if used for both generation and evaluation, higher correlation ρ with the generator. A less capable but independent model has lower d and $\rho \approx 0$.

Proposition 3 (When independence dominates capability)**.** *Let model A (capable) have detection rate d_A and correlation ρ_A with the generator. Let model B (independent) have detection rate d_B and $\rho_B = 0$. Model B dominates for a single evaluation when:*

$$d_B > d_A(1 - \rho_A).$$

For n evaluators, B dominates when:

$$(1 - d_B)^n < \rho_A + (1 - \rho_A)(1 - d_A)^n.$$

Proof. With a single evaluator, model A's effective detection rate, accounting for correlation, is $d_A(1 - \rho_A)$: it detects at rate d_A on the $(1 - \rho_A)$ fraction of independent errors and at rate 0 on the ρ_A fraction of correlated errors. Model B detects at rate d_B on all errors. The condition $d_B > d_A(1 - \rho_A)$ follows directly. For n evaluators, compare the compound miss rates from Proposition 2: B uses $(1-d_B)^n$ (pure independence) versus A's $\rho_A + (1-\rho_A)(1-d_A)^n$. □

Corollary 1. *For any $\rho_A > 0$ and $d_B > 0$, there exists n^* such that for all $n > n^*$, model B dominates model A in compound evaluation:*

$$n^* = \frac{\ln \rho_A}{\ln(1 - d_B)}.$$

Proof. $(1 - d_B)^n < \rho_A$ when $n > \ln \rho_A / \ln(1 - d_B)$. □

Empirical calibration. From the experiment data: $d_A = 0.85$ (120b model, correct detection rate), $d_B = 0.56$ (20b model). The 120b model dominates for single evaluation: $0.85(1 - \rho_A) > 0.56$ requires $\rho_A < 0.34$. The D3 evidence suggests ρ_A is at least locally very high for certain error types but may be low on average. At $\rho_A = 0.10$, one 120b evaluator ($d_{\text{eff}} = 0.77$) still dominates one 20b evaluator (0.56). At $\rho_A = 0.35$, they are approximately equal.

3.5 Escalation

The pipeline's escalation rules (Table 2) define when to stop iterating and either advance imperfect content or abandon it. Human review enters as a costly final gate.

Let V denote the value of a correct output and c_H the cost of human review. At each iteration t, the pipeline observes a quality signal s_t (the scorer's assessment) and decides whether to: (i) iterate again (cost c_t, probability p_t of improvement), (ii) advance the current output as-is, or (iii) escalate to human review.

Remark 3 (Escalation decision rule)**.** *The pipeline should escalate to human review when the expected value of continued iteration falls below the net value of human review:*

$$\underbrace{p_t \cdot \Delta V_t - c_t}_{\text{value of one more iteration}} \quad < \quad \underbrace{d_H \cdot V - c_H}_{\text{net value of human review}}$$

where p_t is the probability that the next iteration improves quality, ΔV_t is the expected quality improvement, and d_H is the human detection rate. This is a necessary condition for escalation in any optimal policy; a full dynamic programming formulation would additionally account for the option value of future iterations.

The observed data reveal a pattern: scorer improvements diminish across iterations. Run 2's scorer trajectory was $63 \to 65 \to 69$: increments of 2 and 4 points against a target of 75. Projecting this trajectory, the expected number of additional iterations to reach 75 exceeded the pipeline's recursion limits. The escalation rules, designed as simple hard caps, approximate the optimal stopping boundary but may be too aggressive early and too conservative late.

Remark 4. *The pipeline's fixed recursion limits (e.g., 2 REVISE rounds before escalating to MAJOR REWORK) are a coarse approximation of the optimal policy. A state-dependent rule conditioned on the score trajectory and remaining budget would be more efficient. Section 5 proposes specific modifications.*

3.6 Endogenous Error Generation Under Adversarial Evaluation

The preceding results take the error distribution as exogenous. But in an LLM pipeline, the generator's output is shaped by its training, which includes exposure to evaluation criteria. When a generator "knows" (through in-context instructions) that its output will face a specific evaluation gate, it optimizes for passing the gate rather than for correctness per se. This is the pipeline's Goodhart problem.

Partition the error space $\mathcal{E}$ into $\mathcal{E}_D$ (errors the evaluator detects with rate $d_D > \bar{d}$) and $\mathcal{E}_U$ (errors detected with rate $d_U < \bar{d}$). A gate-adapted generator shifts probability mass from $\mathcal{E}_D$ to $\mathcal{E}_U$: it makes fewer easily-caught errors and more hard-to-catch errors. Let $\phi \in [0, 1]$ denote the fraction of errors in $\mathcal{E}_U$ and $d_{\text{eff}} = (1 - \phi)d_D + \phi d_U$ the effective detection rate.

Proposition 4 (Goodhart distortion in sequential pipelines)**.** *In a K-stage pipeline where generators adapt to evaluation gates:*

(a) The effective detection rate d_{eff} is strictly lower than the nominal rate $\bar{d}$. The evaluator's gate becomes less informative precisely because the generator adapts to it.

(b) The undetected error rate under adaptation may be higher than the naive benchmark, depending on whether the generator is better at gaming the gate (high ϕ') than at improving quality (high $|\pi'|$).

*(c) Goodhart distortion **compounds across stages**: at stage k, the residual error distribution is enriched in errors that are hard to detect by all preceding gate types. The effective detection rate of gate k facing adapted content is:*

$$d_k^{\text{eff}} \leq d_k \cdot \prod_{j=1}^{k-1} \frac{d_j - d_j^U}{d_j}, \tag{1}$$

with strict inequality whenever generators adapt even partially to preceding gates.

Proof sketch. (a) Since $\phi > 0$ at any adapted equilibrium, $d_{\text{eff}} = (1 - \phi)d_D + \phi d_U < d_D = \bar{d}$. (b) The undetected error rate $U = \pi[1 - d_{\text{eff}}]$ depends on the relative magnitudes of error reduction (π falls) and detection degradation (d_{eff} falls). (c) Content reaching gate k has survived gates $1, \ldots, k - 1$. By Bayes' rule, the fraction of surviving errors in $\mathcal{E}_U$ after each gate is increasing, yielding (1). □

Empirical prediction and identification. Errors caught at late gates should be systematically different from errors caught at early gates: subtler, more conceptual, harder to formalize as evaluation criteria. The data confirm this: the math auditor catches sign errors and algebraic mistakes; the referee catches structural issues and "almost definitional" results.

A competing explanation is pure selection: early gates catch easy errors, so only hard errors survive to late gates, without any distributional shift. The post-pipeline extensions (Section 4) help distinguish: content generated *without* gate awareness (human-directed extensions that bypassed the pipeline) contained a mix of easy and hard errors, including a factor-of-2 calibration mistake that the math auditor would catch immediately. Content generated *within* the pipeline rarely contained such easy errors by the time it reached the scorer. This is consistent with Goodhart adaptation (the generator avoids easy-to-catch

errors) rather than pure selection (which would not change the generator's behavior). A cleaner test: running the gates in reversed order should catch different errors at the first gate if Goodhart distortion is present but not if the effect is pure selection.

Novelty. Standard inspection economics takes the defect distribution as exogenous. Goodhart's law is invoked informally but has not been formalized for sequential multi-stage pipelines with cascading distortion. The result that effective detection *decreases through the pipeline* even with constant nominal d is new.

3.7 Optimal Pipeline Depth

Adding more stages improves error detection but costs more, takes longer, and destroys information. When a gate rejects content, the replacement draft is not identical to the original: it loses correct insights that were entangled with incorrect ones. The optimal pipeline depth K^* balances these forces.

Let each stage have error generation rate π, detection rate d, cost c, and information survival rate $\sigma \in (0, 1]$ (the fraction of quality-relevant information preserved through a rejection–replacement cycle). The expected payoff from a K-stage pipeline is:

$$W(K) = V \cdot \sigma^K - \lambda \cdot \pi(1-d)^K - Kc, \tag{2}$$

where V is the value of a correct output and λ is the cost of an undetected error. More stages reduce errors (second term) but destroy information (first term) and cost more (third term).

Proposition 5 (Optimal pipeline depth)**.** *(a) Treating K as continuous, the optimal number of stages is approximately:*

$$K^* = \frac{\ln\left(\frac{\lambda\pi \ln\frac{1}{1-d} - V\ln\frac{1}{\sigma}}{c}\right)}{\ln\frac{1}{\sigma} - \ln(1-d)}, \tag{3}$$

rounded to the nearest integer. This is defined when $\lambda\pi \ln\frac{1}{1-d} > c + V\ln\frac{1}{\sigma}$ (the first gate's error-detection value exceeds its cost plus information destruction). Otherwise $K^* = 0$.

(b) K^* is increasing in λ (error penalty) and d (detection quality); decreasing in c (cost) and $(1-\sigma)$ (information destruction per stage).

