QUALITY-BLIND DETECTION OF SETTLING IN AGENTIC AI LOOPS

Anonymous authors
Paper under double-blind review

ABSTRACT

In an agentic AI loop, a generator proposes a change, an evaluator accepts or rejects it, the accepted change is applied, and the cycle repeats. The loop usually cannot see the true quality of what it produces, so practitioners often give it a fixed budget of rounds, and the loop continues to run even after it has stopped improving. In this paper, we study what a quality-blind detector that sees only the artifact and the evaluator's accept or reject decision can conclude about whether a loop has settled, and propose a stopping algorithm that detects the settling. We first show that no such detector can tell whether the quality is rising or falling, and the most it can claim is a bound on how much the quality could still change, in terms of how much the artifact keeps changing. To measure that change, we hash any artifact into a sketch whose update across a round is the round's edit. On the path of those sketches, the coherence measures the share of the magnitude of the recent updates that carries the artifact a net distance. Once a loop has settled, its successive edits no longer change the artifact in a common direction, and we derive the value that the coherence takes at that point, as a closed-form function of the algorithm's decay setting. Our algorithm, named LOOPSTOP, scores each change by how far its coherence falls below that value and accumulates the scores over the rounds that changed the artifact. A rejected round enters the sum as a change of magnitude zero, so a loop whose evaluator has stopped accepting is covered by the same accumulation. Our evaluation shows that LOOPSTOP delivers 8.3% higher quality than the next best adaptive baseline on average across multiple families of loops, at approximately 50% fewer rounds.

1 INTRODUCTION

Language model agents nowadays can improve their own work in a loop, where a generator offers a proposal and an evaluator judges it. Then, the accepted change by the evaluator is applied to the *artifact* before the cycle repeats (Madaan et al., 2023), where the artifact is the object that the loop improves, e.g., files, numerical parameters, etc. Designing such loops has become its own practice (Schluntz & Zhang, 2024), which is sometimes referred to as *loop engineering* (Osmani, 2026). A challenge is that the true quality of the artifact is usually not known to the loop, as a coding agent typically does not have access to hidden test suites or real-world deployment results at development time, and an agent tuning a controller rarely knows the real cost of the drive, for instance.

In current practice, the harness, which is the program that runs the agent, usually gives the loop a fixed budget of rounds that may be pre-specified by the user, and all the rounds run without knowing whether they actually improve the artifact. Recent works suggest that the quality that a generator-evaluator cascade can reach depends on the evaluator's judgment that decides what updates made by the generator are kept (Aksu, 2026; Han, 2026). When using self-correction, the loop stalls when a model cannot verify its own reasoning (Huang et al., 2024a; Kamoi et al., 2024; Stechly et al., 2025). It has also been shown that most of a repair loop's gain arrives early (Arimbur, 2026; Javaji et al., 2025), and coding agents that have reached the right code can continue and overwrite it (Kim et al., 2026). A research gap is that the round at which a loop has *settled*, meaning that its true quality has stopped rising, has not been measured. We therefore ask the following question in this paper:

> *Can a detector tell whether a loop has settled from the artifact and the evaluator's accept or reject decision alone, without knowing the true quality of the artifact?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The problem is hard because nothing inside the loop has information about the hidden true quality, so the artifacts come from a process that could behave identically under a completely different quality function. Change in the artifact is therefore not always evidence of progress. Moreover, a detection rule that fits a single kind of evaluator can fail under some other evaluator. For instance, a naive rule that counts consecutive rejections works only with an evaluator that rejects, whereas loops with an inspector that names a problem instead of rejecting the update would reject nothing. We make the following contributions in this paper.

Impossibility result for quality-blind detection (Theorem 1). A detector is *quality-blind* when it sees the artifacts and the evaluator's accept or reject decisions but not the true quality. We show that no such detector can tell the sign of the quality change, and the most it can claim is a bound on the magnitude of change. With an assumption that the quality is Lipschitz in a distance between artifacts, so that a small edit changes the quality by at most a proportional amount, we turn that claim into a bound on the distance between the artifacts. With each artifact mapped to a sketch vector, stopping becomes the question of whether the *artifact path*, which accumulates those updates, still has a common direction, i.e., whether its consecutive updates still share a direction.

Novel quality-blind stopping algorithm LOOPSTOP (Algorithm 1, Theorems 2 and 3). We propose an online algorithm that stops a loop based on the artifact path and the evaluator's decision alone. Our proposed sketch turns text, structured data, and numerical parameters in the artifact into vectors whose difference across a round is exactly that round's edit. On the path of sketches, the coherence of the recent updates has a closed-form value when the updates share no direction, and its excess above that value is used to derive the signal-to-noise ratio of the updates. We bound the probability that the accumulated scores stop a loop whose signal-to-noise ratio is above a threshold, and their delay once the updates have stopped sharing a direction.

Extensive evaluations across 5 benchmarks. We evaluate LOOPSTOP across a set of benchmarks with varying difficulties, showing that LOOPSTOP performs well for different types of tasks and provides approximately 8% quality improvement on average across all benchmarks while reducing the rounds used by approximately 50% compared to the next highest quality baseline.

Related work. Studies of self-improvement and coding-agent trajectories, which find that failed trajectories repeat the same actions and overwrite correct code (Bouzenia & Pradel, 2025; Majgaonkar et al., 2025; Kim et al., 2026; Zhao et al., 2026), score completed trajectories with the true quality after the fact. Deployed agent frameworks run a loop to its budget (Wang et al., 2025; Wu et al., 2024; Yang et al., 2024), or stop it on repeated actions and errors after a number of rounds that is set by hand (OpenHands, 2026). Stopping rules for a loop in progress mostly rely on a verifier that observes the quality at every round (Hammar et al., 2026; Sadhuka et al., 2025), the model's own verdict or confidence (Chen et al., 2026; Lin et al., 2026), a judge over the transcript (Pandya et al., 2026), or telemetry from earlier episodes (Dubey, 2026). The exception is semantic early stopping (Shrivastava, 2026), which thresholds the distance between consecutive drafts under a model embedding at a threshold tuned on held-out runs. None of these works derives its stopping threshold from the settled value of a path statistic instead of tuning it, and none states what the artifact path and the evaluator's decisions alone can establish. An extended related work discussion is in Appendix A.

2 WHAT A QUALITY-BLIND DETECTOR CAN CONCLUDE

We index the rounds in a loop by $t = 0, 1, \ldots, T-1$. Round 0 makes the first attempt, and every later round draws a proposal from the generator and either applies it or not. Let $A_t \in \{0, 1\}$ equal 1 when the change was applied, and ξ_t denote the artifact that round t starts from. Round t then leaves the artifact ξ_{t+1}, where a rejected round leaves $\xi_{t+1} = \xi_t$. A *loop* is a single pass of this cycle on a single task, and its *trajectory* is the sequence of artifacts and decisions that it leaves. The evaluator either accepts or rejects each proposal, or acts as an *inspector* and names a problem for the generator to fix. In the case of an inspector, every change is applied and $A_t = 1$ at every round t. Let $q \in [0, 1]$ denote the true quality of an artifact, and $\Delta_{t,W} = q(\xi_{t+W}) - q(\xi_t)$ denote the change in quality over the W rounds after round t, for $t + W \leq T$.

Assumption 1 (what the detector sees)**.** *The loop never consults q. A detector run at the end of round t is a function of $\xi_0, \ldots, \xi_{t+1}$ and $A_0, \ldots, A_t$ alone, and it receives no value of q at any round and no other input that depends on q either.*

Assumption 1 describes a deployed loop, whose harness has the artifact and the evaluator's decisions but not the true quality score that the loop is meant to raise. Appendix B lists the mathematical notations, and full proofs of all the theorems are in Appendix C.

Theorem 1 (no quality-blind detector can favor an increase in quality over a decrease)**.** *Let $\mathbb{P}$ and $\mathbb{P}'$ denote the distributions of the loop scored by q and of the same loop scored by $1-q$, and let $\Delta'_{t,W} = (1-q)(\xi_{t+W}) - (1-q)(\xi_t)$ denote the change in quality of the second loop. Then $\mathbb{P}'(C) = \mathbb{P}(C)$ for every event C that is a statement about $\xi_0, \ldots, \xi_T$ and $A_0, \ldots, A_{T-1}$ alone, and for every $\zeta > 0$ and every such C with $\mathbb{P}(C) > 0$,*

$$\mathbb{P}'(\Delta'_{t,W} > \zeta \mid C) \;=\; \mathbb{P}(\Delta_{t,W} < -\zeta \mid C).$$

This holds because replacing q by $1-q$ leaves the distribution of the observations unchanged and negates $\Delta_{t,W}$. Intuitively, for any given quality definition, one can define the quality in the opposite way so that the same path of artifacts and decisions is consistent with a quality that increases and with one that decreases. Therefore, a quality-blind detector can at most conclude that the quality has stopped changing. This is why LOOPSTOP detects settling, which is when the artifact stops changing. The direction of that change is set by the evaluator, whose accepted improvements raise the quality, although the detector cannot infer it.

Assumption 2 (the quality is Lipschitz in a distance between artifacts)**.** *There is a distance d between artifacts and a constant $L < \infty$ with $|q(\xi) - q(\xi')| \le L\, d(\xi, \xi')$ for all artifacts ξ, ξ'.*

On a single loop producing finitely many artifacts, Assumption 2 holds with L being the largest of the finitely many ratios $|q(\xi_t) - q(\xi_{t'})|/d(\xi_t, \xi_{t'})$ for $d(\xi_t, \xi_{t'}) > 0$. Let d be the magnitude of the difference of the sketch vectors (defined in Section 3.1), and the updates be independent with a common distribution of finite variance. Under Assumption 2, the expected magnitude of the change in quality $\Delta_{t,W}$ is then at most L times the root mean squared magnitude of the sum of the W updates after round t (see Appendix D for details).

3 LOOPSTOP

3.1 SKETCH OF THE ARTIFACT

Requirements for the sketch. Let $S(\xi) \in \mathbb{R}^m$ denote the *sketch* of an artifact ξ, defined below, which is a vector with a dimension m fixed before the loop starts and its m *coordinates* indexed by $r = 1, \ldots, m$. Let $u_t \in \mathbb{R}^m$ denote the *update* of round t, defined in (1) below, which is the difference $S(\xi_{t+1}) - S(\xi_t)$ plus a term for any text carried elsewhere within the artifact, and is zero when the round is rejected. The *path* is the sequence $x_0, x_1, \ldots$ of the accumulated updates, $x_t = \sum_{s<t} u_s$, i.e., the trajectory of Section 2 mapped into $\mathbb{R}^m$ without the decisions. The *magnitude* of a vector is its Euclidean norm, e.g., $\|u_t\|$ for an update. The *distance* between two artifacts is the magnitude $\|S(\xi) - S(\xi')\|$ of the difference of their sketches, which is the d of Assumption 2, and it is at most the magnitude of the sum of the updates between them, with equality when no text was carried elsewhere. The sketch cuts an artifact into *parts*, each of which is a *name* paired with a *quantity*. The name identifies a piece of the artifact, e.g., a token of a text, the key of a value in structured data, or the index of an entry of a parameter vector, and the quantity is how much of that piece is held by the artifact. The sketch needs to meet the following requirements so that it is usable for quality-blind detection.

- R1: *The sketch vector has a fixed dimension m independent of which names the loop will produce.* A vector that grows in dimension as later rounds bring tokens and parameter names would put early and late updates in spaces of different dimensions, which cannot be compared.
- R2: *A part that the round did not touch contributes nothing to the update.* Otherwise the same edit made to two different artifacts would give two different updates, and computing an update would cost a pass over the entire artifact.
- R3: *The update tracks how much the artifact changed.* The common direction of the path is measured against the scatter of its updates, so a single-line fix and a rewrite of the whole artifact should give updates of different magnitudes. A change of the same relative size has to count the same, e.g., doubling 0.5 and doubling 50 should give updates of equal magnitude.

Following these requirements, we describe the design of LOOPSTOP's sketch next.

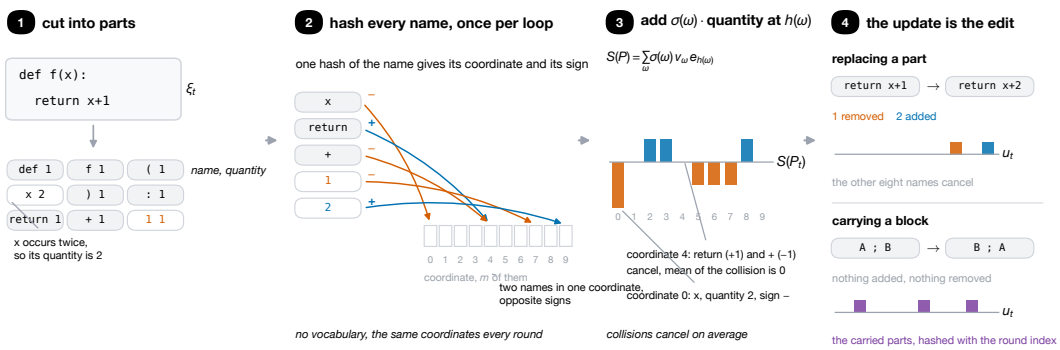

Figure 1: The four stages of a sketch. Parts are cut from the artifact, each name is hashed to a coordinate of the sketch and a sign, the signed quantities are added into those coordinates (the bars), and the update keeps only the edit.

The parts. The type of a component in the artifact decides how it is partitioned into parts. Text is partitioned into sequences of n consecutive tokens, with a token being a word or a symbol, each distinct sequence carrying as its quantity how many times it occurs. A scalar or a vector of scalars is partitioned into its entries, each carrying $\operatorname{sign}(v)\log(1+|v|/s)$ for the value v of the entry, where s is the magnitude of that entry at the start of the loop, or $s = 1$ if the magnitude is zero. A relative change of a given amount is then the same distance for every entry, since going from s to $2s$ adds $\log 3 - \log 2$ regardless of what s is, which is what R3 requires. Structured data is partitioned into the parts of its innermost values, with each name prefixed by the keys that lead to the value, e.g., the depth value 3 in $\{\texttt{"model"}:\{\texttt{"depth"}:3\}\}$ is a part named `model/depth`. Names come from content instead of position, to meet R2 since a line index changes for every line below an insertion.

The sketch. Figure 1 shows the construction. A hash function H is a pseudo-random map from strings to integers whose seed is drawn once at the start of the loop. Its bits above the lowest give the coordinate $h(\omega) = \lfloor H(\omega)/2 \rfloor \bmod m$ of the sketch into which a name ω is hashed, and its lowest bit gives the sign $\sigma(\omega)$, as in signed feature hashing (Weinberger et al., 2009). The sign makes two names that share a coordinate cancel on average instead of accumulating, so the sketched inner products are unbiased and their variance falls as $1/m$ (see Appendix E). Hashing needs no vocabulary and fixes m in advance, which is what R1 requires. Let P_t denote the set of parts of the artifact ξ_t, in which each name occurs once. The *sketch* of a set of parts P is $S(P) = \sum_{(\omega,v)\in P} \sigma(\omega)\, v\, e_{h(\omega)}$, with e_r the unit vector of coordinate r, and $S(\xi_t) = S(P_t)$. The sketch is linear in the quantities, so a name whose quantity is the same in both artifacts cancels in the difference $S(P_{t+1}) - S(P_t)$. The update of round t is defined as

$$u_t = S(\text{added}) - S(\text{removed}) + S_t(\text{carried elsewhere}), \tag{1}$$

where the first two arguments are the names whose quantity round t raised or lowered, each paired with the size of the change, the third is the parts that the round carried elsewhere, and S_t is the sketch taken under names with the round index t appended. The first two terms are the difference of the two sketches, at the cost of a single pass over the edit, which is what R2 requires. For text the update is divided by the square root of the mean number of parts of the two artifacts, which is the magnitude that the sketch of either artifact would have if every quantity were 1 and no names collided. An update's magnitude is then relative to the size of the artifact, which is what R3 requires.

Recording a block carried elsewhere in the third term of (1)**.** A round that only carries a block of text to another place leaves the quantity of every part unchanged, so the first two terms are zero. A version control diff, e.g., the output of `git diff`, aligns the two artifacts, and a name in both its deleted and its inserted list is carried, with the smaller of its two counts as its quantity. The third term hashes the carried parts under names with the round index appended, so that the carried blocks of different rounds fall in unrelated coordinates. A carried block then adds to the magnitude of its round's update and shares no direction with any other round.

3.2 The Coherence of the Artifact Path and Its Settled Value

The updates of Section 3.1 turn the question of loop settling detection into whether the recent updates still *share a direction*, i.e., have positive inner products, so that each update keeps part of the change

of the updates before it. An improving loop builds on its earlier edits, so the magnitude of the sum of its updates grows with the number of updates. The edits of a settled loop, in contrast, undo each other or scatter over unrelated parts, so the sum of its updates has a small magnitude although the updates themselves can still be as large. Let $\lambda \in (0, 1)$ denote a decay factor and

$$M_t = \frac{\sum_{j=0}^{t} \lambda^j u_{t-j}}{\sum_{j=0}^{t} \lambda^j}, \qquad V_t = \frac{\sum_{j=0}^{t} \lambda^j \|u_{t-j}\|^2}{\sum_{j=0}^{t} \lambda^j}, \qquad c_t = \frac{\|M_t\|^2}{V_t},$$

so that M_t and V_t are decay-weighted averages of the updates and their squared magnitudes, respectively, emphasizing the last few updates. The *coherence* c_t is the share of the magnitude of the recent updates that carries the artifact a net distance. We have $c_t \in [0, 1]$ by Jensen's inequality for the square. If every recent update points in the same direction with the same magnitude, $c_t = 1$, and if the updates point in unrelated directions they cancel and c_t is small.

Theorem 2 (the settled coherence)**.** *Let the updates $u_0, \ldots, u_t$ be independent with a common distribution, with $\mathbb{E}\|u\|^2 < \infty$ and $\mathbb{E}\|u - \mathbb{E}u\|^2 > 0$, and let $r = \|\mathbb{E}u\|^2 / \mathbb{E}\|u - \mathbb{E}u\|^2$ denote the squared signal-to-noise ratio of a single update. Then*

$$\frac{\mathbb{E}\|M_t\|^2}{\mathbb{E}[V_t]} = \frac{r + w_t}{r + 1}, \qquad w_t = \frac{1-\lambda}{1+\lambda} \cdot \frac{1+\lambda^{t+1}}{1-\lambda^{t+1}},$$

and w_t decreases in t to $(1 - \lambda)/(1 + \lambda)$.

The novelty in the proof of Theorem 2 is in what the expansion of the second moment establishes. First, the expected coherence depends on the update distribution only through r. Second, the settled coherence is the ratio z'_t/z_t^2 of the geometric sums $z_t = \sum_j \lambda^j$ and $z'_t = \sum_j \lambda^{2j}$ of the decay weights and of their squares, and the factorization $1 - \lambda^{2(t+1)} = (1 - \lambda^{t+1})(1 + \lambda^{t+1})$ gives it in closed form at every finite t.

The ratio r measures how strongly the updates share a direction, from $r = 0$ when they share none to $r = 1$ when the shared part is as large as the random part. At $r = 0$ the ratio of the theorem equals w_t, which is therefore the coherence of a path with no common direction on average. This value is the *settled coherence*, i.e., the mean coherence of a settled loop, against which the algorithm measures every loop. The decay weights average about $\tau = (1 + \lambda)/(1 - \lambda)$ updates, which is the *memory* of the statistic. The average of τ independent updates with no common direction has $1/\tau$ of the expected squared magnitude of a single update. The coherence of such a path is therefore about $1/\tau$, which is the limit of w_t. The theorem is stated for a ratio of two means, and the algorithm below uses that ratio as a proxy for the mean of c_t, which we refer to as the *mean-ratio approximation*.