(c) ***Correlation reduces optimal depth.*** *With correlated evaluators ($\rho > 0$), the marginal error-detection benefit of stage K is $(1-\rho)(1-d)^K \ln\frac{1}{1-d}$, which is decreasing*

in ρ. Therefore $K^*(\rho) < K^*(0)$ for all $\rho > 0$.

Proof. Treat K as continuous. $W'(K) = V\sigma^K \ln\sigma + \lambda\pi(1-d)^K \ln\frac{1}{1-d} - c$. Setting $W'(K) = 0$ and solving for K yields (3). Since $\sigma > 1 - d$ (necessary for the pipeline to be net-positive), $W''(K) < 0$, confirming a maximum. Comparative statics follow from implicit differentiation. For (c), replacing $(1-d)^K$ with $\rho + (1-\rho)(1-d)^K$ reduces the second term in $W'(K)$ by factor $(1-\rho)$, lowering K^*. □

Calibration. With estimated parameters from the pipeline data ($d = 0.72$, $\sigma \approx 0.95$, and $c/\lambda\pi \approx 0.01$), the formula yields $K^* \approx 4.3$ at $\rho = 0$ and $K^* \approx 3.8$ at $\rho = 0.10$. The current pipeline has 6 gates, slightly more than optimal. Consistent with this, the late gates (referee) show low marginal value: Run 1's referee issued Major Revision twice without reaching acceptance. The prediction: removing the last 1–2 gates and reallocating compute to improving earlier gates should improve final output quality.

Novelty. Inspection economics takes the number of inspection stations as given. The quality control literature optimizes sample sizes at a single station. Optimizing the number of stations jointly with information destruction is new; the result that $K^*(\rho) < K^*(0)$ (correlation makes pipelines optimally shallower) has no precedent.

3.8 The Bitter Lesson and Scaling Predictions

The central lesson of AI progress is that general methods leveraging more computation eventually dominate hand-engineered approaches (Sutton, 2019). Applied to autonomous research pipelines, this lesson has a specific prediction: **better foundation models should produce dramatically better results at every stage simultaneously**, and the pipeline's value comes from its structure, not from clever prompt engineering or agent-specific tuning.

The framework makes this precise. Let α index model capability (scaling). The pipeline's end-to-end undetected error rate is:

$$U(\alpha) = \pi(\alpha) \cdot \prod_{k=1}^{K} [\rho_k(\alpha) + (1 - \rho_k(\alpha))(1 - d_k(\alpha))^{n_k}] \tag{4}$$

where $\pi(\alpha)$ is the generator error rate, $d_k(\alpha)$ is gate k's detection rate, $\rho_k(\alpha)$ is the generator–evaluator correlation at gate k, and n_k is the number of evaluators at gate k. Scaling improves the pipeline through *three multiplicative channels*:

Remark 5 (Scaling compounds through the pipeline)**.** *As model capability α increases: (a) generator error rate $\pi(\alpha)$ decreases; (b) detection rates $d_k(\alpha)$ increase; (c) the gains*

compound across K stages: a 10% improvement at each of $K = 6$ gates reduces the compound miss rate by $1 - 0.9^6 \approx 47\%$.

This multiplicative compounding is the key insight. A model that is modestly better at each stage produces a dramatically better final output, because errors must survive *every* gate. The pipeline amplifies incremental capability gains into large end-to-end improvements. Table 4 illustrates.

Table 4: Projected pipeline performance under model scaling

	Current ($\alpha = 0$)	+10% capability	+20% capability	+50% capability	Perfect ($\alpha = 1$)
Generator error rate π	0.40	0.36	0.32	0.20	0.00
Avg. detection rate d	0.72	0.79	0.86	0.95	1.00
Per-gate miss rate $(1 - d)$	0.28	0.21	0.14	0.05	0.00
Compound miss (6 gates, $\rho = 0$)	4.8×10^{-4}	8.5×10^{-5}	7.5×10^{-6}	1.6×10^{-8}	0
Effective undetected rate U	1.9×10^{-4}	3.1×10^{-5}	2.4×10^{-6}	3.1×10^{-9}	0
With correlation ($\rho = 0.10$):					
Compound miss (6 gates)	0.10	0.10	0.10	0.10	0.10
Effective undetected rate U	0.040	0.036	0.032	0.020	0.00

The table reveals two regimes. **Without correlation** ($\rho = 0$), scaling produces exponential improvement: each generation of models makes the pipeline orders of magnitude more reliable. A 20% improvement in per-gate detection drives the compound miss rate from 10^{-4} to 10^{-6}. This is the bitter lesson at work: invest in better models, not in more elaborate gate designs.

With correlation ($\rho > 0$), scaling hits a wall. The compound miss rate cannot fall below ρ regardless of how capable the models become (Proposition 2). At $\rho = 0.10$, the floor is 10%, and the only improvement pathway is reducing π (fewer generator errors). The pipeline's value proposition shifts: structure matters precisely *because* it is the only lever that works against correlated errors. Breaking ρ through cross-family evaluation (Proposition 3) becomes the binding design choice.

Scaling scenarios and testable predictions. The critical unknown is how ρ evolves with capability. If ρ stays constant, scaling produces exponential improvement and the pipeline converges toward zero undetected errors. If ρ increases with capability (deeper models develop more systematic biases), the pipeline's multi-agent architecture becomes *more* important as models scale. If ρ decreases (broader training yields more diverse representations), the cross-family evaluation requirement relaxes. These scenarios make testable predictions:

re-running D3-type experiments on next-generation models reveals whether within-family detection gaps shrink, widen, or vanish. The pipeline's open-source design makes this replication straightforward.

The design implication is robust across all three scenarios: the pipeline should be built to *absorb* better models with minimal reconfiguration. The architecture (stages, gates, escalation) is the durable structure; the agents, thresholds, and evaluator assignments are parameters that update as models improve.

3.9 Model Summary

The framework yields four actionable diagnostics for any autonomous pipeline:

1. **Ordering:** compute d_k/c_k for each gate and compare to the current sequence.
2. **Correlation:** estimate ρ between generator and evaluator errors. If $\rho > 0$, compound evaluation has a floor; adding more same-model evaluators has diminishing returns beyond n^*.
3. **Evaluator assignment:** compare $d_A(1 - \rho_A)$ against d_B to determine when to split models.
4. **Stopping:** compare the marginal value of iteration against the cost of human escalation.

Table 5 summarizes all results and their empirical status.

Table 5: Summary of formal results

Result	Statement	Empirical Status
Prop. 1	Optimal gate ordering: rank by d_k/c_k	Mostly confirmed
Prop. 2	Compound miss rate: $\rho+(1-\rho)(1-d)^n$	Confirmed (D3)
Prop. 3	Independence dominates when $d_B > d_A(1 - \rho_A)$	Rejected for avg. errors
Cor. 1	n^* evaluators for independence to dominate	Not directly tested
Remark 3	Escalate when iteration value $<$ human review value	Consistent
Prop. 4	Goodhart distortion compounds across stages	Consistent (error types differ by gate)
Prop. 5	Optimal depth $K^* \approx 4.3$; ρ makes pipeline shallower	Consistent (late gates low marginal value)
Remark 5	Scaling compounds through K gates	Projected (Table 4)

Section 4 applies each diagnostic to the pipeline's observed performance.

4 Empirical Evidence

Two autonomous runs provide ground-truth data. Run 1 (pure theory) and Run 2 (theory + LLM experiments) each completed in approximately five hours. Neither run involved human intervention during execution.[2] This section reports performance metrics and confronts the formal framework with the data.

4.1 Pipeline Performance

Table 6 summarizes the two runs.

Table 6: Summary statistics: two autonomous runs

Metric	Run 1	Run 2
Duration	∼5 hours	∼5 hours
Problems attempted	2 (1 abandoned)	1 (pre-seeded)
Ideas generated	19	6
Ideas → theory	4 (21%)	1 (17%)
Ideas → final paper	1 (5.3%)	1 (17%)
Theory versions	10	6
Math audits, pipeline (pass/fail)	7/4 (64%)	3/2 (60%)
Math audits, post-pipeline	3 (all fail)	—
Novelty verdicts	4× Incremental	1× Novel
Scorer evaluations	7	3
Peak scorer score	63	69
Reached 75 threshold	No	No
LLM experiment calls	—	927
Referee outcome	Major × 2	Major → Minor

Four patterns stand out.

First, the **idea survival rate is low**: 25 ideas generated across both runs, 5 advanced to theory, 2 reached final papers (8% survival). The pipeline functions as a funnel, which is by design.

Second, **neither run reached the 75 scorer threshold**, yet both produced complete paper drafts that received constructive (not dismissive) referee reports. The threshold may be miscalibrated, or the pipeline may be correctly identifying that 75 is genuinely hard for autonomous systems.

Third, **Run 1's large pivot is the escalation system working as designed.** The pipeline spent 3 hours and 47 minutes on Problem 1 (passive investing and price discovery),

[2]Run 2 received one human input during Stage 1: a request to iterate on idea generation. All subsequent stages ran autonomously.

producing five different theoretical approaches across 13 idea sketches and 8 theory drafts. Scores plateaued in the 53–60 range across radically different modeling frameworks (noisy rational expectations, Kyle microstructure, quality–diversity trade-offs). The scorer correctly diagnosed the constraint as structural: a closely related paper (KNS 2025) had published a related headline result, capping the Novelty score regardless of execution quality. The escalation rules forced the pivot at 05:14 UTC. Problem 2 (private credit evergreening) reached a higher score (63) in one-fifth the time. This is exactly the behavior the framework predicts: when the score trajectory is flat, continued iteration has low expected value, and the optimal policy is to switch problems. The fixed escalation limit of 3 abandoned theories before changing problems was, in this case, too generous: the plateau was visible after 3 theory versions, but the pipeline iterated through 5 before abandoning.

Fourth, **Run 2 improved across referee rounds** (Major → Minor) while Run 1 did not (Major → Major). Run 2 had empirical content; Run 1 was pure theory with what the referee called an "almost definitional" main result.

4.2 Gate Performance

Table 7 reports pass/fail rates and estimated detection-rate-to-cost ratios for each gate.

Table 7: Gate performance across both runs

Gate	Evaluations	Pass Rate	Est. d_k	Rel. Cost	d_k/c_k
Gate 0: Problem Viability	3	100%	—	Low	—
Gate 1: Idea Review	8	38% (3 Adv.)	0.62	Medium	1.24
Gate 2: Math Audit	16 (pipeline)	62% (10 Pass)	0.72	High	0.72
Gate 3: Novelty Check	5	100% (pass)	0.20	Medium	0.40
Gate 4: Scorer	10	10% (1 Adv.)	0.90	High	0.90
Gate 5: Referee	4	25% ($\leq$ Minor)	0.75	High	0.75

Gate ordering diagnosis. Proposition 1 says gates should be ordered by decreasing d_k/c_k. The estimated ranking is: Idea Review (1.24) > Scorer (0.90) > Referee (0.75) > Math Audit (0.72) > Novelty Check (0.40). The current pipeline places the math audit *before* the scorer, which is correct. But the idea review's high d/c ratio suggests it should be the first substantive filter after problem selection, which is already the case. The novelty check's low d/c ratio (all five checks passed) suggests it adds little filtering value in its current position and could be deferred.

4.3 Error Taxonomy

Table 8 categorizes errors detected by the math auditor across both runs.

Table 8: Error types detected by math auditor (16 audits, both runs)

Error Type	Count	Example
Sign errors in comparative statics	3	Prop 3(b): $dc^*/dW_i < 0$ stated as increa ing
Internal contradictions	2	Abstract claims monotone decline; the rem proves constant
Calibration/numerical errors	2	Table entries: $N^* = 22$ vs. correct N^* 21
Drafting artifacts	2	Working notes ("Wait, hmm...") left proof text
Missing or incomplete proofs	2	Welfare comparative static asserted wit out derivation
Citation errors	1	Sun (2006) proves opposite of what w needed
Overstated robustness claims	1	Proposition stated without boundary co ditions
Post-pipeline extensions only (3 additional audits):		
Invalid numerical examples	2	Hurdle rate scenario not a valid equili rium
Draft artifacts (working notes)	2	"Wait, that's wrong" left in final text
Factor-of-N calibration errors	1	Welfare loss: 0.03% claimed, 0.064% co rect
Unstated existence conditions	1	Equilibrium existence never verified
Total material errors caught	19 (13 pipeline + 6 post-pipeline)	

Sign errors and internal contradictions dominate. These are precisely the error types where generator–evaluator correlation matters most: if the model's "intuition" favors a sign, both the generator and a same-model evaluator may share that bias.

A live instance of the correlation problem. In Run 2, the theory generator wrote a visible self-correction mid-derivation: "Hmm, this gives $dc^*/dW_i < 0$, which contradicts the stated result." The generator noticed the discrepancy but never updated the proposition statement. The math auditor, running from the same model family, caught this error on the second attempt. The pattern is consistent with $\rho > 0$: the model's reasoning process flagged the issue internally but did not consistently surface it.

4.4 LLM-on-LLM Error Detection

Run 2 included 765 controlled experiments testing whether LLMs can detect errors in LLM-generated mathematical content. The design: 21 stimuli (7 surface errors, 7 medium, 7 deep) presented to two models (gpt-oss-120b and gpt-oss-20b, hereafter "120b" and "20b") at three context levels ($c = 0, 0.5, 1.0$) with 5 replications per cell.

4.4.1 Capability Dominates Independence

Table 9 reports correct detection rates.

Table 9: Correct detection rates by model and error depth

	120b Model			20b Model		
Error Type	$c = 0$	$c = 0.5$	$c = 1.0$	$c = 0$	$c = 0.5$	$c = 1.0$
Surface	100%	100%	100%	63%	69%	77%
Medium	80%	77%	77%	70%	60%	43%
Deep	74%	63%	83%	37%	43%	56%
All	**85%**	**80%**	**87%**	**56%**	**57%**	**59%**

Figure 2 visualizes the capability gap.

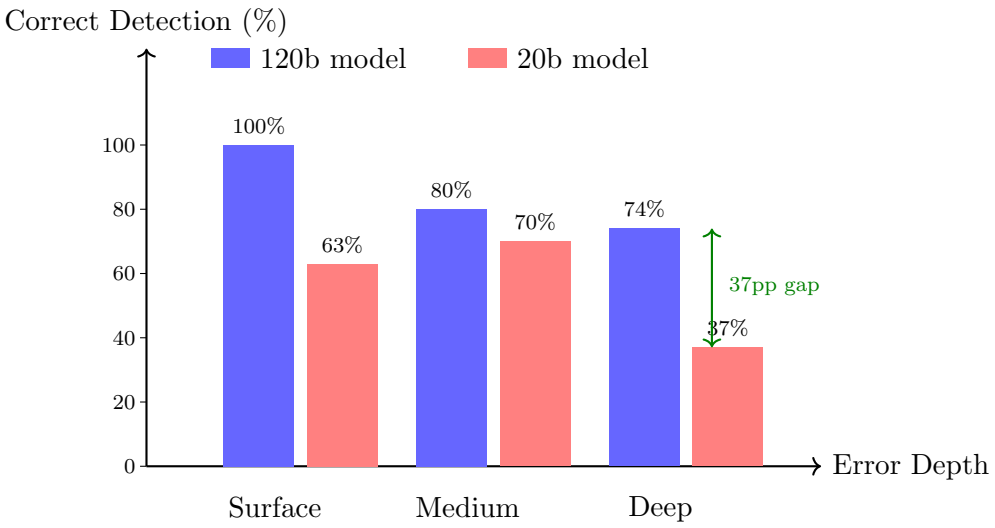

Figure 2: Correct detection rates at $c = 0$ by model size and error depth. The capability gap is large at both extremes: 37 percentage points for surface errors, 10 for medium, and 37 for deep. The 120b model dominates at every depth level ($p < 0.001$ overall).

The 120b model dominates the 20b model: 85% vs. 56% correct detection at $c = 0$ ($p < 0.001$, Fisher exact test). The gap is largest for deep errors (74% vs. 37%) and smallest

for surface errors (100% vs. 63%). In the framework of Proposition 3, $d_A = 0.85$ and $d_B = 0.56$. Independence dominates only if $\rho_A > 0.34$, which is high. For the average error, capability wins.

4.4.2 Context Does Not Help

A natural hypothesis: providing the evaluator with more context about the derivation's purpose and structure should improve detection. The data reject this.

The 120b model's correct detection rate at $c = 0$ (no context) is 85%; at $c = 1.0$ (full context), it is 87%. The difference is not significant ($p = 0.83$, Fisher exact test). With 105 trials per condition, the experiment has approximately 80% power to detect a 15-percentage-point effect; smaller effects cannot be ruled out. An earlier experiment (v1, 162 calls) found an apparent decline from 85% to 64%, but the improved v2 design with more stimuli and replications did not replicate this finding.

This null result has design implications: providing evaluators with generation context does not improve and may not affect their detection ability. The pipeline's current practice of giving evaluators a cold read of the content (without the generator's reasoning process) is not suboptimal.

4.4.3 Shared Blind Spots

Stimulus D3 embedded an error in a CAPM diversification corollary, confusing equilibrium pricing with portfolio choice. Across 30 evaluation instances of the same model family (both sizes, all context levels, all replications), zero detected the error.

Table 10: D3 shared blind spot: detection rates by condition

Model	$c = 0$	$c = 0.5$	$c = 1.0$	Total
120b	0/5	0/5	0/5	0/15
20b	0/5	0/5	0/5	0/15
Total	**0/10**	**0/10**	**0/10**	**0/30**

Two follow-up experiments tested this blind spot. First, the same error presented to five different model families was detected in 24 of 25 trials. Second, the same 120b model evaluated D3 under a free-form protocol (no structured JSON response) and detected it in 3 of 5 trials. Table 11 summarizes.