Let $y_t = c_t/w_t$ denote the *normalized coherence*, which is the coherence in units of a path with no common direction. Under the mean-ratio approximation its mean is $(r + w_t)/(w_t(r + 1))$, which equals 1 at $r = 0$, increases with r, and depends on nothing else once λ and t are fixed. The statistic y_t is therefore comparable across loops with no calibration. A value $y_t = 1$ means that the loop appears settled, $y_t > 1$ that the recent updates agree more than chance, and $y_t < 1$ that they cancel more than chance (consecutive changes reverse each other).

3.3 The Round at Which the Loop Stops

In general, LoopStop decides to stop the loop when the updates of the path no longer share a direction. A loop with an evaluator may leave the artifact unchanged on most of its rounds, so the evidence has to separate a round that changed nothing from a change with no direction. Each change is scored as $\kappa_k - y_k$, i.e., by how far the normalized coherence lies below a *reference value* κ_k, chosen so that the score has a non-negative mean once the loop has settled and a negative mean while it improves, and the scores accumulate into the evidence that stops the loop.

The statistic is formed over the changes. A round *changes* the artifact when its update is non-zero, and $k = 1, 2, \ldots$ counts those changes. Over every round, a decay window on a loop that accepts few proposals would usually hold a single change. A single update u followed by j unchanged rounds has $\|M\|^2 = \lambda^{2j}\|u\|^2/z^2$ and $V = \lambda^j\|u\|^2/z$, where z is the sum of the weights, so its coherence is λ^j/z. This value decreases by at least the factor λ per unchanged round. Forming M, V and the weights over the changes gives back the equality of Theorem 2 with the number of changes in place of the round index, when the updates of the changes are independent with a common distribution. The k-th change then has the settled coherence w_{k-1}, and with $y_k = c_k/w_{k-1}$ the mean of y_k under the mean-ratio approximation is 1 under no common direction regardless of the number of unchanged rounds.

Assumption 3 (the updates of a settled path share no direction)**.** *After settling,* $\mathbb{E}[y_k] \leq 1$ *at every later change.*

Assumption 3 holds in three cases, including changes that are independent with a common distribution, which give y_k the mean 1 by Theorem 2 under the mean-ratio approximation, consecutive changes that reverse each other, whose settled coherence is below w_{k-1}, and a loop that has stopped changing the artifact. The assumption can fail on a loop whose evaluator agrees with the true quality and accepts only the few changes that keep pointing in the same direction. Such a loop is stopped by its unchanged rounds instead, as described below.

The reference value depends on how much direction the loop has shown. Before settling, a common direction raises the mean of y_k above 1, to $(r + w_{k-1})/(w_{k-1}(r + 1))$ under the mean-ratio approximation. Let $\kappa = (1 + w)/(2w)$ with $w = (1 - \lambda)/(1 + \lambda)$ denote the value that mean takes at $r = 1$ in the limit of having many changes. Let κ' be a reference value held fixed, with $1 \leq \kappa' \leq \kappa$. A change scored as $\kappa' - y_k$ contributes a mean of at least $\kappa' - 1 \geq 0$ once the loop has settled, by Assumption 3. It contributes a negative mean while the mean of y_k lies above κ', i.e., while the signal-to-noise ratio r of the changes exceeds a threshold that grows with κ' and is zero at $\kappa' = 1$ (Appendix F gives the threshold in closed form). Let

$$\bar{y}_t = \begin{cases} y_1 & \text{round } t \text{ made the first change,} \\ \lambda\, \bar{y}_{t-1} + (1 - \lambda)\, y_k & \text{round } t \text{ made the } k\text{-th change, } k \geq 2, \\ \lambda\, \bar{y}_{t-1} + (1 - \lambda) & u_t = 0 \text{ and } k \geq \tau, \\ \bar{y}_{t-1} & u_t = 0 \text{ and } k < \tau, \end{cases} \tag{2}$$

$$y^*_t = \max\big\{1,\ \max\{\bar{y}_j : j \leq t,\ k_j \geq \tau\}\big\},$$

where k is the number of changes up to round t and k_j that up to round j, denote the smoothed statistic and the *demonstrated peak*, which is the largest smoothed value at any round with at least τ changes, floored at 1. The smoothing averages y_k over the recent changes with the same decay as M_t and V_t, so that the peak reflects a sustained direction instead of a single noisy value, and once τ changes have occurred an unchanged round enters with the value 1 of an update with no direction, so that $\bar{y}_t$ decays toward 1, which the score of an unchanged round is measured against. The reference value of the k-th change, made at round t, is $\kappa_k = \min\{\kappa, 1 + \rho\,(y^*_t - 1)\}$ with a slope $\rho > 0$, so $\kappa_k = 1$ for a loop whose changes have shown no direction, increases with the peak, and is capped at κ once $y^*_t \geq 1 + (\kappa - 1)/\rho$. This definition of κ_k is a design choice, as it is the simplest rule that raises the reference value in proportion to the direction that the loop has already demonstrated. We set $\rho = \tau/2$ so that the cap applies once the peak reaches $2 - 1/\tau$. At $\kappa_k = 1$ the score $1 - y_k$ has a non-negative mean once the loop has settled, by Assumption 3, and a negative mean under any common direction, since any $r > 0$ puts the mean of y_k above 1.

Unchanged rounds are scored so that a loop whose evaluator stops accepting can be stopped. An unchanged round leaves k and y_k as they are, since both are formed over the changes, and lowers $\bar{y}$ toward 1 by (2). It scores $\kappa - \min\{\bar{y}, \kappa\}$ against the resulting smoothed statistic $\bar{y}$. The reference value is the fixed κ, so that a loop that has shown no direction is still stopped once its evaluator stops accepting. The round adds nothing while the recent changes still share a direction, so that $\bar{y} \geq \kappa$, and adds $\kappa - 1$ once that direction has faded, which takes a few rounds at the decay λ. A loop that has stopped changing is therefore stopped soon after the fade.

LOOPSTOP. Page's cumulative sum rule (Page, 1954) adds the scores of the rounds into a cumulative sum Λ_t, resets Λ_t to zero when it becomes negative, and stops at a threshold Λ^*. The *warm-up* is the round of the τ-th change, before which the averages hold too few updates to tell whether they share a direction. We make a novel extension to the standard rule to support the different causes of stopping. In essence, LOOPSTOP runs the following rule over the loop's rounds:

$$\Lambda_t = \begin{cases} 0 & \text{round } t \text{ made the } k\text{-th change, } k \leq \tau, \\ \max\big(0,\ \Lambda_{t-1} + X_t\big) & \text{otherwise,} \end{cases}$$

$$X_t = \begin{cases} \kappa_k - y_k & \text{round } t \text{ made the } k\text{-th change, } k > \tau, \\ \kappa - \min\{\bar{y}_t, \kappa\} & u_t = 0 \text{ and } k \geq \tau, \\ 1/(2\max\{G_k, D\}) & u_t = 0 \text{ and } k < \tau, \end{cases} \tag{3}$$

starting from $\Lambda_{-1} = 0$, and LOOPSTOP stops the loop at $\Lambda_t \geq \Lambda^*$. Before the warm-up a change is therefore not scored and resets Λ_t, and each unchanged round adds $1/(2\max\{G_k, D\})$, where G_k is the longest gap in rounds between the changes so far and D is the *tolerated delay*, i.e., the

Algorithm 1: LOOPSTOP, quality-blind stopping of a loop.

given: a memory τ in changes, a tolerated delay D in rounds, and how the artifact's type is cut into parts (Algorithm E.1)

1 $\lambda \leftarrow (\tau-1)/(\tau+1), \quad \kappa \leftarrow (\tau+1)/2, \quad \rho \leftarrow \tau/2, \quad \Lambda^* \leftarrow ((\tau-1)D-(\tau+1))/2$;

2 $\Lambda \leftarrow 0, z \leftarrow 0, M \leftarrow 0, V \leftarrow 0, k \leftarrow 0, y^* \leftarrow 1, \bar{y} \leftarrow 1, G \leftarrow 0, r \leftarrow 0$;

3 **for** *each round* $t = 1, 2, \ldots$ **do**

4 take the artifact ξ_{t+1} that the round leaves and $u \leftarrow$ the update (1) from ξ_t to ξ_{t+1};

5 **if** $u \neq 0$ **then**

6 $k \leftarrow k+1, \quad M \leftarrow \lambda M + u, \quad V \leftarrow \lambda V + \|u\|^2, \quad z \leftarrow \lambda z + 1, \quad y \leftarrow \|M\|^2/(zVw_{k-1})$;
$\triangleright$ $\|M\|^2/(zV) = c_k$ *and* $w_{k-1} = (1-\lambda)(1+\lambda^k)/((1+\lambda)(1-\lambda^k))$, *Theorem 2*

7 **if** $k = 1$ **then** $\bar{y} \leftarrow y$ **else** $\bar{y} \leftarrow \lambda\,\bar{y} + (1-\lambda)\,y$, as in (2);

8 **if** $k \geq \tau$ **then** $y^* \leftarrow \max(y^*, \bar{y})$, the warm-up ending at $k = \tau$;

9 **if** $k > \tau$ **then** $\Lambda \leftarrow \max\big(0,\ \Lambda + \min\{\kappa, 1+\rho\,(y^*-1)\} - y\big)$, the reference value κ_k of Section 3.3;

10 **if** $k \leq \tau$ **then** $G \leftarrow \max(G, t-r), \quad r \leftarrow t, \quad \Lambda \leftarrow 0$;

11 **else**

12 **if** $k \geq \tau$ **then** $\bar{y} \leftarrow \lambda\,\bar{y} + (1-\lambda), \quad \Lambda \leftarrow \max(0,\ \Lambda + \kappa - \min\{\bar{y}, \kappa\})$, a change of magnitude zero;

13 **if** $k < \tau$ **then** $\Lambda \leftarrow \Lambda + 1/(2\max\{G, D\})$, the loop's own gaps setting the pace, with $G = 0$ before any change;

14 **if** $\Lambda \geq \Lambda^*$ **then**

15 **return** stop, and deliver ξ_{t+1};

number of rounds after settling within which the loop should be stopped, which is the second setting of LOOPSTOP. Since every change before the warm-up resets Λ_t, a loop that stops changing after fewer than τ changes is stopped $2\Lambda^* \max\{G_k, D\}$ rounds after its last change, when that run of unchanged rounds has raised Λ_t to Λ^*. Algorithm 1 states the full LOOPSTOP procedure.

What the two settings mean. The memory τ is the number of recent changes over which the updates are averaged, and it should cover the few consecutive changes over which an improving loop keeps its direction, and no more. It fixes the decay $\lambda = (\tau-1)/(\tau+1)$, at which the weighted average has the variance of an unweighted average of τ updates. The settled coherence is then $w = 1/\tau$, and the reference $\kappa = (\tau+1)/2$ is the mean of y_k at a signal-to-noise ratio $r = 1$, and $\rho = \tau/2$ is a convention. The reference value of a change is capped at κ, and for a loop whose reference value has reached that cap the expected delay of Theorem 3(ii) below is $(\Lambda^* + \kappa)/(\kappa - 1)$. Setting it equal to the tolerated delay D gives the threshold $\Lambda^* = D(\kappa-1) - \kappa = ((\tau-1)D - (\tau+1))/2$ of Line 1 in Algorithm 1. Appendix G derives every constant from the two settings.

Theorem 3 (the false-stop probability and the delay of LOOPSTOP under independent statistics)**.** *Let* Λ_t *follow* (3)*,* $B = (1+\lambda)/(1-\lambda) = \tau$ *so that* $0 \leq y_k \leq B$ *at every change, and* $(x)^+ = \max\{x, 0\}$*. Let the* past *of a round be all observations before it and whether it changed the artifact.*

(i) Assume that the statistics y_k *of distinct changes are independent,* $\Lambda_{t_0} = 0$*, and each of the rounds* $t_0+1, \ldots, t_0+n$ *changes the artifact. When* $\mathbb{E}[y_k] \geq \kappa + \delta$ *at each of these changes for some* $\delta > 0$*,* LOOPSTOP *stops within these* n *rounds with probability at most* $n^2 \exp(-8\Lambda^*\delta/B^2)$*.*

(ii) Let $p > 1$ *and* $\kappa(p) = \min\{\kappa,\ 1+\rho\,(p-1)\}$*. Fix a round* t_1 *past the warm-up at which* $y^*_{t_1} \geq p$ *and* $\bar{y}_{t_1} \leq 1$*, and let every change after* t_1 *have* $y_k \leq 1$*. Then the expected number of rounds after* t_1 *until* LOOPSTOP *stops is at most* $(\Lambda^* + \kappa)/(\kappa(p) - 1)$*.*

The proof contains novel steps. Part (i) first establishes that the sum at the adaptive reference value is dominated by the sum at the fixed value κ, then proves by induction on (3) that Page's sum equals the maximum of the block sums over blocks of consecutive rounds. A stop within the window therefore implies that the sum over some block of changes, with each contribution within an interval of length B, reaches Λ^*. Hoeffding's inequality bounds that probability per block, and a union bound over at most n^2 blocks completes the argument. Part (ii) introduces a potential that is the cumulative sum without its floor at zero minus $c\,(\bar{y}_n - 1)^+$ with $c = \lambda/(1-\lambda)$, which subtracts a multiple of the positive excess of the smoothed statistic over 1, so that each increment has a conditional mean of at least $\kappa(p) - 1$, and a stopped-sum inequality for $\min\{N, n\}$ with monotone convergence in n yields the bound.

Intuitively, Theorem 3 says that LOOPSTOP stops an improving loop only by an unlikely accident and stops a settled loop within a bounded number of rounds. In Part (i), a loop whose changes keep their statistic above the reference value by a margin δ is stopped within n rounds only if the scores of some block of changes fall short of their mean by a total of Λ^*, whose probability decays exponentially in $\Lambda^*\delta$. In Part (ii), once a loop that has demonstrated a peak p has settled, every round adds at least $\kappa(p) - 1$ to the evidence on average, so the threshold is reached after about $\Lambda^*/(\kappa(p) - 1)$ rounds plus the overshoot of a single score.

4 EXPERIMENTS

The loops. The trajectories come from five *families* of loops, each of which is a benchmark run under several *scenarios*. A scenario is a single combination of evaluator kind and model size, and every scenario of a family runs the same tasks under the same seeds. The evaluator is none, a program, a model that accepts or rejects, or an inspector that names a problem. Qwen3.5 with 9 or 4 billion parameters fills every role on four benchmark families, and Claude Haiku 4.5 fills every role on the fifth benchmark family. The *cruise control* family tunes the two gains of a cruise controller for a simulated car, and the *cart-pole* family tunes the four gains of a controller that balances a pole on a cart, so in both the artifact is the vector of gains. The quality is the best achievable cost of a drive divided by the cost that the tuned controller achieves, in $(0, 1]$, on a drive that the loop does not see. An *ARC-AGI* task (Chollet, 2019) shows a few example pairs of grids and asks for the output grid of a new input, and the quality is the share of cells that match the hidden answer. The *BigCodeBench-Hard* family repairs a source file against a task's tests (Zhuo et al., 2025), hidden from the loop by our protocol, and its quality is the share of those tests passed, of which the evaluator sees at most a visible subset. The *MLAgentBench-Kaggle* family edits the model section of the starter script, which is the part left to the agent, on the tasks of that benchmark (Huang et al., 2024b) that are drawn from Kaggle, on tables, text and images. Its quality is the score on a hidden subset of the rows, which no role sees, and an evaluator sees only what the script prints. The harness holds the last script that ran, as a deployed harness keeps its last working version. Appendix H gives the prompt templates.

The protocol. As soon as LOOPSTOP triggers stopping for a benchmark task, we use the final artifact to evaluate against the ground truth to obtain the loop's performance. We generally repeat the evaluation for 20 seeds (5 on BigCodeBench-Hard, 3 on MLAgentBench-Kaggle due to resource constraints) and report the standard error. We compare against four baseline rules: *Run to the budget* never stops early and spends every round the harness allows, which is what many currently deployed agent frameworks do (Wang et al., 2025; Wu et al., 2024; Yang et al., 2024). *Patience on rejections (PoR)* stops after a set number of consecutive rejected rounds. *Patience on update magnitude (PoU)* stops after a set number of consecutive updates whose magnitudes are each below a fraction of the largest magnitude so far, which is semantic early stopping (Shrivastava, 2026) with our sketch in place of its neural embedding. Both are modeled on the stuck detector that OpenHands enables by default (OpenHands, 2026). *The run-length filter (RLF)* applies Bayesian changepoint detection (Adams & MacKay, 2007) to the coherence. It holds a posterior over the number of rounds since the coherence last changed behaviour, and stops once that posterior concentrates on a recent change and the coherence has averaged below one since. Alongside the four rules, we report the best budget per scenario that stops every loop of a scenario at the round where that scenario's mean quality is highest, which is an *idealized upper bound that cannot be achieved in practice* since the true quality in practice is unknown. Each of the three adaptive baseline rules has one free threshold, such as the number of rounds that a patience rule waits, and we perform grid search for each of them. A rule that is deployed on an unfamiliar benchmark cannot be tuned on it, so each baseline keeps the grid value that maximizes its mean quality over all 5 benchmarks as the threshold. In contrast, we do not tune the threshold for our LOOPSTOP algorithm. We fix the memory to $\tau = 5$ changes and its tolerated delay to $D = 4$ rounds for all the benchmark families. The true quality is used for the experimental evaluation alone and no algorithm (including our LOOPSTOP algorithm and the baselines) ever sees it, except for the idealized upper bound. Appendix I provides additional discussion on the exact configurations and additional ablations, including comparison to dynamic per-benchmark thresholds where we tune each baseline on a subset of benchmark scenarios and apply the result to a held-out scenario.

LOOPSTOP delivers at least as much quality as every baseline stopping rule on every family, at fewer rounds than the patience rules. Table 1 gives the rounds and the quality of every rule on every family and their means. LOOPSTOP's largest gains are on the controller families, where the quality falls after its peak and the single threshold that suits the rising families lets every baseline run the falling loops beyond the peak, so that LOOPSTOP leads by several standard errors. On the families where the loop keeps rising the quality, i.e., BigCodeBench-Hard and MLAgentBench-Kaggle, the gaps are smaller, and on ARC-AGI LOOPSTOP reaches its quality at fewer than half the rounds of the patience rules. Against the idealized upper bound, LOOPSTOP gives up a small amount of quality on the controller families, less on BigCodeBench-Hard and MLAgentBench-Kaggle, and none on ARC-AGI, where it exceeds the bound, since the bound uses the same number of rounds for every loop of a scenario whereas LOOPSTOP can adapt for each loop separately.

Table 1: Rounds used and quality delivered, mean over the seeds with the standard error as a subscript, best quality in bold. The Mean pair averages the five families.