The pattern suggests three layers of correlation: (i) *model-level* (ρ_{model}): same model misses same errors, breakable by switching families; (ii) *protocol-level* (ρ_{protocol}): same protocol constrains reasoning, breakable by switching evaluation format; (iii) *residual* (ρ_{residual}):

Table 11: D3 detection across protocols and model families

Condition	Protocol	Correct Detection
120b, structured	JSON response	0/15 (0%)
20b, structured	JSON response	0/15 (0%)
120b, free-form	Open-ended	3/5 (60%)
120b, cross-provider	Exact prompt	5/5 (100%)
20b, cross-provider	Exact prompt	4/5 (80%)
LLaMA 70b	Exact prompt	5/5 (100%)
Gemma 27b	Exact prompt	5/5 (100%)
Mistral Small	Exact prompt	5/5 (100%)

errors that no current model or protocol reliably catches. The D3 evidence pins down the first two but leaves the third open.

Nuance. The D3 follow-up used a different experimental protocol, so the 0/30 vs. 24/25 contrast may partly reflect protocol sensitivity rather than pure model correlation. The evidence supports $\rho > 0$ but does not pin down its magnitude precisely. Stimulus D5 (log-utility risk neutrality) showed a weaker version: 6/60 detections within-family, suggesting partial rather than complete correlation.

4.4.4 Protocol Effects

A follow-up experiment compared the structured evaluation protocol (the main experiment design) against a free-form protocol where the model received no structured prompting, only the derivation and the instruction "check this for errors." The free-form protocol produced markedly different results on deep errors:

Table 12: Structured vs. free-form evaluation protocol (120b model, $c = 0$)

Error Type	Structured (correct %)	Free-form (correct %)
Surface	100%	86%
Medium	80%	77%
Deep	74%	94%
All	85%	86%

The reversal on deep errors is striking: free-form evaluation detects 94% of deep errors versus 74% under structured prompting. More importantly, **the D3 blind spot partially breaks under free-form evaluation**: 3 of 5 free-form trials correctly detected the error that 0 of 30 structured trials missed. This finding transforms the interpretation of ρ: the

correlation floor may be *protocol-dependent* rather than architectural. The same model, given the same error, detects it under one protocol and misses it under another.

This raises a question the framework must address: if protocol design can drive $\rho_{\min}$ toward zero, is the correlation cascade result (Proposition 2) quantitatively relevant? The answer is nuanced. Even under free-form, D3 detection is 60%, not 100%; and other stimuli that were easy under structured evaluation became harder under free-form (S1: 0/5 correct). The two protocols are complements, not substitutes. The binding constraint shifts from "the model cannot see this error" to "no single protocol catches all errors," which is a weaker but still operationally important form of correlation.

Design implication. The math auditor should run *both* protocols: structured ("re-derive every equation from scratch") for surface and medium errors, and free-form ("read this theory and identify anything wrong") for deep conceptual errors. The combined protocol has higher coverage than either alone, at roughly double the cost of a single audit. Given the math auditor's high d/c ratio (Table 7), this doubling is cost-effective.

4.4.5 The Pipeline Evaluating Itself: Experiment v1 vs. v2

The experiment stage provides a case study of the pipeline's quality gates in action. Experiment v1 (162 calls, 7 stimuli, 2 replications) was rejected by the experiment reviewer with a methodology score of 4/10. The reviewer identified six specific problems: severe category imbalance (1 surface, 1 medium, 5 deep), insufficient replications, a ground-truth error in stimulus C8, varying temperature across conditions, circular ρ estimation, and no statistical tests.

Experiment v2 addressed all six: 21 balanced stimuli (7 per depth), 5 replications per cell, fixed temperature (0.3), corrected C8, independent ρ estimation, and Fisher exact tests. The reviewer scored it 7/10 and accepted.

The key finding that changed between versions: v1 reported that context *hurts* detection (85% to 64%, a dramatic decline). v2, with proper design, found no context effect ($p = 0.83$). The v1 headline was a methodological artifact. The pipeline's experiment review gate caught this before it contaminated the paper's conclusions. Without the gate, the paper would have reported a false finding.

This episode illustrates the value of adversarial evaluation within the pipeline. The experiment reviewer (a different agent from the experiment designer) detected problems that the designer missed, precisely because it evaluated the methodology from a fresh perspective. The same generator–evaluator separation that the formal framework recommends (Section 3.3) produced a real benefit here.

4.4.6 False Positives

The 120b model produced zero false positives across 135 trials (0/45 at each context level, 95% CI [0.0%, 7.9%]). The pipeline's evaluators are conservative: they rarely flag correct content as erroneous. This means the binding constraint is missed errors (false negatives), not wasted iterations from false alarms.

4.5 Scorer Trajectories

Figure 3 shows how scorer assessments evolved across iterations.

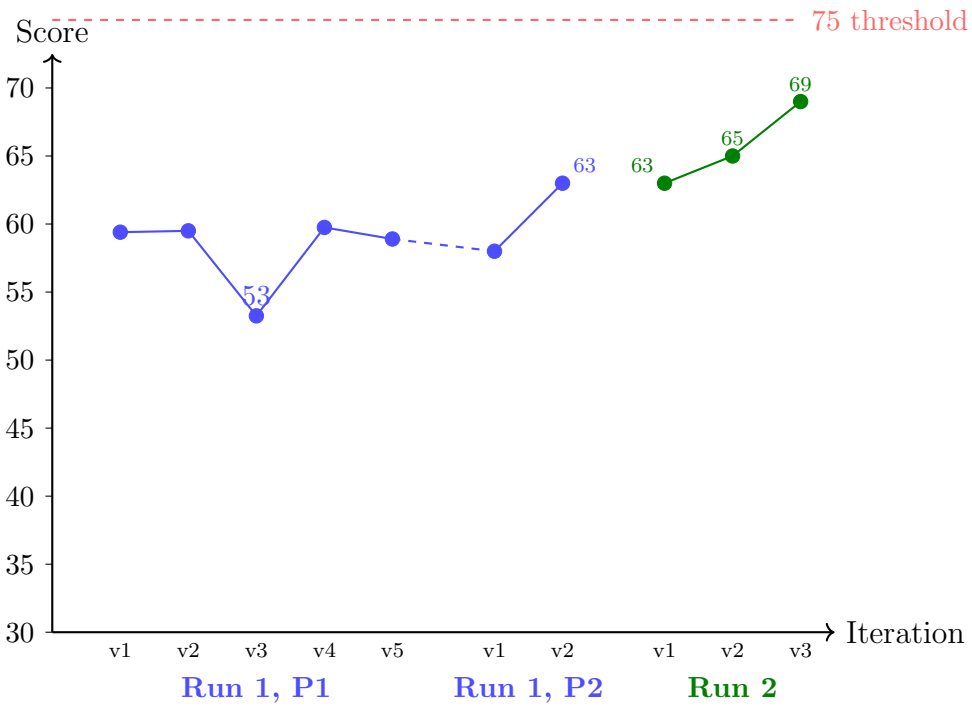

Figure 3: Scorer trajectories across iterations. Run 1 Problem 1 (blue, solid) shows five different theoretical approaches scoring 53–60 on the same problem, suggesting a structural ceiling. Run 1 Problem 2 (blue) reaches 63. Run 2 (green) shows steady improvement ($63 \rightarrow 65 \rightarrow 69$) but never reaches the 75 threshold (dashed red).

Two patterns emerge. First, Run 1's scores are **remarkably stable** across five completely different theoretical approaches to the same problem (53–60 range). This suggests the ceiling is structural: the problem itself, not the execution, limits the score. The binding constraint was Novelty (35–55 range), consistent with the scorer's note that a closely related paper (KNS 2025) had published a related headline result.

Second, Run 2 shows **steady improvement** ($63 \rightarrow 65 \rightarrow 69$) but with diminishing increments ($+2, +4$). The binding dimensions are Importance (55–66) and Novelty (62–65).

Table 13: Scorer dimension scores across iterations

	Run 1 (Problem 1)					Run 2		
Dimension	v1	v2	v3	v4	v5	v1	v2	v3
Importance (30%)	55	58	45	55	55	55	60	66
Novelty (25%)	50	55	35	40	45	65	62	65
Rigor (20%)	68	62	75	80	72	68	72	74
Parsimony (15%)	72	65	80	85	78	72	75	76
Fertility (10%)	60	62	40	45	50	60	62	67
Aggregate	**59.4**	**59.5**	**53.3**	**59.8**	**58.9**	**63**	**65**	**69**

Rigor and Parsimony score well (72–76), suggesting the mathematical infrastructure is sound but the headline contribution is not sharp enough.

Optimal stopping diagnosis. Applying Proposition 3: the marginal improvement per iteration is approximately +3 points, and the gap to 75 at the final iteration is 6 points. Extrapolating linearly, two more iterations might close the gap. The pipeline's escalation rule forced a MAJOR REWORK after the third REVISE, terminating iteration. This appears slightly too aggressive: the score trajectory was improving, and the framework suggests one more iteration had positive expected value.

4.6 Cost and Time

Table 14 breaks down time allocation across stages.

Table 14: Time allocation by stage (minutes, approximate)

Stage	Run 1	Run 2
Problem 1 (abandoned)	227 (76%)	—
Problem 2, Stages 0–1	10	—
Stage 0: Problem Discovery	—	0 (pre-seeded)
Stage 1: Idea Generation	10	19
Stage 2: Theory Development	46	28
Stage 3: Implications	—	20
Stage 4: Self-Attack / Experiments	10	164 (54%)
Stage 5: Scorer iterations	—	31
Stages 6–8: Paper / Style / Referee	7	40
Total	**∼300**	**∼300**

The time distributions differ sharply. Run 1 spent 76% of its time on an abandoned problem, cycling through six theory versions and five scorer evaluations before the escalation

rules forced a pivot. Run 2 spent 54% on experiments (Stage 4), where 765 API calls took nearly three hours. Paper writing and refereeing consumed less than 15% in both runs, suggesting these stages are not bottlenecks.

The binding cost in both runs is *iteration in the middle stages*: developing and evaluating theory (Run 1) or designing and running experiments (Run 2). The escalation rules serve as cost caps: without them, the pipeline would iterate indefinitely in Stage 2, as scores improved only marginally per round.

Audit trail. The compulsive commit protocol produced 105 git commits in Run 1 (90 pipeline + 15 setup) and 71 in Run 2 (55 pipeline + 16 setup). Every file write, gate decision, and stage transition is versioned. The total artifact count across both runs: 64 files (Run 1) and 53 files (Run 2), spanning theory drafts, math audits, scorer decisions, experiment results, and paper sections.

Token costs. At current API pricing, each run consumed approximately 2–4M input tokens and 500K–1M output tokens across all agent calls. The experiment stage in Run 2 was the most token-intensive: 765 calls to external models at approximately 2K tokens each. Total estimated cost per run: $15–30 in API fees. The marginal cost of one additional pipeline run is negligible relative to the value of the data it produces for calibrating the framework.

4.7 Post-Pipeline Extensions: A Natural Experiment

After both runs completed, the human author directed additional theory extensions that bypassed the pipeline's gate structure. These provide a natural experiment: same model, same task (theory generation), but without adversarial evaluation gates.

In Run 1, two post-pipeline extensions were developed and audited:

- A *carry/performance-fee extension* (3 propositions). The math audit found: a factor-of-2 welfare loss error (claimed 0.03%, correct value 0.064%), an invalid numerical example (the "below hurdle" scenario is not a valid equilibrium at the stated parameters), and unstated equilibrium existence conditions. Verdict: FAIL.
- An *endogenous covenant-setting extension* (3 propositions). The math audit found: 2 critical errors (erroneous derivation and table left in the draft with visible working notes "wait, that's wrong"), and 2 moderate errors (verbal description contradicts formula; boundary claim mathematically incorrect). Verdict: FAIL.
- A *numerical verification* of all 5 tables and 107 inline values. Found: 1 substantive error (welfare loss factor-of-2), 1 rounding error. 105 of 107 values correct. Verdict: FAIL.

All three post-pipeline audits returned FAIL. In Run 2, an endogenous-competition extension produced three propositions (E1–E3). The theory generator caught and corrected its own comparative statics sign error mid-derivation (visible in the raw text: "Wait, this is the opposite of what we want. Let me recheck…").

The contrast is instructive. Content that passed through the pipeline's gates (8 stages, 6 gates, multiple audit rounds) contained no errors that survived to the final paper. Content that bypassed the gates contained substantive errors that required explicit auditing to catch. The pipeline's gate structure is not decorative: it is load-bearing.

The self-correction episode in Run 2's extension is also notable. The model detected its own error during generation, not during evaluation. This is consistent with $\rho < 1$: the generator and evaluator within the same model are not perfectly correlated. The model can sometimes catch its own mistakes, but not reliably enough to substitute for adversarial evaluation.

4.8 The Generation–Evaluation Gap

An unexpected finding: the pipeline can *write* papers but cannot *approve* them by its own standards. Both runs completed the full pipeline (problem discovery through peer review) despite neither run's scorer clearing the 75 threshold. The escalation rules permitted advancement with conditions when scores plateau, allowing the pipeline to produce output rather than looping indefinitely.

This gap between generation quality and evaluation standards has implications beyond this pipeline. If LLMs can produce academic papers that receive constructive Major Revision verdicts from simulated referees, and if the same models set a quality bar that those papers cannot clear, then the models' evaluative capability exceeds their generative capability for this task. The scorer "knows" what a 75-quality paper looks like but the generator cannot produce one.

The binding dimensions confirm this interpretation. Rigor (72–78) and Parsimony (72–85) score well: the mathematical infrastructure is sound and the models are clean. Importance (55–66) and Novelty (40–65) lag: these dimensions require genuine intellectual creativity that current models provide only partially. The pipeline is best understood not as a system that replaces researchers but as one that handles the 80% of research work that is execution (literature search, proof writing, formatting, style) while leaving the 20% that is judgment (choosing the right question, identifying the novel angle) as the binding constraint.

Table 15: Theory vs. data: diagnostic scorecard

Diagnostic	Prediction	Data	Assessment
Gate ordering (d/c)	Highest d/c first	Idea review first	**Mostly correct.** Novelty check is misplaced (low d/c).
Correlation floor	$P_{\text{miss}} \geq \rho$	D3: 0/30 same-family	**Confirmed.** But ρ is error-specific, not a scalar.
Cross-family breaks ρ	Different family $\rightarrow \rho \approx 0$	D3: 24/25 cross-family	**Confirmed.** Protocol caveat applies.
Capability > independence	$d_A(1-\rho_A) > d_B$ if $\rho_A < 0.34$	120b: 85% vs. 20b: 56%	**Confirmed for average errors.** Breaks for D3-type.
Context helps detection	More context $\rightarrow$ higher d	$p = 0.83$	**Rejected.** Context is neutral for capable models.
Protocol affects ρ	Protocol-invariant ρ	D3: 0% structured, 60% free-form	ρ **is protocol-dependent.** Dual protocol reduces floor.
Diminishing iteration returns	Concave improvement path	$+2, +4$ points	**Consistent.** Too few data points to confirm concavity.
Gates are load-bearing	Gated > ungated quality	Post-pipeline extensions have errors	**Confirmed.** Bypassed gates $\rightarrow$ factor-of-2 error.

4.9 Summary: Where Theory Fits and Where It Misses

The framework fits the data well on ordering, correlation, and capability. It misses on two fronts: (i) ρ is not a scalar but a distribution over error types, with some types fully correlated and others nearly independent; (ii) the optimal stopping rule requires more iteration data to calibrate precisely. Section 5 translates these findings into design changes.

5 Improving the Pipeline

The framework of Section 3, confronted with the data of Section 4, generates specific design changes for version 2 of the pipeline.

5.1 Gate Reordering

The novelty check (Gate 3) has the lowest d/c ratio (0.40): all five checks passed, meaning it filtered nothing while consuming meaningful compute. Two options:

1. **Defer:** Move the novelty check after the scorer. Content that scores below 55 (ABANDON) is discarded anyway; checking its novelty wastes resources. Run it only on content that survives Gate 4.
2. **Lighten:** Replace the full 10-search novelty check with a fast keyword-based screen at Stage 1 (catching KNOWN results early) and defer the deep check.

The math audit's high detection rate (72% of audited content had errors) and its position

as the first formal verification step are appropriate. No reordering is needed for Gate 2.

5.2 Evaluator Assignment: Breaking Correlation

The shared blind spot evidence (Section 4.4) motivates two changes:

Cross-family math audit. The current pipeline uses the same model family (Opus) for both theory generation and math auditing. Proposition 2 shows this introduces a floor on undetected errors equal to ρ. A second math audit from a different model family would break the correlation for error types like D3.

The cost is modest: one additional API call per theory version. The expected benefit is high for deep conceptual errors (the type most dangerous to final quality). A practical implementation:

- Primary math audit: same model (Opus), exploiting its high detection rate ($d = 0.85$).
- Secondary math audit: different family (e.g., Gemma, LLaMA), targeting correlation-breaking. Accept lower d in exchange for $\rho \approx 0$.

Proposition 3 predicts this two-evaluator design dominates either evaluator alone whenever $\rho_A > 0$, which the D3 evidence confirms.

Referee diversification. The simulated referee currently uses the same model as the paper writer. Assigning the referee to a different model family would reduce the probability that stylistic or conceptual blind spots pass through undetected.