	Cruise control		Cart-pole		ARC-AGI	
Rule	Rounds	Quality	Rounds	Quality	Rounds	Quality
Run to the budget	$100_{\pm 0}$	$0.393_{\pm 0.011}$	$100_{\pm 0}$	$0.204_{\pm 0.016}$	$60.0_{\pm 0}$	$0.650_{\pm 0.001}$
Idealized upper bound	$18.2_{\pm 0}$	$0.597_{\pm 0.015}$	$30.1_{\pm 0}$	$0.335_{\pm 0.016}$	$9.0_{\pm 0.0093}$	$0.654_{\pm 0.001}$
PoR	$64.1_{\pm 1.0}$	$0.411_{\pm 0.010}$	$58.1_{\pm 1.1}$	$0.210_{\pm 0.013}$	$51.9_{\pm 0.30}$	$0.650_{\pm 0.001}$
PoU	$56.5_{\pm 1.1}$	$0.402_{\pm 0.010}$	$53.5_{\pm 0.72}$	$0.223_{\pm 0.013}$	$48.2_{\pm 0.32}$	$0.653_{\pm 0.001}$
RLF	$21.1_{\pm 1.4}$	$0.353_{\pm 0.016}$	$6.2_{\pm 0.45}$	$0.207_{\pm 0.012}$	$5.0_{\pm 0.00043}$	$0.652_{\pm 0.001}$
LoopStop (ours)	$16.2_{\pm 0.36}$	$\mathbf{0.519}_{\pm 0.016}$	$26.1_{\pm 1.0}$	$\mathbf{0.278}_{\pm 0.017}$	$22.9_{\pm 0.052}$	$\mathbf{0.656}_{\pm 0.001}$

	BigCodeBench-Hard		MLAgentBench-Kaggle		**Mean**	
Rule	Rounds	Quality	Rounds	Quality	Rounds	Quality
Run to the budget	$60.0_{\pm 0}$	$0.579_{\pm 0.003}$	$60.0_{\pm 0}$	$0.489_{\pm 0.018}$	$76.0_{\pm 0}$	$0.463_{\pm 0.005}$
Idealized upper bound	$45.0_{\pm 0}$	$0.595_{\pm 0.004}$	$32.2_{\pm 0}$	$0.527_{\pm 0.016}$	$26.9_{\pm 0.0019}$	$0.542_{\pm 0.006}$
PoR	$37.2_{\pm 0.14}$	$0.565_{\pm 0.002}$	$40.7_{\pm 0.78}$	$0.478_{\pm 0.019}$	$50.4_{\pm 0.34}$	$0.463_{\pm 0.005}$
PoU	$44.0_{\pm 0.21}$	$0.577_{\pm 0.002}$	$37.7_{\pm 1.6}$	$0.490_{\pm 0.011}$	$48.0_{\pm 0.42}$	$0.469_{\pm 0.004}$
RLF	$5.1_{\pm 0.0067}$	$0.546_{\pm 0.007}$	$6.1_{\pm 0.12}$	$0.437_{\pm 0.014}$	$8.7_{\pm 0.30}$	$0.439_{\pm 0.005}$
LoopStop (ours)	$28.5_{\pm 0.16}$	$\mathbf{0.584}_{\pm 0.004}$	$23.7_{\pm 0.70}$	$\mathbf{0.501}_{\pm 0.017}$	$23.5_{\pm 0.26}$	$\mathbf{0.508}_{\pm 0.006}$

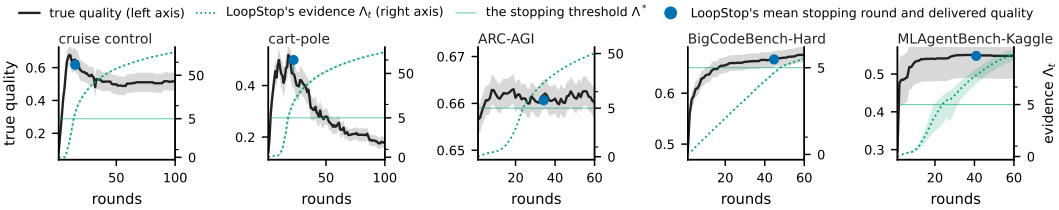

Figure 2: *A single scenario* of each family by round, with the narrow simulation-based evaluator on the controllers, the 9B model evaluator on ARC-AGI, and the visible tests on BigCodeBench-Hard, all with the 9B generator, and the validation check with Haiku on MLAgentBench-Kaggle. Plots show the mean true quality with a band of one standard error *on the left axis*, LoopStop's evidence Λ_t and threshold Λ^* *on the right axis*, and its mean stopping round and quality as a marker.

Table 2: Mean cost per round, and the quality of LoopStop with the sketch and with a sentence embedding of the artifact. The sketch column includes the algorithm's work per round.

	Latency by method (**milliseconds**)		Loop's running time	Quality of LoopStop	
Benchmark	Sketch (CPU)	Embedding (CPU)	per round (**seconds**)	Sketch	Embedding
cruise control	$0.018_{\pm 0.001}$	$8.570_{\pm 0.023}$	$0.150_{\pm 0.008}$	$0.519_{\pm 0.016}$	$0.508_{\pm 0.013}$
cart-pole	$0.020_{\pm 0.001}$	$10.200_{\pm 0.130}$	$0.050_{\pm 0.001}$	$0.278_{\pm 0.017}$	$0.283_{\pm 0.019}$
ARC-AGI	$0.088_{\pm 0.001}$	$19.700_{\pm 0.055}$	$0.460_{\pm 0.025}$	$0.656_{\pm 0.001}$	$0.656_{\pm 0.001}$
BigCodeBench-Hard	$1.290_{\pm 0.037}$	$85.900_{\pm 0.021}$	$0.530_{\pm 0.083}$	$0.584_{\pm 0.004}$	$0.585_{\pm 0.003}$
MLAgentBench-Kaggle	$24.400_{\pm 1.500}$	$87.000_{\pm 0.056}$	$124.000_{\pm 41.000}$	$0.501_{\pm 0.017}$	$0.500_{\pm 0.015}$

Running a loop longer is not always better, and LoopStop stops near the peak. In Figure 2, the two controller scenarios rise to an early peak and then fall, and LoopStop stops near the peak. The BigCodeBench-Hard and MLAgentBench-Kaggle scenarios rise quickly and then hold, and LoopStop stops after the rise has ended.

Our sketch is efficient to run. Table 2 times the sketch against a sentence embedding of the artifact by a small sentence-transformer `all-MiniLM-L6-v2` on the same CPU, where the sketch's cost grows with the size of the edit and the embedding's cost with the size of the whole artifact. On every family, LoopStop's quality with the embedding is within one standard error of its quality with the sketch, so the sketch gives a similar quality at a much lower cost.

5 Conclusion

In this paper, we have proposed LoopStop, which is a novel algorithm that stops an agentic loop from the change that each round applies to the artifact and the evaluator's accept or reject decision alone, without knowing the artifact's true quality. It sketches each artifact so that a round's update is its edit, measures the coherence of the recent updates against a settled value from theory, and accumulates the evidence for stopping in a cumulative sum. On five benchmark families, LoopStop improves the delivered quality with about half the rounds of the next best baseline rule. A limitation is that the analysis assumes that successive coherence statistics are independent, and relaxing this assumption is left for future work.

AI USE STATEMENT

In this work, we used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat datasets, and interpret results. We have not used generative AI tools to generate synthetic data sets, assist with translation, or support qualitative and thematic data analysis. Additionally, we used generative AI tools to create or edit software code, summarize or analyze existing literature, and edit a research paper to improve readability. We have reviewed all AI-assisted work. In particular, we reviewed the theorem statements and proofs, the code, and the text, and every reported number was produced by the released code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The code is available at `https://anonymous.4open.science/r/loopstop-C0C1`. It holds LOOPSTOP, the sketch, every baseline, the harness that ran the five families of loops with one run script per family, the scripts that fetch the benchmark data, and the analysis chain that computes every table and figure of the paper from the recorded rounds, with a README that lists the reproduction steps in order and a smoke test that runs without data. Appendix I gives every setting of every rule, and Appendix H every prompt template. Full proofs of every stated result are in Appendix C.

REFERENCES

Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

Hidayet Aksu. Odds law: The decomposition algebra on how intelligence organizes itself to solve difficult problems reliably. *arXiv preprint arXiv:2606.15712*, 2026.

Johin Johny Arimbur. How many tries does it take? Iterative self-repair in LLM code generation across model scales and benchmarks. *arXiv preprint arXiv:2604.10508*, 2026.

Islem Bouzenia and Michael Pradel. Understanding software engineering agents: A study of thought-action-result trajectories. In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 2846–2857, 2025.

Hongyu Chen, Liang Lin, and Guangrun Wang. SVR: Self-verifying refinement via joint verdict-confidence reinforcement learning for adaptive test-time compute. *arXiv preprint arXiv:2607.28457*, 2026.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

John H. Cochrane. How big is the random walk in GNP? *Journal of Political Economy*, 96(5): 893–920, 1988.

Sunny Dubey. Real-time detection and repair of LLM agent failures. *arXiv preprint arXiv:2608.02464*, 2026.

Paul Fearnhead and Zhen Liu. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B*, 69(4):589–605, 2007.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866, 2023.

Kim Hammar, Tansu Alpcan, and Emil C. Lupu. Optimal stopping of self-refining foundation models. In *Proceedings of the 65th IEEE Conference on Decision and Control (CDC)*, 2026. To appear; arXiv:2608.10729.

Jiangang Han. Partially correlated verifier cascades in LLM harnesses: Concave log-odds, polynomial reliability, and blind-spot ceilings. *arXiv preprint arXiv:2607.13918*, 2026.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Tianyu Hu, Zhen Tan, Song Wang, Huaizhi Qu, and Tianlong Chen. Multi-agent debate for LLM judges with adaptive stability detection. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024a.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309, 2024b.

Shashidhar Reddy Javaji, Bhavul Gauri, and Zining Zhu. Another turn, better output? A turn-wise analysis of iterative LLM prompting. *arXiv preprint arXiv:2509.06770*, 2025. Presented at the NeurIPS 2025 Workshop on Multi-Turn Interactions in Large Language Models.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Myeongsoo Kim, Dingmin Wang, Siwei Cui, Farima Farmahinifarahani, Terry Yue Zhuo, Shweta Garg, Baishakhi Ray, Rajdeep Mukherjee, and Varun Kumar. Coherence collapse: Diagnosing why code agents fail after reaching the right code. *arXiv preprint arXiv:2603.24631*, 2026.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. BAGEN: Are LLM agents budget-aware? *arXiv preprint arXiv:2606.00198*, 2026.

Aofan Liu and Jingxiang Meng. Self-correction as feedback control: Error dynamics, stability thresholds, and prompt interventions in LLMs. *arXiv preprint arXiv:2604.22273*, 2026.

Andrew W. Lo and A. Craig MacKinlay. Stock market prices do not follow random walks: Evidence from a simple specification test. *The Review of Financial Studies*, 1(1):41–66, 1988.

Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.

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. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Oorja Majgaonkar, Zhiwei Fei, Xiang Li, Federica Sarro, and He Ye. Understanding code agent behaviour: An empirical study of success and failure trajectories. *arXiv preprint arXiv:2511.00197*, 2025.

Tural Mehtiyev and Wesley Assunção. Beyond resolution rates: Behavioral drivers of coding agent success and failure. *arXiv preprint arXiv:2604.02547*, 2026.

OpenHands. Stuck detector, 2026. Software documentation, `https://docs.openhands.dev/sdk/guides/agent-stuck-detector`, accessed September 2026.

Addy Osmani. Loop engineering, 2026. Online essay, https://addyosmani.com/blog/loop-engineering/.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Marut Pandya, Kasey Zhang, and Baiqing Lyu. Strained coherence: A pre-failure signal in coding agent execution trajectories. *arXiv preprint arXiv:2606.07889*, 2026.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Kai Ruan, Zihe Huang, Ziqi Zhou, Qianshan Wei, Jinghao Lin, Xuan Wang, and Hao Sun. Doomed from the start: Early abort of LLM agent episodes via a recall-controlled probe cascade. *arXiv preprint arXiv:2607.06503*, 2026.

Shuvom Sadhuka, Drew Prinster, Clara Fannjiang, Gabriele Scalia, Bonnie Berger, Aviv Regev, and Hanchen Wang. E-valuator: Reliable agent verifiers with sequential hypothesis testing. *arXiv preprint arXiv:2512.03109*, 2025.

Erik Schluntz and Barry Zhang. Building effective agents, December 2024. Anthropic engineering documentation, https://www.anthropic.com/engineering/building-effective-agents.

I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938.

A. N. Shiryaev. On optimum methods in quickest detection problems. *Theory of Probability and Its Applications*, 8(1):22–46, 1963.

Sahil Shrivastava. Semantic early-stopping for iterative LLM agent loops. *arXiv preprint arXiv:2606.27009*, 2026.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *International Conference on Learning Representations*, 2025.

Alexander G. Tartakovsky and Venugopal V. Veeravalli. General asymptotic Bayesian theory of quickest change detection. *Theory of Probability and Its Applications*, 49(3):458–497, 2005.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1113–1120, 2009.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling (COLM)*, 2024.

Xuening Wu, Qianya Xu, Yanlan Kang, Zeping Chen, Yubin Liu, and Shenqin Yin. Do language models converge to themselves? Recursive self-refinement as textual relaxation. *arXiv preprint arXiv:2607.22653*, 2026.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, pp. 4184–4195, 2026.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xiangxin Zhao, Han Li, Shuaiting Li, Tianyi Zhao, Earl T. Barr, Federica Sarro, and He Ye. Failure as a process: An anatomy of CLI coding agent trajectories. *arXiv preprint arXiv:2607.09510*, 2026.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, David Lo, Binyuan Hui, Niklas Muennighoff, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025.

CONTENTS OF THE APPENDIX

A Extended Related Work **15**

B Notation **17**

C Proofs of the Results in the Main Text **18**

C.1 Proof of Theorem 1 . . . 18

C.2 Proof of Theorem 2 . . . 18

C.3 Proof of Theorem 3 . . . 19

D Supporting Results **22**

D.1 The Bound on the Remaining Change . . . 22

D.2 The Bound on the Coherence . . . 22

E The Construction in Detail **24**

F LOOPSTOP in Detail **28**

G How the Constants of LOOPSTOP Follow from Its Two Settings **31**

H Prompt Templates **35**

H.1 Cruise Control . . . 35

H.2 Cart-Pole . . . 36

H.3 ARC-AGI . . . 37

H.4 BigCodeBench-Hard . . . 38

H.5 MLAgentBench-Kaggle . . . 39

I Experiments in Detail **43**

I.1 The Standard Error . . . 43

I.2 The Loops in Detail . . . 43

I.3 The Two Settings and the Sketch . . . 45

I.4 Sensitivity to the Memory and the Delay . . . 46

I.5 The Settings of the Sketch . . . 46

I.6 The Thresholds, Fixed Once and Cross-Fitted . . . 47

I.7 The Selection Effect . . . 47

I.8 Two Rules Under a Sentence Embedding . . . 48

I.9 Why LOOPSTOP Takes the Rounds That It Takes . . . 49

I.10 The Quality Remaining past the Best Round . . . 50

I.11 The Rules Split by Mechanism . . . 50

I.12 LOOPSTOP with the Unchanged Rounds Left Unscored . . . 50

I.13 The Rules Compared . . . 51

I.14 A Family of Views . . . 51

I.15 The Cruise-Control Family by Evaluator . . . 52

I.16 Paired Differences by Seed . . . 52

I.17 The Best Budget per Scenario, Cross-Fitted . . . 52

I.18 Every Scenario on Its Own . . . 53

I.19 The Sources of the Evidence . . . 55

A Extended Related Work

The rounds at which agentic loops stop. Measurements of iterative refinement place the gain in the first few rounds, and analyses of self-correction predict the level at which a loop settles. Madaan et al. (2023) measures iterative self-refinement and reports the gain flattening after a few rounds. Arimbur (2026) runs up to five execution-feedback repair rounds over 7 models and finds most of the gain in the first two. Javaji et al. (2025) scores every turn of 12-turn refinement conversations and finds the gain arriving early for code and ideas. Wu et al. (2026) track recursive self-refinement of paper abstracts and find that the edit magnitude per round decays and saturates, which is the quantity that patience on update magnitude thresholds. Huang et al. (2024a) and Stechly et al. (2025) show that a model asked to check its own reasoning frequently cannot, and Kamoi et al. (2024) surveys the scenarios under which self-correction helps. Aksu (2026) and Han (2026) predict the level at which a cascade of a generator and an evaluator settles from the rate at which the evaluator accepts the generator's mistakes. Liu & Meng (2026) models self-correction as a two-state Markov chain and derives the threshold at which repeated correction helps or hurts. Panickssery et al. (2024) measures how an evaluator's judgments shift toward a generator that shares its weights, and Gao et al. (2023) how optimizing against a proxy reward model degrades the true reward. None of these works gives an online rule that decides, for a particular loop in progress, that the remaining rounds no longer justify their cost.

Trajectories of coding agents. Studies of coding-agent trajectories find that failed trajectories cycle, are longer, and overwrite correct edits. Bouzenia & Pradel (2025) studies 120 trajectories of three repair agents and finds repetitive cycles of the same actions in the failed trajectories. Majgaonkar et al. (2025) finds the failed trajectories of three agents on SWE-bench (Jimenez et al., 2024) consistently longer, and describes unproductive reasoning loops. Zhao et al. (2026) finds, on 1,794 command-line agent trajectories, that failures begin in the first steps and stay hidden until recovery is impossible. Kim et al. (2026) finds that most failures of capable agents reach and edit the correct code, and that overwriting it is the largest failure theme, which is a direct reason to observe the artifact path itself. Mehtiyev & Assunção (2026) shows over 9,374 trajectories that the correlation between the length of a trajectory and its failure reverses once task difficulty is controlled. The length of a trajectory alone is therefore a confounded signal. None of these studies proposes a stopping statistic.

Stopping rules for a loop in progress. The baselines decide from a score of the quality, the model's own self-report, or data from earlier episodes. Hammar et al. (2026) frames the number of self-refinement rounds as an optimal stopping problem that trades expected improvement against the cost of a call. It solves that problem by stochastic approximation, with a verifier that observes the quality at every round. Sadhuka et al. (2025) turns verifier scores over an agent's actions into a sequential procedure with a controlled false alarm rate, again on a score of the quality. Chen et al. (2026) trains the policy to emit a verdict and a confidence at each turn and stops on them. Lin et al. (2026) asks the agent to predict its remaining budget and stop when success appears unlikely. Both rules delegate the decision to the model's own self-report. Pandya et al. (2026) flags spans of a transcript where a judge finds the agent acknowledging a problem and proceeding anyway. Dubey (2026) runs a cumulative sum on anomaly scores learned from telemetry of earlier episodes. Ruan et al. (2026) abort an agent episode early from linear probes on the model's hidden activations, calibrated to a recall target, which observes the model's internals and not the artifact. Shrivastava (2026) halts a loop between a drafting agent and a critic once the cosine distance between consecutive drafts under a model embedding stays below a threshold for a patience window and a quality-stagnation check agrees. That threshold is tuned on held-out experiment runs. Hu et al. (2025) stops a multi-agent debate once a check finds the distribution of the judges' votes stable. Xi et al. (2026) trains a process reward model that scores each action by its progress, for training and for selection. None of these rules states what can be concluded from the artifact path alone or gives the level that a path statistic takes once the updates have stopped sharing a direction.