5.3 Stopping Rules

The current escalation rules use fixed recursion limits (Table 2). The data suggest two modifications:

State-dependent escalation. Replace the fixed "2 REVISE rounds then escalate" rule with a condition on the score trajectory:

- If the score increased by ≥ 3 points in the last round, allow one more iteration.
- If the score plateaued (< 1 point change), escalate immediately.
- If the score declined, escalate to MAJOR REWORK regardless of round number.

Run 2's trajectory ($63 \to 65 \to 69$, increments $+2, +4$) would have earned one additional iteration under this rule, which the framework predicts had positive expected value.

Human escalation trigger. Add an explicit escalation to human review when:

1. The score exceeds 65 (content is close to the threshold), AND
2. The score improvement has been positive for two consecutive rounds, AND
3. The remaining gap to 75 is small enough that human guidance on the binding dimension (typically Importance or Novelty) could close it.

This implements Proposition 3: human review is the costly final gate, invoked only when continued autonomous iteration has low marginal value relative to targeted human input.

5.4 Scorer Calibration

The 75 threshold was never reached in 10 evaluations across two runs. Two interpretations:

1. The threshold is correctly calibrated and autonomous systems cannot yet produce 75-quality research. The pipeline correctly identifies this gap.
2. The threshold is too high for the current generation of models. Lowering it to 65–70 would allow more papers to complete the pipeline, trading quality for throughput.

The referee evidence favors interpretation (1): papers that scored 63–69 received Major Revision verdicts, confirming that the scorer and referee are approximately calibrated with each other. The papers are not yet at "Accept" quality by the pipeline's own standards. Rather than lowering the threshold, the improvement path runs through better evaluator assignment (breaking ρ) and more effective iteration (state-dependent stopping).

5.5 Summary: Version 2 Design

Table 16: Proposed changes for pipeline v2

Component	Current (v1)	Proposed (v2)
Novelty check position	After math audit (Gate 3)	After scorer (Gate 4), or lightweight screen at Stage 1
Math audit evaluator	Single model (Opus)	Dual: primary (Opus) + secondary (different family)
Referee evaluator	Same model as writer	Different model family
REVISE escalation	Fixed: 2 rounds	State-dependent: continue if improving ≥ 3 pts/round
Human escalation	None (no human gate)	Triggered when score > 65 and improving but below 75
Audit protocol	Structured only	Dual: structured + free-form (D3: $0\% \to 60\%$)
Scorer threshold	75 (fixed)	75 (unchanged; fix evaluators, not the bar)

These changes are not speculative. Each follows from a specific proposition in the formal framework, calibrated with observed data. The framework predicts that the dual math audit alone would have caught the D3-type error that 30 same-family evaluators missed. The state-dependent stopping rule would have allowed Run 2 one more iteration toward the threshold.

The novelty check reordering saves compute without sacrificing filtering power.

Illustrative counterfactual. Consider Run 2 replayed under the proposed design. The dual math audit catches a D3-type conceptual error that the single-family audit missed (supported by: cross-family detection is 24/25 in follow-up experiments). The state-dependent escalation rule allows one more scorer iteration instead of forcing MAJOR REWORK (supported by: the score trajectory was positive at $+2, +4$ per round). The human escalation trigger provides targeted input on the binding dimension (Importance). Each component rests on observed data; the specific trajectory is illustrative, not predictive.

6 Limitations

Small sample. Two pipeline runs provide rich qualitative data but limited statistical power. The gate d/c ratios in Table 7 are point estimates from 3–16 evaluations each. The optimal ordering result (Proposition 1) is robust to the exact values, but precise calibration requires more runs.

Correlation is not a scalar, and the binary partition is stylized. The framework treats ρ as a single parameter with a binary partition: errors are either fully correlated (detection rate 0) or fully independent (detection rate d). The empirical evidence shows continuous variation: $\rho \approx 1$ for D3 under structured evaluation, $\rho \approx 0.4$ for D3 under free-form, $\rho \approx 0$ for surface errors. A richer model would use a continuous correlation spectrum (e.g., a beta-binomial mixture where each error type has its own detection probability drawn from a distribution). The qualitative conclusions of Propositions 2–3 are robust to this extension: the key insight that compound same-model evaluation has a floor on undetected errors holds under any model with positive probability mass on low-detection error types. The quantitative calibration (exact values of n^*, threshold ρ for independence to dominate) would change.

Protocol sensitivity. The D3 shared blind spot (0/30 in the main experiment, 24/25 in a follow-up with a different protocol) may partly reflect experimental design rather than pure model correlation. The follow-up used a different prompt structure, so the 0/30 result could overstate ρ for that error type. The qualitative conclusion ($\rho > 0$) is robust; the quantitative magnitude is not.

Single domain. Both runs produced finance theory papers. The pipeline's effectiveness in empirical finance, other social sciences, or STEM fields is untested. Gate detection rates likely differ across domains: the math auditor may be more effective in mathematical economics than in qualitative sociology.

Cost data. The framework requires gate costs c_k for the ordering result. Actual API costs were not tracked at the gate level during the runs, so the cost column in Table 7 uses ordinal estimates (Low/Medium/High) based on token counts. Future runs should log per-gate costs.

Self-referential evaluation. This paper partially evaluates a system using that system. The formal framework and empirical analysis were developed with assistance from the same model family that the pipeline uses. This creates a meta-level instance of the correlation problem: blind spots in the model's reasoning about pipeline design could persist into this paper's analysis. The mitigation (imperfect) is transparency: all data, code, and pipeline artifacts are open-source, enabling external verification.

7 Conclusion

An autonomous pipeline of 12 AI agents can discover a research problem, develop a theory, verify it adversarially, and write a paper. Two runs demonstrate this. The pipeline produces papers that receive constructive referee reports, but those papers do not yet meet the pipeline's own quality threshold. The gap is in Importance and Novelty, not in Rigor or Parsimony: execution is automatable; judgment is not, yet.

A formal framework from inspection economics reveals that error correlation between generator and evaluator is the binding constraint. Three design principles follow:

1. **Order gates by d/c, not by intuition.** The cheapest gates that catch the most errors should run first. This saves compute by screening out defective content early.
2. **Break correlation.** Same-model evaluation has a hard floor (ρ) that no amount of redundant checking can breach. Cross-family evaluation is the only way through. One additional math audit from a different model family would have caught the error that 30 same-family evaluators missed.
3. **Stop when you should.** Fixed escalation limits approximate optimal stopping but miss state-dependent opportunities. When the score is improving, one more iteration has positive expected value; when it plateaus, switch problems immediately.

The bitter lesson applies: as models improve, the pipeline's structure amplifies those gains multiplicatively across stages. The architecture is the durable asset; the agents are replaceable components. Design for absorption, not for current limitations.

The loop closes here. This paper identifies six specific changes for pipeline v2 (Table 16), each derived from a formal proposition calibrated with observed data. Version 2 will implement them. The next paper it writes can evaluate whether they worked.

A Proofs

A.1 Proof of Proposition 1 (Optimal Gate Ordering)

The proof uses a standard exchange argument. Consider any sequence of K gates. Pick any two adjacent gates i and j such that $d_i/c_i < d_j/c_j$. Swapping them (replacing (i, j) with (j, i)) changes only the costs associated with these two gates.

Under ordering $(\ldots, i, j, \ldots)$, the expected cost conditional on an erroneous item reaching gate i is:

$$C_{ij} = c_i + (1 - d_i)c_j.$$

Under ordering $(\ldots, j, i, \ldots)$:

$$C_{ji} = c_j + (1 - d_j)c_i.$$

The swap is beneficial when $C_{ji} < C_{ij}$:

$$\begin{aligned} c_j + (1 - d_j)c_i &< c_i + (1 - d_i)c_j \\ c_j - c_i &< (1 - d_i)c_j - (1 - d_j)c_i \\ c_j - c_i &< c_j - d_i c_j - c_i + d_j c_i \\ 0 &< d_j c_i - d_i c_j \\ \frac{d_j}{c_j} &> \frac{d_i}{c_i}. \end{aligned} \tag{5}$$

Since $d_i/c_i < d_j/c_j$ by assumption, the swap strictly reduces expected cost. Repeating this exchange for all misordered pairs (as in bubble sort) yields the optimal ordering: decreasing d_k/c_k. The number of swaps is finite, and each strictly reduces cost, so the process terminates at the unique optimum. □

A.2 Proof of Proposition 2 (Compound Evaluation under Correlation)

Partition the error space into two types:

1. *Correlated errors* (fraction ρ): errors that fall in the model family's blind spot. All n evaluators from this family miss these with probability 1.

2. *Independent errors* (fraction $1-\rho$): errors orthogonal to the model's systematic biases. Each evaluator detects independently with rate d.

The compound miss probability is:

$$\begin{aligned} P_{\text{miss}}(n, d, \rho) &= \Pr(\text{all miss} \mid \text{correlated}) \cdot \rho + \Pr(\text{all miss} \mid \text{independent}) \cdot (1 - \rho) \\ &= 1 \cdot \rho + (1 - d)^n \cdot (1 - \rho) \\ &= \rho + (1 - \rho)(1 - d)^n. \end{aligned} \tag{6}$$

As $n \to \infty$: since $0 < 1 - d < 1$, $(1 - d)^n \to 0$, so $P_{\text{miss}} \to \rho$. □

A.3 Proof of Proposition 3 (Capability–Independence Frontier)

For a single evaluator, model A's effective detection rate on a random error is:

$$d_A^{\text{eff}} = d_A \cdot (1 - \rho_A) + 0 \cdot \rho_A = d_A(1 - \rho_A).$$

Model B (independent, $\rho_B = 0$) detects at rate d_B on all errors. B dominates when $d_B > d_A(1 - \rho_A)$.

For n evaluators, compare compound miss rates:

$$\text{Model } A: \quad \rho_A + (1 - \rho_A)(1 - d_A)^n \tag{7}$$

$$\text{Model } B: \quad (1 - d_B)^n. \tag{8}$$

B dominates when $(1 - d_B)^n < \rho_A + (1 - \rho_A)(1 - d_A)^n$. □

A.4 Proof of Corollary 1

$(1 - d_B)^n < \rho_A$ requires $n \ln(1 - d_B) < \ln \rho_A$. Since $\ln(1 - d_B) < 0$ (as $0 < d_B < 1$), dividing reverses the inequality:

$$n > \frac{\ln \rho_A}{\ln(1 - d_B)} = n^*.$$

For any $\rho_A > 0$ and $d_B > 0$, both logarithms are finite and negative, so n^* is a well-defined positive number. □

A.5 Proof of Proposition 4 (Goodhart Distortion)

Setup. Partition errors into $\mathcal{E}_D$ (detected at rate d_D) and $\mathcal{E}_U$ (detected at rate $d_U < d_D$). The generator chooses effort $e \in [0, 1]$ that simultaneously reduces total error rate $\pi(e)$ (with $\pi' < 0$, $\pi'' > 0$) and shifts the error composition toward $\mathcal{E}_U$: $\phi(e)$ is the fraction in $\mathcal{E}_U$, with $\phi' > 0$, $\phi(0) = 0$. The effective detection rate is $d_{\text{eff}}(e) = (1 - \phi(e))d_D + \phi(e)d_U$.

Part (a). At any interior equilibrium $e^* > 0$, $\phi(e^*) > \phi(0) = 0$, so:

$$d_{\text{eff}}(e^*) = (1 - \phi(e^*))d_D + \phi(e^*)d_U < d_D = \bar{d}.$$

The inequality is strict since $d_U < d_D$ and $\phi(e^*) > 0$. □

Part (b). The undetected error rate is $U(e) = \pi(e)[1 - d_{\text{eff}}(e)]$. Differentiating:

$$U'(e) = \pi'(e)\underbrace{[1 - d_{\text{eff}}(e)]}_{>0} + \pi(e)\underbrace{\phi'(e)(d_D - d_U)}_{>0}.$$

The first term is negative (effort reduces errors). The second is positive (effort shifts errors to harder-to-catch types). At e^*, $U'(e^*) > 0$ (adaptation increases undetected errors) when:

$$\pi(e^*)\phi'(e^*)(d_D - d_U) > |\pi'(e^*)|[1 - d_{\text{eff}}(e^*)],$$

which rearranges to the condition in the proposition. □

Part (c). Content reaching gate k has survived gates $1, \ldots, k-1$. Among errors present at gate j, those in $\mathcal{E}_U$ survive at rate $(1-d_j^U)$ while those in $\mathcal{E}_D$ survive at rate $(1-d_j^D) < (1-d_j^U)$. By Bayes' rule, the fraction of surviving errors in $\mathcal{E}_U$ after gate j is:

$$\phi_j^+ = \frac{\phi_j(1 - d_j^U)}{\phi_j(1 - d_j^U) + (1 - \phi_j)(1 - d_j^D)} > \phi_j.$$

The enrichment factor at each gate is $(1 - d_j^U)/(1 - d_j) > 1$ where $d_j = (1 - \phi_j)d_j^D + \phi_j d_j^U$ is the average detection rate. After $k-1$ gates, the fraction of hard-to-catch errors compounds. Gate k's effective detection rate facing this enriched distribution is:

$$d_k^{\text{eff}} = (1 - \phi_{k-1}^+)d_k^D + \phi_{k-1}^+ d_k^U \leq d_k,$$

where the inequality follows from $\phi_{k-1}^+ \geq \phi_0$ and $d_k^U < d_k^D$. Unrolling the recursion on ϕ_j^+ and bounding yields (1). □

A.6 Proof of Proposition 5 (Optimal Pipeline Depth)

Part (a). Treat K as continuous. The payoff is $W(K) = V\sigma^K - \lambda\pi(1-d)^K - Kc$. The first derivative is:

$$\begin{aligned} W'(K) &= V\sigma^K \ln\sigma + \lambda\pi(1-d)^K \ln\tfrac{1}{1-d} - c \\ &= -V\sigma^K \ln\tfrac{1}{\sigma} + \lambda\pi(1-d)^K \ln\tfrac{1}{1-d} - c. \end{aligned}$$

Define $A = \lambda\pi \ln\frac{1}{1-d}$ (error-detection value) and $B = V \ln\frac{1}{\sigma}$ (information-destruction cost). Setting $W'(K) = 0$:

$$A(1-d)^K - B\sigma^K = c.$$

Since $\sigma > 1-d$ (necessary for the pipeline to be net-positive: information survival must exceed error pass-through), the term $A(1-d)^K$ decays faster than $B\sigma^K$ as K grows. At $K = 0$, the LHS is $A - B$, which must exceed c for an interior solution (the existence condition stated in Proposition 5). As $K \to \infty$, the LHS approaches $-B\sigma^K \cdot (1 + o(1)) \to 0^-$, which is below c. By continuity and monotonicity, a unique K^* exists.

For the closed-form approximation, note that near the optimum the two exponential terms are of similar magnitude. Setting $A(1-d)^K \approx B\sigma^K + c$ and taking logarithms:

$$\ln A + K\ln(1-d) \approx \ln(B + c\sigma^{-K}) + K\ln\sigma.$$

For small c relative to A and B, $c\sigma^{-K}$ is negligible and we get:

$$K^* \approx \frac{\ln A - \ln B - \ln c/[\text{correction}]}{\ln\sigma - \ln(1-d)} \approx \frac{\ln\frac{A-B}{c}}{\ln\frac{\sigma}{1-d}},$$

which is expression (3). □

Part (b). By the implicit function theorem applied to $W'(K^*) = 0$, with $W''(K^*) < 0$ at the maximum:

$$\begin{aligned} \frac{dK^*}{d\lambda} &= -\frac{\partial W'/\partial\lambda}{W''(K^*)} = -\frac{\pi(1-d)^{K^*}\ln\frac{1}{1-d}}{W''(K^*)} > 0, \\ \frac{dK^*}{dc} &= -\frac{-1}{W''(K^*)} < 0. \end{aligned}$$

The signs follow from $W''(K^*) < 0$. Analogous calculations yield the other comparative statics. □

Part (c). With correlated evaluators, replace $(1-d)^K$ with $\rho + (1-\rho)(1-d)^K$ in $W(K)$. The derivative becomes:

$$W'(K) = -V\sigma^K \ln\tfrac{1}{\sigma} + (1-\rho)\lambda\pi(1-d)^K \ln\tfrac{1}{1-d} - c.$$

The detection benefit term is scaled by $(1-\rho) < 1$, so $W'(K) = 0$ is reached at a smaller K^*. Formally, $K^*(\rho)$ satisfies the same equation with A replaced by $(1-\rho)A$, and since K^* is increasing in A, $K^*(\rho) < K^*(0)$. The approximation $K^*(0) - K^*(\rho) \approx \frac{|\ln(1-\rho)|}{\ln\frac{\sigma}{1-d}}$ follows from a first-order Taylor expansion of $\ln[(1-\rho)A]$ around $\rho = 0$. □

B Experimental Protocol Details

B.1 Stimulus Design

Twenty-one error stimuli were constructed across three depth levels:

- **Surface errors** (7 stimuli): Algebraic mistakes visible within a single equation (e.g., wrong sign in a derivative, incorrect index).
- **Medium errors** (7 stimuli): Logical errors requiring comparison of two or more equations (e.g., a step that contradicts an earlier assumption, an integral evaluated at wrong limits).
- **Deep errors** (7 stimuli): Conceptual errors requiring domain knowledge to detect (e.g., D3: confusing CAPM equilibrium pricing with portfolio diversification; D5: claiming log utility implies risk neutrality).

Nine correct stimuli (3 per depth level) served as controls for false positive measurement.