Quickest change detection. Quickest change detection decides online that an observed signal has changed its distribution. Page (1954) gives the cumulative sum rule, which in its likelihood-ratio form (Lorden, 1971) accumulates the log likelihood ratio between a pre-change and a post-change distribution and stops when it exceeds a threshold. Shiryaev (1963) places a geometric prior on the change point and derives the posterior rule that minimizes the expected delay under a bound on the false alarm probability. Lorden (1971) gives the worst-case delay analysis of the cumulative sum

rule. Tartakovsky & Veeravalli (2005) gives the asymptotic Bayesian theory covering dependent observations. Adams & MacKay (2007) and Fearnhead & Liu (2007) keep a posterior over the length of the current segment instead of a two-state latent variable. That handles many change points at the cost of memory that grows with the segment. Applying these rules to an agentic loop is not a direct application, for three reasons. The quality is not observed while the loop runs, so the series has to be formed from the artifact path, whose elements are source files or grids instead of scalars, and none of these works says which observable of a loop carries the event. The likelihood-ratio form needs the distribution of the series before and after the change, and the posterior forms need a hazard rate and a predictive model of the observations. For a loop these have to be set by hand or fit on earlier loops, which is what the run-length filter of Section 4 does and LOOPSTOP avoids. Theorem 2 gives the level that the coherence takes once the loop has settled, in closed form and for every loop alike, so LOOPSTOP forms Page's sum against that level with no distribution estimated. A loop with an evaluator also leaves most rounds unchanged, so the series is formed over the changes and the unchanged rounds are scored on their own (Section 3.3), which none of these rules provides.

Separating a path with a common direction from a path without. The variance ratio of Lo & MacKinlay (1988) and Cochrane (1988) separates a series whose increments have a non-zero mean from a series whose increments do not. Let the variance ratio denote the variance of a W-update increment divided by W times the variance of a one-update increment. It is 1 for a path with independent increments and departs from 1 when the increments correlate. Lo & MacKinlay (1988) uses it on asset prices and Cochrane (1988) uses it to size the permanent component of output. In both applications the series itself is the quantity of interest, and its history is long and under a fixed regime. Applying the ratio to a loop is not direct either. The artifact path is a vector path of a few dozen updates whose regime is what changes, and its quality is held out. The coherence forms the ratio on the vector updates with a decay, so that only the recent updates count, and Theorem 2 gives its value at every finite window and inverts it to the signal-to-noise ratio of the updates. Neither work gives that value or that inversion.

B NOTATION

Table B.1: The notation of the main text, in order of appearance.

Symbol	Meaning
t, T	the index of a round and the number of rounds that the harness allows
ξ_t	the artifact that round t starts from, with ξ_0 the initial artifact
A_t	the accept decision of round t, $A_t = 1$ when the change was applied
$q(\xi)$	the true quality of an artifact, in $[0, 1]$, which the loop never consults
$\Delta_{t,W}$	the change in quality over the W rounds after round t
$\mathbb{P}$, $\mathbb{P}'$	the distributions of the loop scored by q and by $1 - q$
C, ζ	an event about the observations alone and a margin on the change in quality
$d(\xi, \xi')$, L	the distance between two artifacts and the Lipschitz constant of q in d
$S(\xi)$, m	the sketch of an artifact, a vector in $\mathbb{R}^m$, and its dimension
ω, v	the name of a part and the quantity that it carries
P_t	the set of parts of ξ_t
n, s	the number of tokens per part of a text (in Theorem 3, n is a number of rounds) and the scale of a numeric entry
H, $h(\omega)$, $\sigma(\omega)$	the hash function, the coordinate and the sign that it gives a name
e_r	the unit vector of coordinate r
u_t	the update of round t, zero when the round is rejected
S_t	the sketch under names with the round index t appended
x_t	the path, the accumulated updates $\sum_{s<t} u_s$
$\|\cdot\|$	the magnitude of a vector, its Euclidean norm
λ	the decay factor of the averages, in $(0, 1)$
M_t, V_t	the decay-weighted averages of the updates and of their squared magnitudes
z_t	the sum of the decay weights
c_t	the coherence $\|M_t\|^2/V_t$ of the artifact path
r	the squared signal-to-noise ratio $\|\mathbb{E}u\|^2/\mathbb{E}\|u - \mathbb{E}u\|^2$ of an update
w_t, w	the settled coherence at t updates and its limit $(1 - \lambda)/(1 + \lambda)$
τ	the memory of the statistic in changes, $(1 + \lambda)/(1 - \lambda)$
y_t	the normalized coherence c_t/w_t
k	the number of changes, i.e., rounds with a non-zero update, so far
y_k	the normalized coherence formed over the changes, at the k-th change
$\bar{y}_t$, y_t^*	the smoothed statistic and the demonstrated peak of (2)
κ, κ', κ_k	the mean of y_k at $r = 1$, a fixed reference value, and the reference value of change k
ρ	the slope that converts the demonstrated peak into the reference value
X_t, Λ_t, Λ^*	the score of round t, the cumulative sum of (3), and the stopping threshold
G_k, D	the longest gap in rounds between the changes so far and the tolerated delay
B	the upper bound $(1 + \lambda)/(1 - \lambda)$ on y_k
δ, p	the margin and the peak in the hypotheses of Theorem 3
t_0, t_1	the round at which the cumulative sum starts and a round past the warm-up in Theorem 3

C Proofs of the Results in the Main Text

C.1 Proof of Theorem 1

Assumption 1 states that no part of the loop consults q. Fix t and W with $t + W \le T$, so that $\Delta_{t,W}$ is defined. The proposal, the verdict and the update of round t are each a function of what the loop has produced before that round, namely the artifacts, the proposals and the verdicts, and of the role's own randomness. Composing these functions round by round, every artifact $\xi_1, \dots, \xi_T$ and every indicator $A_0, \dots, A_{T-1}$ is a function of the initial artifact and of the roles' randomness alone, and by Assumption 1 the quality q enters none of these functions. Replacing q by $1 - q$ therefore leaves every artifact and indicator unchanged, so the distribution of $\xi_1, \dots, \xi_T$ and $A_0, \dots, A_{T-1}$ is the same in the two loops. Substituting $1 - q$ into the definition of the change,

$$\big(1 - q(\xi_{t+W})\big) - \big(1 - q(\xi_t)\big) = -\big(q(\xi_{t+W}) - q(\xi_t)\big) = -\Delta_{t,W},$$

along every path.

Every event C that is a statement about the observations alone therefore has $\mathbb{P}'(C) = \mathbb{P}(C)$, which is the first claim. Now take $\zeta > 0$ and such an event C with $\mathbb{P}(C) > 0$, so that $\mathbb{P}'(C) > 0$ and the conditional probabilities under both distributions are defined. Let H be the event $\{\Delta_{t,W} < -\zeta\}$. In the second loop the change in quality along a path is $\Delta'_{t,W} = -\Delta_{t,W}$ by the equality above, so

$$\mathbb{P}'\big(\Delta'_{t,W} > \zeta \mid C\big) \;=\; \mathbb{P}'\big(-\Delta_{t,W} > \zeta \mid C\big) \;=\; \mathbb{P}'\big(H \mid C\big) \;=\; \mathbb{P}\big(H \mid C\big).$$

The last equality holds because H and C are both determined by the artifacts, whose distribution is the same in the two loops, and $\mathbb{P}(H \mid C)$ is the right side of the second claim. □

C.2 Proof of Theorem 2

Let $\mu = \mathbb{E}u$ denote the mean update and $\epsilon_s = u_s - \mu$ the deviation of round s, and

$$z_t = \sum_{j=0}^{t} \lambda^j = \frac{1 - \lambda^{t+1}}{1 - \lambda}, \qquad z'_t = \sum_{j=0}^{t} \lambda^{2j} = \frac{1 - \lambda^{2(t+1)}}{1 - \lambda^2},$$

so that $M_t = \frac{1}{z_t} \sum_j \lambda^j u_{t-j}$ and $V_t = \frac{1}{z_t} \sum_j \lambda^j \|u_{t-j}\|^2$, and $r = \|\mu\|^2 / \mathbb{E}\|\epsilon\|^2$.

Taking the expectation of the numerator and splitting each update into its mean and its deviation,

$$\begin{aligned} z_t^2\, \mathbb{E}\|M_t\|^2 &= \mathbb{E}\Big\| \sum_{j=0}^{t} \lambda^j \mu + \sum_{j=0}^{t} \lambda^j \epsilon_{t-j} \Big\|^2 \\ &= z_t^2 \|\mu\|^2 + 2\, z_t\, \mu^\top \sum_{j=0}^{t} \lambda^j\, \mathbb{E}\epsilon_{t-j} + \sum_{j=0}^{t} \sum_{k=0}^{t} \lambda^{j+k}\, \mathbb{E}\big[\epsilon_{t-j}^\top \epsilon_{t-k}\big] \\ &= z_t^2 \|\mu\|^2 + \sum_{j=0}^{t} \lambda^{2j}\, \mathbb{E}\|\epsilon\|^2 = z_t^2 \|\mu\|^2 + z'_t\, \mathbb{E}\|\epsilon\|^2, \end{aligned}$$

where the middle term vanishes because $\mathbb{E}\epsilon_{t-j} = \mathbb{E}u - \mu = 0$ at every j. In the double sum, a term with $j \neq k$ has ϵ_{t-j} and ϵ_{t-k} built from two different updates, which are independent. The expectation of the product therefore factors,

$$\mathbb{E}\big[\epsilon_{t-j}^\top \epsilon_{t-k}\big] = \big(\mathbb{E}\epsilon_{t-j}\big)^\top \big(\mathbb{E}\epsilon_{t-k}\big) = 0,$$

and only the $j = k$ terms remain, each equal to $\mathbb{E}\|\epsilon\|^2$.

Taking the expectation of the denominator and using $\mathbb{E}\|u\|^2 = \|\mu\|^2 + \mathbb{E}\|\epsilon\|^2$,

$$\mathbb{E}[V_t] = \frac{1}{z_t} \sum_{j=0}^{t} \lambda^j\, \mathbb{E}\|u_{t-j}\|^2 = \|\mu\|^2 + \mathbb{E}\|\epsilon\|^2.$$

Dividing the two and then dividing numerator and denominator by $z_t^2\, \mathbb{E}\|\epsilon\|^2$,

$$\frac{\mathbb{E}\|M_t\|^2}{\mathbb{E}[V_t]} = \frac{z_t^2 \|\mu\|^2 + z'_t\, \mathbb{E}\|\epsilon\|^2}{z_t^2 \big(\|\mu\|^2 + \mathbb{E}\|\epsilon\|^2\big)} = \frac{r + z'_t / z_t^2}{r + 1}.$$

Evaluating z_t'/z_t^2 with the two geometric sums,

$$\frac{z_t'}{z_t^2} = \frac{\dfrac{1-\lambda^{2(t+1)}}{1-\lambda^2}}{\left(\dfrac{1-\lambda^{t+1}}{1-\lambda}\right)^2} = \frac{(1-\lambda)^2\left(1-\lambda^{t+1}\right)\left(1+\lambda^{t+1}\right)}{(1-\lambda)(1+\lambda)\left(1-\lambda^{t+1}\right)^2} = \frac{1-\lambda}{1+\lambda}\cdot\frac{1+\lambda^{t+1}}{1-\lambda^{t+1}} = w_t,$$

where the second equality factors $1-\lambda^{2(t+1)}$ and $1-\lambda^2$. Setting $t\to\infty$ sends λ^{t+1} to zero and w_t to $(1-\lambda)/(1+\lambda)$. Also w_t decreases in t, because $(1+y)/(1-y)$ increases in y and $y=\lambda^{t+1}$ decreases in t.

For the inversion, let $\vartheta = (r+w_t)/(r+1)$ denote the ratio of the two means and solve for r,

$$\vartheta\, r + \vartheta = r + w_t \implies r\,(\vartheta-1) = w_t - \vartheta \implies r = \frac{\vartheta - w_t}{1-\vartheta},$$

which is defined for $\vartheta < 1$. □

C.3 Proof of Theorem 3

Let X_t denote the contribution of round t in (3). On a round that made the k-th change, $X_t = \kappa_k - y_k$ with $1 \le \kappa_k \le \kappa$, and on a round that changed nothing after the warm-up, $X_t = \kappa - \min\{\bar{y}, \kappa\}$ with $\bar{y} > 0$ (Appendix F). In both cases $X_t \le \kappa$, and $\kappa - y_k$ lies in the interval $[\kappa - B, \kappa]$ of length B, by the bound on y_k in Appendix F. Every round considered below is past the warm-up. Let $\bar{y}_t$ denote the value of the smoothed statistic after round t, which Algorithm 1 revises on every round after the warm-up, by $\bar{y}_t = \lambda\,\bar{y}_{t-1} + (1-\lambda)\,y_t$ on a changing round, with y_t the statistic of that change, and by $\bar{y}_t = \lambda\,\bar{y}_{t-1} + (1-\lambda)$ on an unchanged round. The proof rests on five lemmas, each proved in full.

Lemma C.3.1 (a fixed reference value stops no later)**.** *Let Λ_t^{κ} be the sum of* (3) *formed from the same rounds with κ in place of κ_k on every changing round, where both sums start at zero at a round t_0. Then $\Lambda_t \le \Lambda_t^{\kappa}$ for every $t \ge t_0$.*

Proof. By induction on t. Both sums are zero at t_0. Let X_t^{κ} denote the contribution with the fixed reference value, which is $\kappa - y_k$ on a changing round and equals X_t on an unchanged round. Since $\kappa_k \le \kappa$, $X_t \le X_t^{\kappa}$ at every round. If $\Lambda_{t-1} \le \Lambda_{t-1}^{\kappa}$, then $\Lambda_{t-1} + X_t \le \Lambda_{t-1}^{\kappa} + X_t^{\kappa}$, and since $z \mapsto \max(0, z)$ is non-decreasing, $\Lambda_t = \max(0, \Lambda_{t-1} + X_t) \le \max(0, \Lambda_{t-1}^{\kappa} + X_t^{\kappa}) = \Lambda_t^{\kappa}$. □

Lemma C.3.2 (Page's sum is the largest block sum)**.** *Let $\Lambda_{t_0} = 0$ and let $S_{i,t} = X_i + \cdots + X_t$ for $t_0 < i \le t$, with $S_{t+1,t} = 0$ for the empty block. Then $\Lambda_t = \max_{t_0 < i \le t+1} S_{i,t}$ for every $t \ge t_0$.*

Proof. By induction on t. At $t = t_0$ both sides are zero, where the right side holds only the empty block. If the claim holds at $t-1$, then adding X_t to every block sum ending at $t-1$ gives every block sum ending at t with a non-empty block, so $\Lambda_{t-1} + X_t = \max_{t_0 < i \le t} S_{i,t}$, and taking the maximum with zero adds the empty block, which is the recursion of (3). □

Lemma C.3.3 (a sum of bounded independent terms rarely exceeds its mean by much)**.** *Let $Z_1, \ldots, Z_j$ be independent, let each Z_i lie in an interval $[a_i, a_i + B]$ of length B, and let $a \ge 0$. Then*

$$\mathbb{P}\Big(\sum_{i=1}^{j} Z_i - \sum_{i=1}^{j} \mathbb{E}[Z_i] \ge a\Big) \;\le\; \exp\Big(-\frac{2a^2}{jB^2}\Big).$$

This is Hoeffding's inequality (Hoeffding, 1963), and its proof is included so that the argument is self-contained.

Proof. Let $W_i = Z_i - \mathbb{E}[Z_i]$, which has mean zero and lies in the interval $[b_i, b_i + B]$ with $b_i = a_i - \mathbb{E}[Z_i]$, where $b_i \le 0 \le b_i + B$ since the mean lies in the interval. Fix $s > 0$. The first step bounds $\mathbb{E}[e^{sW_i}]$. The function $x \mapsto e^{sx}$ is convex, so on the interval $[b_i, b_i + B]$ it lies below the chord through its endpoints,

$$e^{sx} \;\le\; \frac{b_i + B - x}{B}\, e^{sb_i} + \frac{x - b_i}{B}\, e^{s(b_i + B)} \qquad \text{for } b_i \le x \le b_i + B.$$

Substituting $x = W_i$ and taking expectations, with $\mathbb{E}[W_i] = 0$,

$$\mathbb{E}[e^{sW_i}] \;\le\; \frac{b_i + B}{B}\, e^{sb_i} - \frac{b_i}{B}\, e^{s(b_i+B)}.$$

Let $\theta = -b_i/B$, which lies in $[0,1]$, and let $u = sB$. The right side is then $(1-\theta)\, e^{-\theta u} + \theta\, e^{(1-\theta)u} = e^{-\theta u}(1 - \theta + \theta e^u) = e^{L(u)}$ with $L(u) = -\theta u + \log(1 - \theta + \theta e^u)$. Differentiating twice, with $\pi(u) = \theta e^u/(1 - \theta + \theta e^u)$, which lies in $[0,1]$,

$$L'(u) = -\theta + \pi(u), \qquad L''(u) = \frac{\theta e^u (1-\theta)}{(1 - \theta + \theta e^u)^2} = \pi(u)\,(1 - \pi(u)) \;\le\; \frac{1}{4},$$

since $\pi(1-\pi) \le 1/4$ on $[0,1]$. With $L(0) = 0$ and $L'(0) = -\theta + \theta = 0$, Taylor's theorem with the remainder at a point between 0 and u gives $L(u) \le u^2/8$, hence $\mathbb{E}[e^{sW_i}] \le e^{s^2 B^2/8}$. The second step is Markov's inequality, $\mathbb{P}(Y \ge c) \le \mathbb{E}[Y]/c$ for a non-negative Y and $c > 0$, applied to $Y = e^{s\sum_i W_i}$ and $c = e^{sa}$,

$$\mathbb{P}\Big(\sum_{i=1}^{j} W_i \ge a\Big) \;\le\; e^{-sa}\,\mathbb{E}\big[e^{s\sum_i W_i}\big] \;=\; e^{-sa}\prod_{i=1}^{j}\mathbb{E}[e^{sW_i}] \;\le\; \exp\Big(-sa + \frac{s^2 j B^2}{8}\Big),$$

where the product uses the independence of the W_i. Choosing $s = 4a/(jB^2)$ makes the exponent $-4a^2/(jB^2) + 2a^2/(jB^2) = -2a^2/(jB^2)$, which is the claim. □

Lemma C.3.4 (a stopped sum of terms with a positive conditional mean)**.** *Index the rounds after some round as $1, 2, \ldots$, let $Y_1, Y_2, \ldots$ be bounded, and let $\mathcal{F}_i$ denote the past of round i, namely everything observed before round i together with whether round i changed the artifact. Let the mean of Y_i given $\mathcal{F}_i$ be at least $m > 0$ for every i. Let N be a stopping round, meaning that whether $N \ge i$ is determined by the observations before round i, and suppose that $Y_1 + \cdots + Y_{\min\{N,n\}} \le K$ on every path for every n. Then $\mathbb{E}[N] \le K/m$. This is the argument behind Wald's equation for a stopped sum, and it is proved here in full.*

Proof. Fix n. Since $\min\{N,n\} = \sum_{i=1}^{n} \mathbf{1}[N \ge i]$,

$$\sum_{i=1}^{\min\{N,n\}} Y_i \;=\; \sum_{i=1}^{n} Y_i\, \mathbf{1}[N \ge i].$$

The indicator $\mathbf{1}[N \ge i]$ is determined by $\mathcal{F}_i$, so the mean of the product $Y_i\,\mathbf{1}[N \ge i]$ equals the mean of $\mathbf{1}[N \ge i]$ times the mean of Y_i given $\mathcal{F}_i$, which is at least m, so $\mathbb{E}[Y_i\,\mathbf{1}[N \ge i]] \ge m\,\mathbb{P}(N \ge i)$. Taking the mean of the equality above term by term, which is a finite sum of bounded terms, and using the bound on every path,

$$K \;\ge\; \mathbb{E}\Big[\sum_{i=1}^{n} Y_i\,\mathbf{1}[N \ge i]\Big] \;\ge\; m\sum_{i=1}^{n}\mathbb{P}(N \ge i) \;=\; m\,\mathbb{E}[\min\{N,n\}].$$

So $\mathbb{E}[\min\{N,n\}] \le K/m$ at every n. As n grows, $\min\{N,n\}$ increases to N, so $\mathbb{E}[N] = \lim_n \mathbb{E}[\min\{N,n\}] \le K/m$ by monotone convergence. □

Lemma C.3.5 (a single round of the potential)**.** *Let $e_t = (\bar{y}_t - 1)^+$ and $c = \lambda/(1-\lambda)$, so that $c\,(1-\lambda) = \lambda$, and let $Y_t = X_t - c\,(e_t - e_{t-1})$, which is X_t corrected by a term that absorbs the direction that the smoothed statistic still carries. For rounds after a round at which $y^* \ge p$, $Y_t \ge \kappa(p) - 1$ on an unchanged round and $Y_t \ge \kappa(p) - y_t - \lambda\,(y_t - 1)^+$ on a changing round.*

Proof. The demonstrated peak is a running maximum, so $y^*_k \ge p$ at every later change, and $x \mapsto \min\{\kappa, 1 + \rho(x-1)\}$ is non-decreasing, so $\kappa_k \ge \kappa(p)$ on a changing round, while the value κ of an unchanged round is also at least $\kappa(p)$. On an unchanged round $\bar{y}_t = \lambda\,\bar{y}_{t-1} + (1-\lambda)$, so $\bar{y}_t - 1 = \lambda\,(\bar{y}_{t-1} - 1)$ and, since $\lambda > 0$, $e_t = \lambda\, e_{t-1}$. Also $\min\{\bar{y}_t, \kappa\} \le \bar{y}_t \le 1 + e_t$, so $X_t \ge \kappa - 1 - e_t$. Hence

$Y_t \;\ge\; \kappa - 1 - e_t - c\,(\lambda e_{t-1} - e_{t-1}) \;=\; \kappa - 1 - \lambda\, e_{t-1} + c\,(1-\lambda)\, e_{t-1} \;=\; \kappa - 1 \;\ge\; \kappa(p) - 1.$

On a changing round $\bar{y}_t = \lambda\,\bar{y}_{t-1} + (1-\lambda)\,y_t$, so $\bar{y}_t - 1 = \lambda\,(\bar{y}_{t-1} - 1) + (1-\lambda)\,(y_t - 1)$. Since $(a+b)^+ \le a^+ + b^+$ and $(\alpha a)^+ = \alpha a^+$ for $\alpha \ge 0$, $e_t \le \lambda\, e_{t-1} + (1-\lambda)\,(y_t - 1)^+$. Hence, using

$X_t \geq \kappa(p) - y_t$, $c\,(1-\lambda) = \lambda$ and $e_{t-1} \geq 0$,

$$\begin{aligned} Y_t \;&\geq\; \kappa(p) - y_t - c\big(\lambda\, e_{t-1} + (1-\lambda)\,(y_t - 1)^+\big) + c\, e_{t-1} \\ &=\; \kappa(p) - y_t + \lambda\, e_{t-1} - \lambda\,(y_t - 1)^+ \;\geq\; \kappa(p) - y_t - \lambda\,(y_t - 1)^+. \end{aligned}$$

□

Part (i). By Lemma C.3.1 it suffices to bound the probability that Λ_t^κ reaches Λ^* within the n rounds, and by Lemma C.3.2 applied to Λ^κ, which also starts from zero at t_0, that happens only if some block of $j \leq n$ consecutive rounds inside the window has a sum of fixed-value contributions of at least Λ^*. Every round of the window changes the artifact, so such a block sum is $\sum(\kappa - y_k)$ over j changes, a sum of j independent terms, each in an interval of length B, with mean at most $-j\delta$, since $\mathbb{E}[y_k] \geq \kappa + \delta$ at each change. Apply Lemma C.3.3 with $a = \Lambda^* - (\text{mean of the block sum}) \geq \Lambda^* + j\delta > 0$. Since $(\Lambda^* + j\delta)^2 \geq 4\Lambda^* j\delta$, which is $(\Lambda^* - j\delta)^2 \geq 0$ rearranged,

$$\mathbb{P}\big(\text{the block sum reaches } \Lambda^*\big) \;\leq\; \exp\Big(-\frac{2(\Lambda^* + j\delta)^2}{jB^2}\Big) \;\leq\; \exp\Big(-\frac{8\Lambda^*\delta}{B^2}\Big).$$

The window holds at most n^2 blocks, and the probability of a union is at most the sum of the probabilities, so the probability that Λ^κ, and hence Λ, reaches Λ^* inside the window is at most $n^2 \exp(-8\Lambda^*\delta/B^2)$. LoopStop stops when Λ reaches the threshold, which is the claim.

Part (ii). We prove a more general bound and then specialize it. Let the smoothed statistic at t_1 be $\bar{y}$, and let every change after t_1 have, given the past of its round, $\mathbb{E}[y_k] \leq 1$ and $\mathbb{E}[(y_k - 1)^+] \leq \varepsilon$ with $\kappa(p) > 1 + \lambda\varepsilon$. We show that the expected number of rounds after t_1 until the stop is at most $(\Lambda^* + \kappa + \lambda\,(\bar{y} - 1)^+/(1-\lambda))/(\kappa(p) - 1 - \lambda\,\varepsilon)$, and the statement of the theorem is the case $\bar{y} \leq 1$ and $y_k \leq 1$, in which $\varepsilon = 0$ and $(\bar{y} - 1)^+ = 0$. Every mean below is conditional on the state of the algorithm at t_1. Index the rounds after t_1 as $1, 2, \ldots$, let $\Lambda'_n = X_1 + \cdots + X_n$ be the sum without the floor at zero, with $\Lambda'_0 = 0$, and let $N = \min\{n : \Lambda'_n \geq \Lambda^*\}$. First, $\Lambda_n \geq \Lambda'_n$ at every n, by induction. At $n = 0$ the sum Λ_0 is at least zero by (3) and $\Lambda'_0 = 0$. If $\Lambda_{n-1} \geq \Lambda'_{n-1}$, then $\Lambda_n = \max(0, \Lambda_{n-1} + X_n) \geq \Lambda_{n-1} + X_n \geq \Lambda'_{n-1} + X_n = \Lambda'_n$. So LoopStop stops within N rounds after t_1, and it is enough to bound $\mathbb{E}[N]$. Second, apply Lemma C.3.4 to the Y_n of Lemma C.3.5, with $e_0 = (\bar{y} - 1)^+$ at t_1. The terms Y_n are bounded, since X_n lies in $[1 - B, \kappa]$ and $0 \leq e_n \leq B - 1$. Given the past of round n, an unchanged round has $Y_n \geq \kappa(p) - 1$ with no randomness left, and a changing round has a conditional mean of at least $\kappa(p) - \mathbb{E}[y_n \mid \mathcal{F}_n] - \lambda\,\mathbb{E}[(y_n - 1)^+ \mid \mathcal{F}_n] \geq \kappa(p) - 1 - \lambda\varepsilon$ by the hypotheses of the theorem, so $m = \kappa(p) - 1 - \lambda\varepsilon > 0$ serves for both kinds of round. The event $\{N \geq i\}$ is the event that $\Lambda'_1, \ldots, \Lambda'_{i-1}$ all lie below Λ^*, which is determined by the observations before round i. Third, on every path the sum of the Y_i telescopes in e,

$$\sum_{i=1}^{\min\{N,n\}} Y_i \;=\; \Lambda'_{\min\{N,n\}} - c\, e_{\min\{N,n\}} + c\, e_0 \;\leq\; \Lambda^* + \kappa + c\, e_0,$$

because $e_{\min\{N,n\}} \geq 0$, and because either $N > n$, so that $\Lambda'_n < \Lambda^*$, or $N \leq n$, so that $\Lambda'_N = \Lambda'_{N-1} + X_N < \Lambda^* + \kappa$, as $\Lambda'_{N-1} < \Lambda^*$ and $X_N \leq \kappa$. Lemma C.3.4 with $K = \Lambda^* + \kappa + c\, e_0$ gives $\mathbb{E}[N] \leq K/m$, which is the stated bound with $e_0 = (\bar{y} - 1)^+$ and $c = \lambda/(1-\lambda)$. Under the hypotheses of the theorem, $\bar{y} \leq 1$ at t_1 and every later $y_k \leq 1$, so $e_0 = 0$ and $(y_k - 1)^+ = 0$, the hypotheses above hold with $\varepsilon = 0$, and the bound is $(\Lambda^* + \kappa)/(\kappa(p) - 1)$. □

Why the argument gives nothing at $p = 1$. At $p = 1$ the reference value is $\kappa(1) = 1$, the lower bound on $\mathbb{E}[X_t]$ on a changing round is $\kappa(1) - 1 = 0$, and on a loop that keeps changing Lemma C.3.4 above gives $0 \geq 0$, which fixes no $\mathbb{E}[\min\{N, n\}]$. This gap belongs to the algorithm and not to the argument. Under Assumption 3 with equality, which Theorem 2 delivers for changes that are independent with a common distribution, every X_t on a changing round has mean exactly zero. Then Λ_t is a random walk reflected at zero with mean-zero increments, and the round at which it first reaches Λ^* is fixed by the variance of y_k, which no assumption of this paper supplies.

D SUPPORTING RESULTS

D.1 THE BOUND ON THE REMAINING CHANGE

The following result turns Assumption 2 into the bound on the change in quality that Section 2 states.

Theorem D.1 (the remaining change is bounded by observables and L)**.** *Under Assumption 2 with $d(\xi_s, \xi_t) \le \|x_s - x_t\|$ for the path vectors of Section 3.1, with the updates $u_t, \dots, u_{t+W-1}$ independent with a common distribution, $\mathbb{E}\|u\|^2 < \infty$ and $\mathbb{E}\|u - \mathbb{E}u\|^2 > 0$, and with $r = \|\mathbb{E}u\|^2 / \mathbb{E}\|u - \mathbb{E}u\|^2$,*

$$\mathbb{E}\big|\Delta_{t,W}\big| \;\le\; L\sqrt{\frac{\mathbb{E}\|u\|^2\, W\,(Wr+1)}{r+1}},$$

which is $L\sqrt{\mathbb{E}\|u\|^2\, W}$ at $r = 0$ and grows as W once $Wr \gg 1$.

Proof. Applying Assumption 2 to the two artifacts, then $d(\xi_t, \xi_{t+W}) \le \|x_{t+W} - x_t\|$ and $x_{t+W} - x_t = \sum_{s=1}^{W} u_{t+s-1}$,

$$\big|q(\xi_{t+W}) - q(\xi_t)\big| \;\le\; L\,\Big\|\sum_{s=1}^{W} u_{t+s-1}\Big\|.$$

Let Z denote $\|\sum_{s=1}^{W} u_{t+s-1}\|$, which is non-negative. The variance of Z is non-negative, so $0 \le \mathbb{E}Z^2 - (\mathbb{E}Z)^2$ and therefore $\mathbb{E}Z \le \sqrt{\mathbb{E}Z^2}$. Taking expectations of both sides of the inequality above and applying that bound gives

$$\mathbb{E}\big|q(\xi_{t+W}) - q(\xi_t)\big| \;\le\; L\sqrt{\mathbb{E}\Big\|\sum_{s=1}^{W} u_{t+s-1}\Big\|^2}.$$

Let $\mu = \mathbb{E}u$ denote the mean update and $\epsilon_s = u_s - \mu$ the deviation of round s, so that $\mathbb{E}\epsilon_s = 0$. Expanding the square of the sum of the W updates,

$$\mathbb{E}\Big\|\sum_{s=1}^{W} u_{t+s-1}\Big\|^2 = W^2\|\mu\|^2 + 2\,W\,\mu^\top \sum_{s=1}^{W} \mathbb{E}\,\epsilon_{t+s-1} + \sum_{s=1}^{W}\sum_{s'=1}^{W} \mathbb{E}\big[\epsilon_{t+s-1}^\top \epsilon_{t+s'-1}\big].$$

The middle group vanishes because $\mathbb{E}\epsilon_{t+s-1} = 0$ at every s. In the double sum, a term with $s \ne s'$ has two deviations built from different updates, which are independent, so $\mathbb{E}[\epsilon_{t+s-1}^\top \epsilon_{t+s'-1}] = (\mathbb{E}\epsilon_{t+s-1})^\top (\mathbb{E}\epsilon_{t+s'-1}) = 0$. The W terms with $s = s'$ each equal $\mathbb{E}\|\epsilon\|^2$, so

$$\mathbb{E}\Big\|\sum_{s=1}^{W} u_{t+s-1}\Big\|^2 = W^2\|\mu\|^2 + W\,\mathbb{E}\|\epsilon\|^2.$$

Substituting $\|\mu\|^2 = \mathbb{E}\|u\|^2\, r/(1+r)$ and $\mathbb{E}\|\epsilon\|^2 = \mathbb{E}\|u\|^2/(1+r)$, which follow from $\mathbb{E}\|u\|^2 = \|\mu\|^2 + \mathbb{E}\|\epsilon\|^2$ and $r = \|\mu\|^2/\mathbb{E}\|\epsilon\|^2$,

$$W^2\|\mu\|^2 + W\,\mathbb{E}\|\epsilon\|^2 = \frac{\mathbb{E}\|u\|^2\, W^2 r}{1+r} + \frac{\mathbb{E}\|u\|^2\, W}{1+r} = \frac{\mathbb{E}\|u\|^2\, W\,(Wr+1)}{r+1},$$

which gives the stated bound. At $r = 0$ the factor $Wr + 1$ is 1 and the bound is $L\sqrt{\mathbb{E}\|u\|^2\, W}$, which grows as $\sqrt{W}$. Once $Wr \gg 1$ the same factor is close to Wr and the bound is close to $L\,W\sqrt{\mathbb{E}\|u\|^2\, r/(r+1)}$, which grows as W. □

D.2 THE BOUND ON THE COHERENCE

The claim is that $\|M_t\|^2 \le V_t$, so that $c_t \in [0, 1]$. Letting $g_j = \lambda^j / \sum_{i=0}^{t} \lambda^i$, so that the g_j are non-negative and sum to 1, $M_t = \sum_j g_j u_{t-j}$ and $V_t = \sum_j g_j \|u_{t-j}\|^2$. The triangle inequality gives $\|M_t\| \le \sum_j g_j \|u_{t-j}\|$. Let $c_j = \|u_{t-j}\|$. Expanding the square and using $\sum_k g_k = 1$ in the first and the last sum,

$$\sum_{j,k} g_j g_k (c_j - c_k)^2 = \sum_{j,k} g_j g_k c_j^2 - 2\sum_{j,k} g_j g_k c_j c_k + \sum_{j,k} g_j g_k c_k^2 = 2\sum_j g_j c_j^2 - 2\Big(\sum_j g_j c_j\Big)^2.$$

The left side is a sum of non-negative terms, so $\sum_j g_j c_j^2 - \left(\sum_j g_j c_j\right)^2$, which is half of it, is non-negative. Squaring the triangle inequality and applying this gives $\|M_t\|^2 \leq \left(\sum_j g_j c_j\right)^2 \leq \sum_j g_j c_j^2 = V_t$. Both quantities are non-negative, so $c_t \in [0, 1]$. □

E THE CONSTRUCTION IN DETAIL

The distances that place the artifacts in a vector space. By a classical characterization, a distance is carried by an embedding of the artifacts into a vector space exactly when its square is conditionally negative definite (Schoenberg, 1938). That property means that for every finite set of artifacts and every set of weights summing to zero, the weighted sum of the pairwise squared distances is at most zero. Any distance meeting it would serve in Theorem E.1(ii) in place of the sketch, so the sketch below is a choice among such distances and not a special case.

The set of parts. An artifact is turned into a set of parts, in which each part is a name paired with a quantity. Three choices are left, namely what a name is, what quantity it carries, and how a set of parts becomes a vector. The paragraphs below take them in turn, and a paragraph after those shows that the unit, the origin, and the orientation of the vector space cannot change the answer.

The granularity and the order that it carries. Let the *granularity* denote how finely text is cut, namely a token, a token pair or a line, and let a *view* denote a way of turning what the loop leaves at a round, namely the artifact, the proposal or the transcript, cut at a chosen granularity, into a set of parts and so into a vector path. Demand R3 requires a granularity small enough that the number of changed names tracks the amount of the artifact that the round changed, and a token, a token pair, and a line all satisfy it. A single name for a whole artifact does not, since every edit is then the same size regardless of what it changed, and a name for each position is positional, which R2 rules out. Order is carried by the granularity and not by the name. A single token carries no order, so a round that only reorders content gives a zero update in the first two terms of (1), which the third term records. A token pair or a line changes at every boundary that the reordering changes, and the update then grows with the size of the reordering. This is why several granularities are offered as candidate views. A name in a repository is the path of a file followed by a token of that file. Every occurrence of that token in that file gives the same name.

Deciding the kind. Algorithm E.1 is the rule that turns any artifact into its parts. The kind of each piece is decided by its type, so nothing is decided by hand. The rule descends through structured data to its leaves while carrying the path to each leaf as a prefix of the leaf's names, and π/ω denotes the name ω under the prefix π. The items of a sequence that is not a vector of scalars are named by their content under the common prefix $\pi/[\,]$, so an insertion touches the inserted item alone. A sequence of scalars is a vector, whose coordinates are named by their index, since there the index is the coordinate's name. A grid of the ARC-AGI family is cut by its own rule, into a single part per cell named by the cell's position and color, since the positions of a grid are fixed.

Algorithm E.1: The parts of an artifact by type. n is the granularity of text and s_i the magnitude of coordinate i at the start of the loop, or 1 if that is zero, and $s_i = 1$ for a scalar inside structured data.

```
function Parts(x, π)
    // π is the path to x, empty at the root
    if x is a string then
        return {(π/ω, 1)} for every run ω of n consecutive tokens of x, with multiplicity;
    else if x is a scalar or a sequence of scalars x_1, ..., x_d then
        return {(π/i, sign(x_i) log(1 + |x_i|/s_i))} for every coordinate i;
    else if x is a mapping from keys ω to values v then
        return the union over the pairs (ω, v) of Parts(v, π/ω);
    else if x is a sequence of values v then
        return the union over the values v of Parts(v, π/[]);
    else
        return {(π/ω, 1)} with ω the printed form of x;
```

The quantity a part carries. A name occurring several times in an artifact contributes their total, so multiplicity is carried by the quantity and not by the name. A numeric part takes $\operatorname{sign}(v)\log(1+|v|/s)$, with v the value at that coordinate. Here $s > 0$ is the magnitude of that coordinate at the start of the loop, or 1 if that magnitude is zero. The outer sign makes the map odd, meaning that $-v$ is sent to the negative of what v is sent to, and strictly increasing, so v and $-v$ give quantities of the same size and opposite sign instead of the same quantity. A coordinate that starts at a small non-zero value has a small s, so a given absolute change there counts as a large relative change, which is what R3 requires, whereas a coordinate that starts at zero has no magnitude to measure against and takes

$s = 1$. The quantity a part carries is therefore not continuous in the start across zero. A floor on s would remove that discontinuity at the cost of a further setting, and the experiments do not need it, since every gain of the two controller families starts at a non-zero value.