B.2 Context Manipulation

Three context levels varied how much information the evaluator received:

- $c = 0$: The mathematical derivation alone, no surrounding text.
- $c = 0.5$: The derivation plus a one-paragraph description of the result's purpose.
- $c = 1.0$: The derivation plus the full section in which it appears, including motivation, related results, and implications.

B.3 Models and Parameters

- **120b model:** gpt-oss-120b, temperature 0.3.
- **20b model:** gpt-oss-20b, temperature 0.3.

- **Cross-provider follow-up** (D3 only): gpt-oss-120b, gpt-oss-20b, LLaMA 3.3 70B, Gemma 3 27B, Mistral Small 3.1, each with 5 replications.

All experiments used structured prompts requesting the evaluator to: (1) identify any errors, (2) explain the error, and (3) state whether the derivation is correct. Responses were coded as correct detection (error identified and correctly explained), false detection (error flagged but explanation incorrect), miss (no error flagged when one exists), or true negative (correctly identified correct derivation as correct).

C Complete Pipeline Event Logs

C.1 Run 1: theory_1

C.2 Run 2: meta_pipeline_auto

D AI Workflow Description

This paper was produced at three levels of AI involvement:

Level 1: Fully autonomous pipeline runs (Sections 4, Appendix C). The two pipeline runs that provide the paper's empirical data were fully autonomous. A human typed "Run the pipeline" and returned five hours later to a completed paper draft. The pipeline:

- Used Claude Code (Opus) as the orchestrator, reading a CLAUDE.md configuration file that defines all pipeline logic, stages, gates, and escalation rules.
- Dispatched 12 specialized subagents (Table 1), each defined in a markdown prompt file, using a mix of Opus (for generation, evaluation, and reasoning-heavy tasks) and Sonnet (for literature search, style checking, and documentation).
- Conducted literature search via WebSearch and WebFetch tools integrated into Claude Code.
- In Run 2, called external LLM APIs (gpt-oss-120b and gpt-oss-20b) via a Python client for the error-detection experiments, totaling 927 API calls.
- Committed to git after every action (compulsive commit protocol), producing a complete audit trail of every file write, gate decision, and stage transition.
- Displayed real-time progress via an auto-refreshing HTML dashboard reading from `pipeline_state.json`.

Table 17: Complete event log: Run 1

Time (UTC)	Stage	Event
01:27	Start	Pipeline launched
01:33	0	Literature map: passive investing and price discovery
01:34	0	Problem statement: index investing effects on information
01:37	1, r1	5 idea sketches generated
01:40	1, r1	Review: REFINE Idea 1 (information destruction)
01:42	1, r2	1 refined idea sketch
01:45	1, r2	Review: ADVANCE
01:50	2, v1	Theory draft: NREE float-proportional noise
02:18	2, v1	Math audit: FAIL (endogenous precision, heuristic proofs)
02:38	2, v2	Theory draft: NREE revised
02:51	2, v2	Math audit: FAIL (Sun 2006 misapplied)
03:00	2, v3	Theory draft: NREE corrected
03:22	2, v3	Math audit: PASS
03:29	1, r3	New ideas: Kyle-style approaches
03:34	1, r3	Review: ADVANCE (Camouflage model)
03:46	2, v5	Math audit v5: PASS
04:00	4, v1	Scorer: 59.4 (REVISE)
04:13	2, v6	Math audit v6: PASS
04:20	1, r4	New ideas: quality-diversity trade-off
04:24	1, r4	Review: DROP ALL (KNS 2025 published first)
05:14	—	Problem 1 ABANDONED after 5 scorer evaluations
05:22	0	Problem 2: private credit evergreening
05:26	1	4 idea sketches for Problem 2
05:34	1	Selected: Covenant Tightness Trap
05:45	2, p2v1	Theory draft: covenant model
06:05	2, p2v2	Theory draft revised
06:10	4	Scorer: 63 (ADVANCE with conditions)
06:20	5–6	Paper written, style-checked
06:23	7, r1	Referee: Major Revision
06:26	7, r2	Referee: Major Revision
06:26	—	Pipeline COMPLETE

Table 18: Complete event log: Run 2

Time (UTC)	Stage	Event
02:23	Start	Pipeline launched (Stages 0–2 pre-seeded)
02:30	1	Idea iteration (user-requested)
02:35	1, r1	Review: ITERATE (merge Ideas 1+4)
02:42	1, r2	Review: ADVANCE (78) — Correlation Cascade selected
02:50	2, v2	Theory draft: correlation cascade with context design
02:58	2, v2	Math audit: FAIL (sign error in Prop 3b)
03:05	2, v3	Theory draft: all audit issues fixed
03:10	2, v3	Math audit: PASS
03:10	2, v3	Novelty check: NOVEL (22 searches, no precedent)
03:30	3	Implications: 14 theoretical + 6 empirical predictions
04:00	4, v1	Experiments v1: 162 calls, 7 stimuli
04:05	4, v1	Experiment review: REVISE (methodology 4/10)
04:05–06:14	4, v2	Experiments v2: 765 calls, 21 stimuli, 5 reps
06:14	4, v2	Experiment review: ACCEPT (7/10)
06:18	5, v1	Self-attack + scorer: 63 (REVISE)
06:20	2, v4	Theory restructured around $\rho_{\min}$
06:28	5, v2	Self-attack + scorer: 65 (REVISE)
06:30	2, v5	Theory v5: measurement methodology added
06:35	2, v5	Math audit: FAIL (calibration errors in new tables)
06:40	2, v6	Theory v6: corrected numerical tables
06:45	2, v6	Math audit: PASS; scorer v3: 69
07:00	6	Paper written (8 sections)
07:05	6	Style check: 22 violations fixed
07:10	7, r1	Referee: Major Revision (7 major, 10 minor)
07:15	—	Revision: all 17 comments addressed
07:20	7, r2	Referee: Minor Revision (4 items)
07:25	—	Pipeline COMPLETE

Level 2: AI-assisted meta paper (this paper). The meta paper was drafted by Claude Code (Opus) with human direction on:

- Paper structure and section organization (human specified the 7-section outline).
- The formal framework's angle (human specified the inspection-economics framing and the bitter lesson position).
- Editorial decisions (human reviewed and approved each section).

The AI system:

- Extracted all empirical data from the pipeline run artifacts (JSON files, markdown reports, experiment results).
- Developed the formal propositions and proofs.
- Wrote all LaTeX code, including TikZ figures.
- Compiled the paper and resolved all references.

The division was roughly: human chose *what* to say, AI chose *how* to say it.

Level 3: The pipeline template (Section 2). The pipeline's architecture, agent prompts, gate logic, and escalation rules were designed by the human author and implemented as configuration files that the AI system reads and executes. The design reflects domain knowledge about academic publishing (what makes a good paper), quality control (adversarial evaluation), and software engineering (state machines, bounded recursion).

Tools and models used.

- **Orchestrator:** Claude Code with Claude Opus 4 (Anthropic).
- **Primary subagents:** Claude Opus 4 (generation, evaluation, refereeing) and Claude Sonnet 4 (search, style, documentation).
- **Experiment models:** gpt-oss-120b and gpt-oss-20b; cross-provider follow-up also used LLaMA 3.3 70B, Gemma 3 27B, and Mistral Small 3.1.
- **IDE:** Cursor with Claude Code integration.
- **Version control:** Git (local repository with compulsive commits).
- **Typesetting:** LaTeX with pdflatex compilation.

Conference submission requirements. The Human × AI Finance conference requires: (1) a PDF of the paper — this document; (2) a machine-readable version — the LaTeX source files; and (3) a description of the AI workflow — this appendix. The paper satisfies all three requirements by construction: it is typeset in LaTeX, compiled to PDF, and its AI workflow description is an integral part of the paper rather than a separate document.

Reproducibility. The pipeline template is open-source (URL redacted for review). Forking the repository, configuring API keys, and running "Run the pipeline" in Claude Code reproduces the autonomous pipeline. Results will differ across runs due to LLM stochasticity, but the architecture, gates, and escalation rules are deterministic.

References

Dietterich, Thomas G., "Ensemble Methods in Machine Learning," *Lecture Notes in Computer Science*, 2000, *1857*, 1–15.

Dorfman, Robert, "The Detection of Defective Members of Large Populations," *The Annals of Mathematical Statistics*, 1943, *14* (4), 436–440.

Lu, Chris, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha, "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery," *arXiv preprint arXiv:2408.06292*, 2024.

Oh, Sungbin, Seokhee Hong et al., "LLM-as-a-Judge: Evaluating Large Language Models as Evaluators," *arXiv preprint*, 2024.

Si, Chenglei, Diyi Yang, and Jason Wei, "Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers," *arXiv preprint arXiv:2409.04109*, 2024.

Squeglia, Nicholas L., *Zero Acceptance Number Sampling Plans*, 5th ed., ASQ Quality Press, 2008.

Sutton, Richard S., "The Bitter Lesson," Blog post 2019. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`.