The sketch and what the sign is for. Let the hash function H of Section 3.1 be modeled as a family whose values on distinct names are independent and uniform. The coordinate and the sign of a name ω are taken from it,

$$h(\omega) \;=\; \big\lfloor H(\omega)/2 \big\rfloor \bmod m, \qquad \sigma(\omega) \;=\; \begin{cases} +1 & \text{if } H(\omega) \text{ is odd,} \\ -1 & \text{if } H(\omega) \text{ is even.} \end{cases}$$

The two are separate functions of the hash, and a pair of names sharing a coordinate may still carry opposite signs. Taking the coordinate as $H(\omega) \bmod m$ instead would not separate them. Every dimension used here is even, and for an even m the integers $H(\omega)$ and $H(\omega) \bmod m$ have the same parity. The sign would then be a function of the coordinate, and every colliding pair would carry the same sign. Every mean and variance over the sketch in Theorem E.2 is over the single draw of H, with h and σ held fixed for the whole loop. A name therefore maps to the same coordinate at every round with the same sign, which is what makes two rounds comparable. The *sketch* $S(P)$ of a set P of parts is then

$$S(P) \;=\; \sum_{(\omega, v) \,\in\, P} \sigma(\omega)\, v\, e_{h(\omega)}, \qquad e_r \text{ the } r\text{-th coordinate vector,} \tag{E.1}$$

and the update of round t is formed from the sketches of the parts of ξ_t and ξ_{t+1} by (1).

The size of the artifact is divided out. The update of a text or structured artifact is divided by the square root of the mean number of parts of the two artifacts, counted with multiplicity, which is the magnitude that the sketch of either artifact would have if no name repeated and no names collided. A change of a given number of parts is then an update whose magnitude is relative to the size of the artifact. A numeric artifact is left undivided, since the magnitude of a coordinate there is the quantity itself. The divisor is a scalar applied to the whole update, so the first two terms of (1) stay the difference of the two sketches up to that scalar, which is what Theorem E.3 states.

The unit, the origin and the orientation have no effect. Two elementary facts settle the remaining choices, and Theorem E.1 states them. Let $\alpha \neq 0$ be a scalar, let Q be an orthogonal matrix, and let β be a vector. Replacing every artifact x_s by $\alpha Q x_s + \beta$ carries every update u_s to $\alpha Q u_s$, since the shift β cancels in the difference. Every inner product between updates is then multiplied by α^2, and any ratio of two of them is unchanged. The unit, the origin and the orientation therefore need not be fixed. The sketch is the exact count vector over the names multiplied by a matrix with a single entry ± 1 per column, in row $h(\omega)$ with sign $\sigma(\omega)$. If no two names collide, that matrix preserves every inner product, as an orthogonal Q preserves them, so the arbitrary choice of h and σ is such a Q up to the collision term above. Theorem E.1 recovers every inner product between updates from four distances, so the parts are a way of supplying a distance between artifacts and not the only way.

The granularity and the view. A token, a token pair, and a whole line all satisfy R3 and register different edits, and the artifact, the proposal and the transcript are three views of the same loop. Appendix F gives two checks that drop the views that do not belong.

A loop, traced. Table E.1 follows three views of a single loop through the construction of Section 3.1 and through the two checks of Appendix F. An agent is asked to stop a parser crashing on empty input. The repository holds two files. Over three rounds the agent adds a guard to the first, replaces the guard's condition and removes surrounding whitespace, and then proposes a rename that the evaluator rejects. Round 1 adds 7 parts to the artifact view and removes no part, and all of the added parts are tokens under the path of the first file. The second file contributes 7 parts to every sketch and appears in no update, which is the cancellation of unchanged parts in (1).

Table E.1: Three views of the same loop. *parts* counts distinct names in the first and last sketch, and added and removed count the names whose quantity increased or decreased each round. The last two columns are the outcomes of the two checks of Appendix F, whether the view's part quantities only ever grow and whether the view changes on rejected rounds as much as on accepted ones.

view	a part of it	parts	added	removed	only grows	changes regardless
artifact	a token of a file	$18 \to 24$	$7, 6, 0$	$0, 1, 0$	no	no
transcript	a token of a turn	$8 \to 31$	$11, 13, 8$	$0, 0, 0$	yes	yes
proposal	a token of the proposal	$22 \to 24$	$6, 0, 1$	$1, 0, 1$	no	yes

Supporting results. The three results below state the invariance of the coherence and its dependence on distances alone, the guarantee of the sketch, and the equality of the update and the change, on which the construction above relies.

Theorem E.1 (invariance and the kernel form)**.** *(i) For every $\alpha \neq 0$, orthogonal Q and vector β, the artifacts $\alpha Q x_s + \beta$ give the same c_t.*

(ii) Let d be a distance carried by a map of the artifacts into a vector space, so that $d(\xi, \xi') = \|x - x'\|$ for the vectors x and x' that the map gives, and let the updates be the differences of those vectors, $u_t = x_{t+1} - x_t$. Then

$$\langle u_s,\, u_t\rangle \;=\; \tfrac{1}{2}\Big(d^2(\xi_{s+1},\xi_t) + d^2(\xi_s,\xi_{t+1}) - d^2(\xi_{s+1},\xi_{t+1}) - d^2(\xi_s,\xi_t)\Big),$$

so c_t is determined by the distances between artifacts and by nothing else. For the update of (1) *this covers its first two terms, since the third is not a difference of two sketches.*

Proof. For the invariance, the update of round s under the mapped artifacts is

$$(\alpha Q x_{s+1} + \beta) - (\alpha Q x_s + \beta) = \alpha Q(x_{s+1} - x_s) = \alpha Q u_s,$$

since β cancels. Let $z_t = \sum_{j=0}^{t} \lambda^j$, so that $M_t = \frac{1}{z_t}\sum_j \lambda^j u_{t-j}$ and $V_t = \frac{1}{z_t}\sum_j \lambda^j \|u_{t-j}\|^2$. Substituting the mapped updates and using $\|Qv\| = \|v\|$ for orthogonal Q,

$$\frac{1}{z_t}\sum_j \lambda^j \alpha Q u_{t-j} = \alpha\, Q\, M_t, \qquad \big\|\alpha\, Q\, M_t\big\|^2 = \alpha^2 \|M_t\|^2, \qquad \frac{1}{z_t}\sum_j \lambda^j \|\alpha Q u_{t-j}\|^2 = \alpha^2 V_t.$$

Dividing, the factor α^2 cancels and c_t is unchanged.

For the kernel form, expand $d^2(x,y) = \|x-y\|^2 = \|x\|^2 - 2\langle x, y\rangle + \|y\|^2$ over the four pairs and add them with the signs of the claim,

$$\begin{aligned} &d^2(x_{s+1},x_t) + d^2(x_s,x_{t+1}) - d^2(x_{s+1},x_{t+1}) - d^2(x_s,x_t) \\ &\quad= \|x_{s+1}\|^2 - 2\langle x_{s+1},x_t\rangle + \|x_t\|^2 \\ &\qquad + \|x_s\|^2 - 2\langle x_s,x_{t+1}\rangle + \|x_{t+1}\|^2 \\ &\qquad - \|x_{s+1}\|^2 + 2\langle x_{s+1},x_{t+1}\rangle - \|x_{t+1}\|^2 \\ &\qquad - \|x_s\|^2 + 2\langle x_s,x_t\rangle - \|x_t\|^2. \end{aligned}$$

Every squared magnitude cancels, and collecting the four inner products,

$$= 2\big(\langle x_{s+1},x_{t+1}\rangle - \langle x_{s+1},x_t\rangle - \langle x_s,x_{t+1}\rangle + \langle x_s,x_t\rangle\big) = 2\,\langle x_{s+1} - x_s,\; x_{t+1} - x_t\rangle,$$

which is the stated equality after dividing by two. Both $\|M_t\|^2$ and V_t are sums of such inner products with fixed weights, namely $\|M_t\|^2 = \frac{1}{z_t^2}\sum_{j,k}\lambda^{j+k}\langle u_{t-j}, u_{t-k}\rangle$ and $V_t = \frac{1}{z_t}\sum_j \lambda^j \langle u_{t-j}, u_{t-j}\rangle$, so both quantities are determined by the distances and so is their ratio. □

Theorem E.2 (the sketch preserves the inner products)**.** *Let a and b be the exact count vectors of two artifacts over a common set of names, and let $\hat{a}$ and $\hat{b}$ be their sketches. Let the coordinates $h(\omega)$ be pairwise independent, meaning that any two of them are independent though all of them together need not be, and uniform on the m coordinates, and let the signs $\sigma(\omega)$ be independent and uniform on $\{+1,-1\}$ and independent of h. Then*

$$\mathbb{E}\langle \hat{a},\hat{b}\rangle = \langle a,b\rangle, \qquad \mathrm{Var}\langle \hat{a},\hat{b}\rangle \;\le\; \frac{\|a\|^2\|b\|^2 + \langle a,b\rangle^2}{m}.$$

Proof. Index the count vectors by the names, so that a_ω is the quantity of the name ω in the first artifact. Let $\hat{a}_r = \sum_\omega \sigma(\omega)\, a_\omega\, \mathbf{1}[h(\omega) = r]$ and likewise for $\hat{b}$. Summing over the coordinates and using $\sum_r \mathbf{1}[h(\omega)=r]\mathbf{1}[h(\omega')=r] = \mathbf{1}[h(\omega)=h(\omega')]$,

$$\langle \hat{a},\hat{b}\rangle = \sum_{\omega,\omega'} \sigma(\omega)\sigma(\omega')\, a_\omega b_{\omega'}\, \mathbf{1}[h(\omega)=h(\omega')]. \tag{E.2}$$

The signs are independent of the coordinates, so an expectation over both factors into an expectation over σ times an expectation over h. Take the expectation over σ first. Here $\mathbb{E}[\sigma(\omega)\sigma(\omega')] = \mathbf{1}[\omega = \omega']$, since for $\omega \neq \omega'$ the two signs are independent and each sign has mean zero, while for $\omega = \omega'$ the product is $\sigma(\omega)^2 = 1$. Every term with $\omega \neq \omega'$ therefore vanishes, and the terms with $\omega = \omega'$ have $\mathbf{1}[h(\omega) = h(\omega)] = 1$. So $\mathbb{E}\langle \hat{a},\hat{b}\rangle = \sum_\omega a_\omega b_\omega = \langle a,b\rangle$, whatever h is.

For the second moment, squaring (E.2) and naming the four names of a term $\omega_1, \omega_2, \omega_3, \omega_4$,

$$\mathbb{E}\langle \hat{a}, \hat{b}\rangle^2 = \sum_{\omega_1,\omega_2,\omega_3,\omega_4} \mathbb{E}\big[\sigma(\omega_1)\sigma(\omega_2)\sigma(\omega_3)\sigma(\omega_4)\big]\, a_{\omega_1} b_{\omega_2} a_{\omega_3} b_{\omega_4} \times \mathbb{E}\big[\mathbf{1}[h(\omega_1) = h(\omega_2)]\, \mathbf{1}[h(\omega_3) = h(\omega_4)]\big].$$

A product of four independent uniform signs has expectation 1 when every name in it appears an even number of times and zero otherwise. A name appearing once contributes a mean-zero factor independent of the rest. For $(\omega_1, \omega_2, \omega_3, \omega_4)$ that leaves three families, which do not overlap. If $\omega_1 = \omega_2$, evenness forces $\omega_3 = \omega_4$, and that is the first family. If $\omega_1 \neq \omega_2$, then ω_1 must appear again among ω_3 and ω_4, and so must ω_2. So either $\omega_3 = \omega_1$ and $\omega_4 = \omega_2$, or $\omega_3 = \omega_2$ and $\omega_4 = \omega_1$, and those are the second and the third. Only pairs of coordinates enter the expectations below, so pairwise independence of h suffices.

- $\omega_1 = \omega_2$ and $\omega_3 = \omega_4$. Both indicators equal 1, and the sum is $\big(\sum_\omega a_\omega b_\omega\big)^2 = \langle a, b\rangle^2$.
- $\omega_1 = \omega_3$ and $\omega_2 = \omega_4$ with $\omega_1 \neq \omega_2$. Both indicators are $\mathbf{1}[h(\omega_1) = h(\omega_2)]$, whose square is itself and whose expectation is $1/m$, and the sum is $\frac{1}{m}\sum_{\omega\neq\omega'} a_\omega^2 b_{\omega'}^2 \leq \frac{1}{m}\|a\|^2\|b\|^2$.
- $\omega_1 = \omega_4$ and $\omega_2 = \omega_3$ with $\omega_1 \neq \omega_2$. The indicators again coincide, and the sum is $\frac{1}{m}\sum_{\omega\neq\omega'} a_\omega b_\omega a_{\omega'} b_{\omega'} = \frac{1}{m}\big(\langle a, b\rangle^2 - \sum_\omega a_\omega^2 b_\omega^2\big) \leq \frac{1}{m}\langle a, b\rangle^2$.

Adding the three and subtracting the square of the mean,

$$\mathrm{Var}\langle \hat{a}, \hat{b}\rangle = \mathbb{E}\langle \hat{a}, \hat{b}\rangle^2 - \langle a, b\rangle^2 \;\leq\; \frac{\|a\|^2\|b\|^2 + \langle a, b\rangle^2}{m},$$

which is the stated bound. Omitting the signs replaces $\mathbb{E}[\sigma(\omega)\sigma(\omega')]$ by one in (E.2), so the mean gains the term $\sum_{\omega\neq\omega'} a_\omega b_{\omega'} \mathbb{E}\mathbf{1}[h(\omega) = h(\omega')] = \frac{1}{m}\sum_{\omega\neq\omega'} a_\omega b_{\omega'} \geq 0$ for non-negative counts, which is the upward bias. □

The hypotheses hold in the model of Appendix E, in which H takes independent uniform values on distinct names. The coordinate h uses the bits of $H(\omega)$ above the lowest and the sign σ uses the lowest bit, so the two are independent, and the range of $\lfloor H(\omega)/2 \rfloor$ is a power of two and so a multiple of the dimension $m = 256$, which makes h exactly uniform.

Theorem E.3 (the update is the change)**.** *Let $S(P_t)$ and $S(P_{t+1})$ be the sketches of the parts at the two ends of round t. Then the first two terms of* (1) *equal $S(P_{t+1}) - S(P_t)$ exactly, and the number of non-zero terms in that difference is the number of names whose quantity changed, so computing them costs a single pass over the change and nothing over the unchanged parts. Dividing the update by a scalar, such as the square root of the mean number of parts, keeps both statements up to that scalar.*

Proof. Let P_t and P_{t+1} be the sets of parts of the two artifacts, and for a name ω let a_ω and b_ω be its quantity in each artifact, zero where the name is absent. By (E.1) the two sketches are $S(P_t) = \sum_\omega \sigma(\omega)\, a_\omega\, e_{h(\omega)}$ and $S(P_{t+1}) = \sum_\omega \sigma(\omega)\, b_\omega\, e_{h(\omega)}$, with e_r the r-th coordinate vector. Subtracting the two,

$$S(P_{t+1}) - S(P_t) = \sum_\omega \sigma(\omega)\, b_\omega\, e_{h(\omega)} - \sum_\omega \sigma(\omega)\, a_\omega\, e_{h(\omega)} = \sum_\omega \sigma(\omega)\, (b_\omega - a_\omega)\, e_{h(\omega)},$$

and every name with $b_\omega = a_\omega$ contributes the zero vector, so the sum ranges over the names whose quantity changed. Splitting that sum by the sign of $b_\omega - a_\omega$ gives the sketch of the parts added minus the sketch of the parts removed, which are the first two terms of (1). The number of non-zero terms is the number of names whose quantity changed, which is the size of the change. Dividing the whole update by a scalar multiplies every term by that scalar, so the equality and the number of terms are preserved up to it. □

F LoopStop in Detail

The bound on y_k. The coherence lies in $[0, 1]$ by Appendix D.2, and $w_{k-1} \geq (1-\lambda)/(1+\lambda)$ at every change because w_k decreases to that limit. So $y_k = c_k/w_{k-1}$ lies in $[0, B]$ with $B = (1+\lambda)/(1-\lambda)$. The contribution at the fixed reference value κ, which is $\kappa - y_k$, therefore lies in the interval $[\kappa - B, \kappa]$ of length B, and the adaptive contribution $\kappa_k - y_k$ lies in $[1 - B, \kappa]$, since $1 \leq \kappa_k \leq \kappa$. The first change has $y_1 = 1$ whatever its direction, since a single update has coherence 1 and $w_0 = 1$, so $\bar{y}_1 = 1$ and the first change carries no evidence about the direction.

The score of a round that changed nothing. After the warm-up such a round is a change of magnitude zero. It enters the smoothed statistic at the value 1 of a path with no common direction, so that $\bar{y} - 1$ is multiplied by λ on every such round, and it scores $\kappa - \min\{\bar{y}, \kappa\}$. Since $\bar{y}_1 = 1$ and every later revision keeps $\bar{y} \geq \lambda\,\bar{y} > 0$, the score lies in $[0, \kappa)$. It is zero while the recent changes still show a direction at or above κ, it is $\kappa - 1$ when $\bar{y} = 1$, and it exceeds $\kappa - 1$ when the recent changes canceled more than chance, so that $\bar{y} < 1$. Let $\bar{y}_0$ denote the smoothed statistic at the last change and count the unchanged rounds after it as $j = 1, 2, \ldots$, so that $\bar{y}_j = 1 + \lambda^j(\bar{y}_0 - 1)$. If $\bar{y}_0 \geq 1$ then $\bar{y}_j \geq 1$ and the score of round j is $\kappa - \min\{\bar{y}_j, \kappa\} \geq \kappa - \bar{y}_j = \kappa - 1 - \lambda^j(\bar{y}_0 - 1)$, and if $\bar{y}_0 < 1$ then $\bar{y}_j < 1$ and the score is $\kappa - \bar{y}_j > \kappa - 1$. In both cases the score of the j-th unchanged round is at least $\kappa - 1 - \lambda^j(\bar{y}_0 - 1)^+$. Summing over $j = 1, \ldots, n$ and using $\sum_{j=1}^{n} \lambda^j \leq \lambda/(1-\lambda)$,

$$\sum_{j=1}^{n} \big(\kappa - \min\{\bar{y}_j, \kappa\}\big) \;\geq\; n\,(\kappa - 1) - (\bar{y}_0 - 1)^+ \,\frac{\lambda}{1-\lambda}.$$

The sum Λ_t of (3) is at least the sum without the floor of its contributions from any round on, since the floor at zero can only raise it, so a loop that has stopped changing after its warm-up reaches Λ^* within $\lceil (\Lambda^* + (\bar{y}_0 - 1)^+\lambda/(1-\lambda))/(\kappa - 1) \rceil$ unchanged rounds. At the setting used, where $\kappa = 3$, $\Lambda^* = 5$, $\lambda = 2/3$ and $B = 5$, that is $\lceil 5/2 \rceil = 3$ rounds from $\bar{y}_0 \leq 1$ and at most $\lceil (5+8)/2 \rceil = 7$ rounds from any $\bar{y}_0 \leq B$, since $(B-1)\lambda/(1-\lambda) = 8$. The fade from $\bar{y}_0 = B$ to below κ takes $\lceil \log((B-1)/(\kappa-1))/\log(1/\lambda) \rceil = 2$ rounds there, since $1 + 4\,(2/3)^2 = 25/9 < 3$.

The mean of y_k under a common direction and the threshold $r_k^*(\kappa')$. Dividing the equality of Theorem 2 by w_{k-1}, the mean of y_k under the mean-ratio approximation is $\mathbb{E}[y_k] = (r + w_{k-1})/(w_{k-1}(r+1))$. Differentiating in r, and omitting the index on w for the moment,

$$\frac{d}{dr}\,\frac{r+w}{w\,(r+1)} = \frac{1}{w}\cdot\frac{(r+1)-(r+w)}{(r+1)^2} = \frac{1-w}{w\,(r+1)^2} > 0,$$

since $w < 1$ after the first update. So $\mathbb{E}[y_k]$ increases strictly in r, from the value 1 at $r = 0$ to the limit $1/w$ as r grows. Let κ' be a reference value held fixed, with $1 \leq \kappa' \leq \kappa$. Multiplying the inequality $\mathbb{E}[y_k] > \kappa'$ by the positive factor $w(r+1)$ and collecting the terms in r,

$$r + w \;>\; \kappa'\,w\,r + \kappa'\,w \qquad\Longleftrightarrow\qquad r\,\big(1 - \kappa'\,w\big) \;>\; w\,(\kappa' - 1).$$

The right side is non-negative. If $\kappa'\,w \geq 1$ the left side is at most zero and the inequality holds at no $r \geq 0$. If $\kappa'\,w < 1$ then dividing by $1 - \kappa' w > 0$ makes the inequality $r > r_k^*(\kappa')$ with

$$r_k^*(\kappa') \;=\; \frac{w_{k-1}\,(\kappa' - 1)}{1 - \kappa'\,w_{k-1}}. \tag{F.1}$$

At $\kappa' = 1$ the right side is zero and $r_k^*(1) = 0$, so every common direction keeps the sum from accumulating. The threshold is stated for a reference value held fixed because the adaptive reference value κ_k of Section 3.3 is computed from the demonstrated peak, which includes the smoothed value of the current change. Hence κ_k and y_k are not independent, and the mean of $\kappa_k - y_k$ is not a fixed value minus the mean above. The adaptive reference value lies between the fixed values 1 and κ, which is what Theorem 3 uses. Along a window of consecutive changes $k_1, k_1 + 1, \ldots$ the settled coherence w_{k-1} decreases in k by Theorem 2, and the mean $(r/w + 1)/(r+1)$ decreases in w at a fixed r, so the mean of y_k increases along the window and is smallest at its first change k_1. Likewise $r_k^*(\kappa')$ increases in w_{k-1} at a fixed κ', since its derivative in w is $(\kappa' - 1)/(1 - \kappa' w)^2 \geq 0$, so the threshold is largest at k_1 and decreases along the window. That is why δ in Theorem 3(i) may be taken as $\mathbb{E}[y_{k_1}] - \kappa$, which is positive exactly when $r > r_{k_1}^*(\kappa)$.

The algorithm over a family of views. Algorithm F.1 is Algorithm 1 run over K candidate views at once, where a view is defined in Appendix E, with two checks deciding which views enter the round's minimum. Each view carries its own number of changes, coherence, smoothed statistic and demonstrated peak, formed as in Algorithm 1. Both checks are run at every round t on the rounds up

to t, so the set of admitted views is a function of the observations up to t, as Assumption 1 requires. The first check drops a view while it has made at least τ changes without ever lowering the quantity of a part, and admits it again the first time it removes a part. The second check drops a view once the loop has rejected at least n_0 proposals and accepted at least one on which the view changed, and the share φ of the view's update magnitudes that the accept decision accounts for, defined below, is below $1/2$. If every view is dropped, all views are kept. A round on which some admitted view changed contributes the smallest $\kappa_k - y_k$ over those views past their warm-up. A round on which no admitted view changed contributes $\kappa - \min\{\max_j \bar{y}^{(j)}, \kappa\}$ once every admitted view is past its warm-up, with each view's smoothed statistic faded toward 1 as in Algorithm 1. Before that, the pre-warm-up pace of Algorithm 1 applies, with the gap counted between rounds on which some admitted view changed. There is no separate procedure on the accept decisions.

The statistic that the second view check thresholds. For the updates u of a view, let

$$\varphi \;=\; 1 - \frac{\text{mean of } \|u\|^2 \text{ over the rejected rounds}}{\text{mean of } \|u\|^2 \text{ over the accepted rounds}}.$$

The view is dropped when $\varphi < \frac{1}{2}$, which is the midpoint of its range. The denominator is positive once an accepted round has changed the view, and until then the check does not apply and leaves the view admitted, as on a loop that rejects its first n_0 proposals. The statistic is 1 when a rejected round leaves the view unchanged, as happens for the artifact that the loop delivers. It is zero when the view changes by as much on a rejected round as on an accepted round, as a proposal changes.

Algorithm F.1: LOOPSTOP over a family of views. It holds a vector and 6 scalars per view.

given: a memory τ, a tolerated delay D, a minimum count n_0 of rejected rounds, and candidate views $1, \dots, K$ over everything the loop leaves, namely the artifact, the transcript, tool output, and scratch files

λ, κ, ρ and Λ^* from τ and D as in Algorithm 1;

$\Lambda \leftarrow 0$, $G \leftarrow 0$, $r \leftarrow 0$, and $M^{(j)} \leftarrow 0$, $V^{(j)} \leftarrow 0$, $z^{(j)} \leftarrow 0$, $k^{(j)} \leftarrow 0$, $y^{*(j)} \leftarrow 1$, $\bar{y}^{(j)} \leftarrow 1$ for each view j;

for *each round* $t = 1, 2, \dots$ **do**

  take the artifact of every view that the round leaves, and $u^{(j)} \leftarrow$ the update of (1) on view j;

  for *each view* j *with* $u^{(j)} \neq 0$ **do**

    $k^{(j)} \leftarrow k^{(j)} + 1, \quad M^{(j)} \leftarrow \lambda M^{(j)} + u^{(j)}, \quad V^{(j)} \leftarrow \lambda V^{(j)} + \|u^{(j)}\|^2,$
    $z^{(j)} \leftarrow \lambda z^{(j)} + 1;$

    $y^{(j)} \leftarrow \|M^{(j)}\|^2 / (z^{(j)} V^{(j)} w_{k^{(j)}-1}), \quad \bar{y}^{(j)} \leftarrow y^{(j)}$ if $k^{(j)} = 1$, else
    $\bar{y}^{(j)} \leftarrow \lambda \bar{y}^{(j)} + (1-\lambda) y^{(j)}, \quad y^{*(j)} \leftarrow \max(y^{*(j)}, \bar{y}^{(j)})$ once $k^{(j)} \geq \tau$;

  $J \leftarrow$ the views that neither check drops on the rounds up to t, where a view is dropped while $k^{(j)} \geq \tau$ and it has not lowered the quantity of any part, and once the loop has rejected n_0 rounds and accepted a round on which the view changed and its share $\varphi^{(j)} < \frac{1}{2}$, and if every view is dropped, $J \leftarrow$ all views;

  if *some view* $j \in J$ *has* $u^{(j)} \neq 0$ **then**

    if *some such view has* $k^{(j)} > \tau$ **then**

      $\Lambda \leftarrow \max\big(0,\ \Lambda + \min_{j \in J,\, u^{(j)} \neq 0,\, k^{(j)} > \tau} \big(\min\{\kappa, 1 + \rho\,(y^{*(j)} - 1)\} - y^{(j)}\big)\big)$, the reference value κ_k of Section 3.3;

    else $G \leftarrow \max(G, t - r), \quad r \leftarrow t, \quad \Lambda \leftarrow 0;$

  else

    if *every* $j \in J$ *has* $k^{(j)} \geq \tau$ **then** $\bar{y}^{(j)} \leftarrow \lambda\, \bar{y}^{(j)} + (1-\lambda)$ for $j \in J$,
    $\Lambda \leftarrow \max\big(0,\ \Lambda + \kappa - \min\{\max_{j \in J} \bar{y}^{(j)}, \kappa\}\big);$

    else $\Lambda \leftarrow \Lambda + 1/(2\max\{G, D\});$

  if $\Lambda \geq \Lambda^*$ **then**

    return stop, and deliver the artifact of round $t + 1$;

The cost of the two checks when they are wrong. The first check can be wrong, and its cost is at most the delay that the retained views add, since the algorithm keeps every view if all views are dropped. An artifact that has made only additions up to a round cannot be distinguished from a view that removes nothing, so it is dropped until its first removal, and the mistake delays admission without excluding the view. A loop's transcript is appended and not revised, so the first check drops

it at its τ-th change and does not admit it again. The list of removed parts that (1) forms tells the first check whether some part's quantity decreased, so the check costs nothing beyond the update.

G How the Constants of LoopStop Follow from Its Two Settings

Algorithm 1 has two settings, and five constants follow from them. The memory τ is a number of changes and the tolerated delay D a number of rounds. The variance of the decay-weighted average fixes the decay factor, solving (F.1) at $r^* = 1$ fixes the reference value, and the delay bound of Theorem 3(ii) at D for a peak of $1+(\kappa-1)/\rho$ or above fixes the threshold, so all three are functions of the two settings. The slope is fixed by the convention $\rho = \tau/2$, which no derivation fixes and which places the cap at a demonstrated peak of $2 - 1/\tau$, and the warm-up is the round of the τ-th change,

$$\kappa = \frac{\tau+1}{2}, \qquad \rho = \frac{\tau}{2}, \qquad \Lambda^* = \frac{(\tau-1)\,D - (\tau+1)}{2}, \qquad t_0 = \text{the round of the } \tau\text{-th change}, \tag{G.1}$$

which at $\tau = 5$ and $D = 4$ are $\kappa = 3$, $\rho = 5/2$ and $\Lambda^* = 5$, with $\lambda = (\tau-1)/(\tau+1) = 2/3$ and a settled coherence $w = 1/\tau$. A loop whose demonstrated peak p lies below $1 + (\kappa-1)/\rho$ has $D(\kappa-1)/(\kappa(p)-1)$ rounds of delay instead, which grows without bound as p decreases to 1. Nothing in (G.1) depends on the task, the artifact kind, or the quality, which is why a single setting serves every family, and Appendix I.3 gives the values used and what changing them costs.

The memory is the number of equally weighted updates with the variance of the decay-weighted average. The weights λ^j in M_k and V_k sum to $1/(1-\lambda)$, and the variance of the average that they form equals that of an unweighted average over as many updates as the squared sum of the weights divided by the sum of their squares. To see this, let the updates be independent with a common variance v per coordinate and let $g_j = \lambda^j / \sum_i \lambda^i$ be the normalized weights, which sum to 1. The weighted average $\sum_j g_j u_j$ then has variance $\sum_j g_j^2\, v$ per coordinate, and an unweighted average of n updates has variance v/n, so the two agree at $n = 1/\sum_j g_j^2 = (\sum_j \lambda^j)^2 / \sum_j \lambda^{2j}$. Summing the two geometric series,

$$\frac{\left(\sum_{j\ge 0} \lambda^j\right)^2}{\sum_{j\ge 0} \lambda^{2j}} \;=\; \frac{(1-\lambda)^{-2}}{(1-\lambda^2)^{-1}} \;=\; \frac{(1-\lambda)(1+\lambda)}{(1-\lambda)^2} \;=\; \frac{1+\lambda}{1-\lambda} \;=\; \tau,$$

and solving for the decay factor gives $\lambda = (\tau-1)/(\tau+1)$. The settled coherence $w = (1-\lambda)/(1+\lambda)$ is then $1/\tau$, and the range $B = 1/w$ of y_k is τ itself. A short memory lets the statistic follow a change in the common direction within a few changes and makes y_k noisy, since it averages few updates. A long memory has the opposite effect and delays the stop by about the memory, since the updates taken before settling stay in the average. Directional evidence needs at least two changes in the window, so the memory should cover a few changes and no more. At $\tau = 5$ the range of y_k is $[0, 5]$ instead of the range $[0, 2]$ that a memory of two would give. The reference value $\kappa = (\tau+1)/2$ of a mean update as large as the noise then lies two units above the value 1 of a path with no common direction instead of half a unit, which leaves a margin between the two for the noise of the statistic.

The reference value is the mean of y_k at the smallest signal-to-noise ratio to be detected. Let r^* denote that ratio. Multiplying (F.1) at $\kappa_k = \kappa$ by $1 - \kappa w$ and collecting the terms in κ,

$$r^* - \kappa\, w\, r^* \;=\; \kappa\, w - w \qquad \Longleftrightarrow \qquad r^* + w \;=\; \kappa\, w\, (r^* + 1),$$

and dividing by $w(r^*+1) > 0$ gives

$$\kappa \;=\; \frac{r^* + w}{w\,(r^*+1)}. \tag{G.2}$$

By Theorem 2 that value is the mean of y_k in the limit of many changes at the signal-to-noise ratio r^*, so a loop tested at κ accumulates evidence exactly while its signal-to-noise ratio lies below r^*. The value increases in r^* from one at $r^* = 0$ towards B as r^* grows, so every choice of r^* gives a κ with $\kappa w < 1$. The choice $r^* = 1$ makes LoopStop accumulate evidence once the mean update of a change is smaller than its noise. Substituting $r^* = 1$ and $w = 1/\tau$,

$$\kappa \;=\; \frac{1 + 1/\tau}{(1/\tau)\cdot 2} \;=\; \frac{(\tau+1)/\tau}{2/\tau} \;=\; \frac{\tau+1}{2},$$

which is the first part of (G.1) and equals 3 at $\tau = 5$. The constraint $\kappa w < 1$ then holds for every memory $\tau > 1$, since $\kappa w = (\tau+1)/(2\tau)$, and it holds for the adaptive reference value as well because $\kappa_k \le \kappa$.

The slope is a convention for how much of a demonstrated peak counts as a direction. The reference value κ_k of Section 3.3 runs from 1 at a demonstrated peak of 1 to κ at a peak of $1 + (\kappa -$

$1)/\rho$, and it is κ at every larger peak. Solving for the peak at which the cap takes over,

$$1 + \rho\,(y^* - 1) \;=\; \kappa \qquad \Longleftrightarrow \qquad y^* \;=\; 1 + \frac{\kappa - 1}{\rho},$$

which at $\kappa = 3$ and $\rho = 5/2$ is $9/5$. That lies two fifths of the distance from a path with no common direction, where $y^* = 1$, to a path whose mean update is as large as its noise, where $y^* = \kappa = 3$. A loop is therefore tested for a mean update below its noise, $r < 1$, once it has shown two fifths of that much direction, and for a smaller mean update below it. Neither (F.1) nor (G.1) determines ρ, and the convention $\rho = \tau/2$ is what places the cap at the peak $2 - 1/\tau$. A larger ρ reaches the cap sooner and recovers the value κ on every loop that ever showed any direction, and $\rho = 0$ holds every loop at $\kappa_k = 1$.

The threshold is set from the delay tolerated after settling. Theorem 3(ii) bounds the expected number of rounds from settling to the stop by $(\Lambda^* + \kappa)/(\kappa(p) - 1)$, at a demonstrated peak of p or above, for a loop whose smoothed statistic has faded to 1 and stays at or below 1. For any loop at the cap the bound is at most $(\Lambda^* + \kappa + \lambda(B - 1)/(1 - \lambda))/(\kappa - 1 - \lambda) = 12$ rounds at the setting used, since $\bar{y} \le B$ and since ε, which bounds the conditional mean of $(y_k - 1)^+$ in the proof of Theorem 3(ii), is at most 1 under Assumption 3. The last inequality holds because for $y \ge 0$ the positive part obeys $(y - 1)^+ = (y - 1) + (1 - y)^+ \le (y - 1) + 1 = y$, so that $\mathbb{E}[(y_k - 1)^+] \le \mathbb{E}[y_k] \le 1$. Setting the first bound, which is the bound for a loop whose statistic has faded to 1 and stays there, equal to the delay D for a loop that has reached the cap, where $\kappa(p) = \kappa$, and solving for the threshold,

$$\frac{\Lambda^* + \kappa}{\kappa - 1} = D \qquad \Longleftrightarrow \qquad \Lambda^* = (\kappa - 1)\,D - \kappa,$$

and substituting $\kappa = (\tau + 1)/2$,

$$\Lambda^* \;=\; \frac{\tau - 1}{2}\,D - \frac{\tau + 1}{2} \;=\; \frac{(\tau - 1)\,D - (\tau + 1)}{2},$$

which is the third part of (G.1). It is positive for $D > (\tau + 1)/(\tau - 1)$, which is $D > 3/2$ at $\tau = 5$, and it equals $2D - 3$ there, so $D = 4$ gives $\Lambda^* = 5$. That is the smallest delay at which no single round can reach the threshold, since $\Lambda^* = 5 > \kappa = 3$. With $D = 3$ the threshold would be $\Lambda^* = 3 = \kappa$, so a single changing round with $y_k = 0$ at the cap, which contributes exactly κ, would stop the loop by itself. A budget stated as a cost becomes D after division by the cost of a round.

The same threshold on a loop that has demonstrated less. Substituting the threshold back into the bound of Theorem 3(ii) at a demonstrated peak of p, under the same two cases,

$$\frac{\Lambda^* + \kappa}{\kappa(p) - 1} \;=\; \frac{(\kappa - 1)\,D}{\min\{\kappa,\, 1 + \rho\,(p - 1)\} - 1} \;=\; \begin{cases} D & p \ge 1 + (\kappa - 1)/\rho, \\ \dfrac{(\kappa - 1)\,D}{\rho\,(p - 1)} & 1 < p < 1 + (\kappa - 1)/\rho, \end{cases} \tag{G.3}$$

which grows as $1/(p - 1)$ and exceeds every finite value as p decreases to 1. At $p = 1$ no bound exists, since there $\kappa_k = 1$ makes the mean contribution $1 - \mathbb{E}[y_k]$, which Assumption 3 allows to be zero. The guarantee that remains at $p = 1$ is part (i) of Theorem 3, which holds at every peak. It holds because $\kappa_k \le \kappa$ pointwise, so raising the reference value from κ_k to κ can only make the algorithm stop sooner. Such a loop is stopped in practice by a measured $y_k < 1$ instead of $y_k = 1$, and by its unchanged rounds after the warm-up. Each unchanged round contributes $\kappa - \min\{\bar{y}, \kappa\}$, which is $\kappa - 1$ once the direction has faded, so at most $\lceil (\Lambda^* + (\bar{y} - 1)^+\lambda/(1 - \lambda))/(\kappa - 1) \rceil$ of them suffice from a smoothed statistic of $\bar{y}$, as Appendix F derives, however little direction the loop ever showed. Before the warm-up the unchanged rounds are scored at the loop's own pace, as the next paragraph states, so a loop that stops after fewer than τ changes is stopped as well.

The warm-up is the τ-th change. Before the τ-th change the averages M_k and V_k hold fewer updates than the memory, so a single large first update dominates y_k and LOOPSTOP would judge the loop on its first change. Let t_0 be the round of the τ-th change. The demonstrated peak and the unchanged rounds count from t_0 on, and the changing rounds count from the next change on. The first contribution of a change then averages as many changes as every later change, and no loop is stopped before its statistic has been formed at the whole memory. A loop that accepts fewer than τ changes over its horizon is therefore not stopped by its changes. The visible-tests scenarios of BigCodeBench-Hard are such loops, and the rounds that changed nothing carry the evidence there, at the pace of the next paragraph.

Before the warm-up the loop's own gaps set the pace. Let G_k denote the longest of the k gaps in rounds between consecutive changes up to and including the k-th change, the first gap counted from round 0, with $G_0 = 0$, and let $R_k = 2\Lambda^* \max\{G_k, D\}$. Before the τ-th change, an unchanged round adds $1/(2\max\{G_k, D\}) = \Lambda^*/R_k$ to the sum, so the sum reaches Λ^* on the R_k-th rejected round since the last change. A change before or at the warm-up resets the sum to zero, since a change is evidence against settling and since the sum from t_0 on is the sum of the previous paragraphs. The floor is D because the algorithm already tolerates D rounds without a change after settling, so a shorter gap says nothing about how rarely the loop accepts. The run is $2\Lambda^*$ gaps for two reasons. At a gap of g rounds about g rejected rounds pass per change, so a rejected round carries about $1/g$ of the evidence of a change against a further change, and Λ^* gaps match the threshold. The longest of at most $\tau - 1$ gaps is an estimate that the run has to exceed by a margin, which the factor two supplies. Theorem G.1 states the two consequences. A loop that stops changing after $k < \tau$ changes is stopped R_k rounds after its last change, and a loop whose evaluator keeps accepting at a constant rate ν is stopped in this manner before its warm-up with probability at most $(1-\nu)^{40} + 0.14$ at the setting used. That probability is 0.15 at the rate of 1 proposal in 10 that the strict evaluators of the experiments accept.

Theorem G.1 (the stop before the warm-up)**.** *Let G_k be the longest of the k gaps in rounds between consecutive changes up to and including the k-th change, the first gap counted from round 0 and $G_0 = 0$, and let $R_k = 2\Lambda^* \max\{G_k, D\}$.*

(i) A loop whose last change is its k-th, at round r_k, with $k < \tau$, is stopped at round $r_k + R_k$ at the latest, and a loop that does not change is stopped at round $2\Lambda^* D$ at the latest.

(ii) If every accepted proposal changes the artifact, and the accept decisions $A_1, A_2, \ldots$ of the rounds after round 0 are independent with $\mathbb{P}(A_t = 1) = \nu$ and independent of the statistics, the probability that the loop is stopped before its τ-th change is at most $(1-\nu)^{2\Lambda^* D} + \sum_{k=1}^{\tau-1} \big(k/(2\Lambda^* + k)\big)^k$, and at most $\tau\,(1-\nu)^{2\Lambda^* D}$.

Proof. Part (i). After the change at round r_k the sum is zero. Every later round changes nothing and adds $1/(2\max\{G_k, D\}) = \Lambda^*/R_k$, and R_k does not change since no change arrives, so after R_k such rounds the sum is Λ^* and the algorithm stops the loop at round $r_k + R_k$, unless it stopped earlier. A loop that does not change has $G_0 = 0$ and $R_0 = 2\Lambda^* D$, and the same count from round 0 gives the second claim.

Part (ii). Let $g_1, g_2, \ldots$ be the gaps between consecutive changes, each counted in rounds up to and including the next change, with g_1 counted from round 0. Under the hypothesis each gap is the number of independent accept decisions at rate ν up to and including the first acceptance, so the gaps are independent and each gap has the geometric distribution starting at 1, with $\mathbb{P}(g > m) = (1-\nu)^m$ for every integer $m \geq 0$ and $\mathbb{P}(g = m) = \nu(1-\nu)^{m-1}$ for every $m \geq 1$. The loop is stopped before its τ-th change only if $g_1 > R_0$ or if $g_{k+1} > R_k$ for some k from 1 to $\tau - 1$, with strict inequalities because the R_k-th unchanged round after the k-th change is itself not a change, so a stop there means that the next change came more than R_k rounds after the k-th. Here R_k is a function of $g_1, \ldots, g_k$ alone. The probability of a union is at most the sum of the probabilities, so

$$\mathbb{P}(\text{stopped before the } \tau\text{-th change}) \;\leq\; \mathbb{P}(g_1 > R_0) + \sum_{k=1}^{\tau-1} \mathbb{P}(g_{k+1} > R_k).$$

The first term is $(1-\nu)^{2\Lambda^* D}$. For the k-th term, conditioning on $g_1, \ldots, g_k$ and using the independence of g_{k+1}, $\mathbb{P}(g_{k+1} > R_k \mid g_1, \ldots, g_k) = (1-\nu)^{R_k}$. Since $R_k \geq 2\Lambda^* D$ on every path, this is at most $(1-\nu)^{2\Lambda^* D}$, which gives the second bound after adding the first term. For the first bound, the longest of k gaps is at least their mean, so $R_k \geq 2\Lambda^* G_k \geq (2\Lambda^*/k)\,(g_1 + \cdots + g_k)$, and with $x = 1 - \nu$ and $a = 2\Lambda^*/k$,

$$\mathbb{E}\big[(1-\nu)^{R_k}\big] \;\leq\; \mathbb{E}\Big[\prod_{i=1}^{k} x^{a g_i}\Big] \;=\; \prod_{i=1}^{k} \mathbb{E}\big[x^{a g_i}\big],$$

by the independence of the gaps. Summing the geometric series, for a gap g with $\mathbb{P}(g = m) = (1-x)\,x^{m-1}$ at every $m \geq 1$,

$$\mathbb{E}\big[x^{ag}\big] \;=\; \sum_{m \geq 1} (1-x)\,x^{m-1}\,x^{am} \;=\; \frac{(1-x)\,x^a}{1 - x^{a+1}} \;\leq\; \frac{1}{1+a},$$

where the last step is the inequality $1 - x^{a+1} \geq (a+1)\,(1-x)\,x^a$ for $x \in (0,1]$ and $a \geq 0$, which holds since $f(x) = 1 - x^{a+1} - (a+1)(1-x)\,x^a$ has $f(1) = 0$ and $f'(x) = -(a+1)\,a\,(1-x)\,x^{a-1} \leq 0$, so that $f \geq 0$ on $(0,1]$. With $1/(1+a) = k/(2\Lambda^* + k)$ the k-th term is at most $(k/(2\Lambda^* + k))^k$, which gives the first bound. At the setting used, where $\Lambda^* = 5$ and $\tau = 5$, the four terms are $1/11 + (2/12)^2 + (3/13)^3 + (4/14)^4 = 0.091 + 0.028 + 0.012 + 0.007 = 0.14$, and the first term is $0.9^{40} = 0.015$ at $\nu = 1/10$. □

H PROMPT TEMPLATES

Every model call of every family uses the templates below, with thinking off on Qwen3.5, the system message `Reply with JSON only.` except where a template says otherwise, and the model's own sampling settings. Angle brackets mark what the harness fills in. The reference solution, the hidden tests, the simulator's cost and the optimal gains appear in no prompt, and a check on every prompt refuses the words by which the harness knows them. The simulation-based evaluators of the two controller families, the visible-tests evaluator of BigCodeBench-Hard and the validation check of MLAgentBench-Kaggle are programs, described in Table I.2, and make no model call. The MLAgentBench-Kaggle roles are called through the Claude command-line interface with no tools, a thinking budget of 1,024 tokens, and the same system messages. The lines of every template are wrapped to the page width, so a line break inside a sentence belongs to the page and not to the string that the model receives.

H.1 CRUISE CONTROL

The scenario, shared by every role:

```
You are tuning the cruise control on a car.

The driver has set it to 100 km/h. The road turns uphill, the car slows
to
95 km/h, and the controller has to bring it back to 100 and hold it
there for
the
rest of the climb.

Ten times a second the controller looks at two things:

  * the speed error: how far the speed is from 100 km/h right now, in
  km/h
  * the accumulated speed error: the running total of that error since
  the
  climb
    began, which grows the longer the car stays off the set speed

and asks for an acceleration, in m/s^2:

    acceleration = -(k1 * speed_error + k2 * accumulated_speed_error)

k1 answers the gap right now. Raise it and the car pulls back toward 100
km/h
harder and sooner.

k2 answers how long the car has been off the set speed. Without it the
car
would
sit a little under 100 km/h for the whole hill without arriving. Raise
it
and that leftover gap closes faster.

Set them too low and the car crawls back, spending most of the climb
below the
set speed. Set them too high and it shoots past 100 km/h, dips below,
shoots
past
again, and beyond a point those swings grow instead of dying away. The
accelerations asked for also matter: a setting that recovers quickly by
demanding
very hard acceleration is uncomfortable to ride in.
```

The generator, with no evaluator or under a model evaluator:

```
<scenario>

Current setting: k1 = <k1>, k2 = <k2>

Propose one revised setting.
Answer with exactly {"k1": <number>, "k2": <number>}
```

The generator under an inspector, which carries out the inspector's sentence within a band of a factor of two on each gain:

```
<scenario>

Current setting: k1 = <k1>, k2 = <k2>

A reviewer has looked at this setting and says:
  <the inspector's sentence>

Carry that out. Propose the revised setting.
One step at a time: you may not multiply or divide either gain by more
than 2 in this step, so k1 must stay between <k1/2> and <2 k1>, and k2
between <k2/2> and <2 k2>. Choose where inside that range the reviewer's
wording puts you.
Answer with exactly {"k1": <number>, "k2": <number>}
```

The model evaluator, shown both settings and their simulated responses:

```
<scenario>

Current setting:  k1 = <k1>,  k2 = <k2>
Proposed setting: k1 = <k1'>,  k2 = <k2'>

Simulated response to a 5 km/h speed drop, computed independently:
  current setting:  settles in <s> s, overshoots by <o> km/h,
                    peak acceleration <a> m/s^2
  proposed setting: settles in <s'> s, overshoots by <o'> km/h,
                    peak acceleration <a'> m/s^2

Accept the proposed setting only if it would hold the set speed better
overall, counting both how quickly speed error is removed and how much
acceleration is used.
Answer with exactly {"accept": true} or {"accept": false}
```

The inspector, which names a gain, a direction and a magnitude word chosen from three, from which the harness forms the sentence that the generator receives (for example `Increase k2 moderately.`):

```
<scenario>

Current setting: k1 = <k1>, k2 = <k2>

You inspect this setting for the engineer who will change it. Name the
one gain most in need of change, which way to move it, and how far. Do
not give a number.
Answer with exactly {"gain": "k1" or "k2", "direction": "increase" or
"decrease", "magnitude": "slightly" or "moderately" or "substantially"}
```

H.2 CART-POLE

The scenario, shared by every role:

```
You are tuning the controller that balances a pole upright on a
motorized
cart.

The pole is hinged on top of the cart and falls over unless the cart
```

```
moves
under
it. The cart runs on a straight track. When the pole is knocked a few
degrees
off upright, the controller has to bring the pole back to vertical and
the
cart
back to the middle of the track, and hold both there.

Twenty times a second the controller looks at four things:

  * the cart's position: how far it sits from the middle of the track,
  in
  metres
  * the cart's speed, in metres per second
  * the pole's tilt: how far it leans from upright, in radians
  * the pole's tilt rate: how fast the lean is changing, in radians per
  second

and asks for a force on the cart, in newtons:

    force = -(k1 * position + k2 * speed + k3 * tilt + k4 * tilt_rate)

k3 and k4 answer the pole. Raise them and the cart moves under a leaning
pole
harder and sooner. k1 and k2 answer the cart. Because the cart has to
move
away
from the middle first to catch the pole, a controller of this kind has
k1 and
k2
negative and k3 and k4 positive, and the size of each gain is what
tuning
sets.

Set the gains too low and the pole falls. Set some too high and the cart
overshoots and the pole swings from side to side, and beyond a point
those
swings
grow instead of dying away. The forces asked for also matter: a setting
that
recovers quickly by demanding a very hard push is hard on the motor.
```

The generator, the model evaluator and the inspector follow the cruise-control templates with four gains in place of two and with the band of a factor of two applied to the size of each gain. The simulated response to a five-degree knock of the pole is reported as its recovery time, cart travel and peak force, and the acceptance criterion is *"only if it would balance the pole and hold the cart in the middle better overall, counting how quickly the system recovers, how far the cart travels, and how much force is used"*. The inspector is also shown the simulated response of the current setting, and its answer names a single gain of the four, a direction and a magnitude word, rendered as `<Direction> the size of <gain> <magnitude>.`

H.3 ARC-AGI

The scenario, shared by every role, is followed by the task's training pairs, each rendered one row per line with the digits of a row separated by spaces:

```
You are solving a visual puzzle on a grid of colored cells, where each
cell
holds a digit from 0 to 9 and each digit is a color. The puzzle shows a
few
training examples, each an input grid and the output grid that the
hidden rule
produces from it. The same rule turns the test input into the test
```

```
output.
Your
job is to work out the rule from the examples and produce the output
grid for
the test input. Grids are written one row per line, the digits of a row
separated by spaces.

Example 1 input:
<grid>
Example 1 output:
<grid>
...

Test input:
<grid>
```

The generator asks `Produce the output grid for the test input.` on the first round and, on later rounds,

```
Your current output grid:
<grid>

Produce one revised output grid.
Answer with exactly {"grid": [[<digits of row 1>], [<digits of row 2>],
...]}, every row a list of integers 0 to 9.
```

Under an inspector the middle line becomes `A reviewer has looked at this grid and says: <sentence>. Carry that out. Produce the revised output grid.`

The model evaluator:

```
Grid A, currently kept:
<grid>

Grid B, proposed replacement:
<grid>

Replace A with B only if B is closer to the output that the rule of the
examples
produces from the test input.
Answer with exactly {"accept": true} or {"accept": false}
```

The inspector:

```
Current output grid:
<grid>

You inspect this grid for the solver who will change it. Name the one
region that looks most wrong, by its rows and columns counted from 1,
and say in one short sentence what is wrong there. Do not give the
corrected cells.
Answer with exactly {"rows": "<r1>-<r2>", "cols": "<c1>-<c2>",
"problem": "<one sentence>"}
```

The sentence passed on is `Rows <r1>-<r2>, columns <c1>-<c2>: <problem>.`

H.4 BIGCODEBENCH-HARD

The generator's system message is `You are a Python programmer. Reply with code only, in one ```python block.` Its user message opens with the task's instruction from BigCodeBench-Hard and, on the first round, asks `Write the complete solution.` On later rounds:

```
<task instruction>
```

```
Current solution:
```python
<the file the loop holds>
```

Result of the checks you can see: <passed> of <total> passed.
  FAILED <test name>: <message>
  ...

Revise the solution to fix what failed.
Return the entire file, including imports and the function definition.
```

The checks shown are the visible two fifths of the task's tests and not the hidden ones. Under an inspector the request becomes `A reviewer says: <sentence>. Apply that and return the complete revised solution.`

The model evaluator and the inspector use the JSON system message:

```
<task instruction>

Version A (currently kept):
```python
<file>
```

Version B (proposed replacement):
```python
<file>
```

Replace A with B only if B is a better solution to the task above.
Answer with exactly {"accept": true} or {"accept": false}

<task instruction>

Current solution:
```python
<file>
```

Name the single most important thing to change about this solution so
it better solves the task. One sentence, no code.
Answer with exactly {"change": "<one sentence>"}
```

H.5 MLAgentBench-Kaggle

The generator's system message is that of BigCodeBench-Hard. Its user message opens with the task's brief, which holds the benchmark's summary and instructions for the task, its data description, and what the script must print and save, followed by the rule of the editable section:

```
Task: <the benchmark's summary of the task>

The benchmark's instructions: <the benchmark's task description>

Data description:
<the benchmark's data description>

The script runs in a folder that holds train.csv and test.csv. It must
print the line '<Validation MAE or Validation Accuracy>: <number>' for
its validation split of train.csv and write submission.csv for every row
of test.csv. Only the standard library, numpy, pandas and scikit-learn
are installed.

The script below is fixed except for the editable section between the
```

```
markers. Only that section can change. It runs after the rows of
train.csv are loaded, and the fixed part after it prints the scores and
writes submission.csv from prepare(test rows) and model.predict.
```

Here `train.csv` holds the visible four fifths of the task's rows and `test.csv` the features of the hidden fifth without their answers. On the first round the message continues with the benchmark's starter script, its editable section holding the benchmark's feature list and the model left for the generator to fill in, and asks `Write the editable section so that it prepares the features, trains a model and sets the printed scores.` On later rounds:

```
Script:
```python
<the fixed scaffold with the section the loop holds between the markers>
```

What it printed when run:
<Validation MAE or Validation Accuracy>: <the printed value>
Last lines printed:
  <the last six lines the script printed>

Improve the editable section so that its validation score gets better,
by changing the features, the model, or its settings.
Return the complete editable section only, in one ```python block,
without the markers.
```

A script that failed is reported as `The script did not run to completion: <error>` in place of its validation line. Under an inspector the request becomes `A reviewer says: <sentence>. Apply that to the editable section.`

The family uses two versions of this message, and the difference is a single block. The four tabular tasks run with the message above, which shows the generator only the script the loop holds and what it printed, so a proposal that the evaluator did not keep leaves no trace in the next message. The two image tasks run with a second version that adds, before the request, what the rejected proposal's own run printed, as the person running the loop would see it before trying again:

```
Your previous proposal was not kept. What it printed when run:
<its validation line, or The script did not run to completion: <error>>
Last lines printed:
  <the last six lines it printed>
```

The block was added for the image tasks because of identify-contrails, whose starter trains a segmentation model and whose first proposals fail on tensor shapes. There the first version repeated itself, since under the validation check the generator, shown nothing of its rejected proposal, returned the same failing section on every round, for all 60 rounds in two of the first loops. The tabular tasks had run before the block existed, and their scripts run in seconds on features and a model, where the first version produced running scripts from the first round. The block changes what the generator sees and therefore the loop's proposals. A loop under the second version accepts more changes under the validation check, since the generator abandons a rejected proposal instead of repeating it. The two versions are therefore reported as a difference between the tabular and the image tasks of the same family, and every rule is scored on the same loops under either version.

The model evaluator uses the JSON system message and receives the brief and the rule followed by:

```
Editable section A, currently kept:
```python
<the section the loop holds>
```
What the script printed with A:
<its validation line and last lines, as above>

Editable section B, proposed replacement:
```python
<the proposed section>
```
What the script printed with B:
```

```
<its validation line and last lines, as above>

Replace A with B only if B is more likely to predict the unseen rows of
test.csv better.
Answer with exactly {"accept": true} or {"accept": false}
```

The inspector receives the brief and the rule followed by:

```
Editable section under review:
'''python
<the section the loop holds>
'''
What the script printed:
<its validation line and last lines, as above>

Check this section. If nothing should change, say so. Otherwise name the
one change most likely to make it predict the unseen rows better, in one
sentence, without giving code.
Answer with exactly {"correct": true} or
{"correct": false, "problem": "<one sentence>"}
```

On feedback the contract asks for a fitted model over the essay texts in place of the feature function. On parkinsons-disease it asks for the feature function over the visit table that the scaffold builds, with a row per visit that has a protein sample, and for a model that predicts the 16 scores. The validation check makes no model call. It keeps a proposal only when the composed script ran to completion and printed a better validation score than the kept script, and it keeps any proposal that ran when the kept script did not.

On the two image tasks the line of the brief that describes the folder is replaced. On fathomnet:

```
The script runs in a folder that holds train.csv (id, categories),
test.csv (id), category_key.csv and the folder images/ with every image
as <id>.png. It must print the line 'Validation MAP@20: <number>' for
its validation split of train.csv and write submission.csv with a row
per test.csv id: id, categories (up to twenty category ids,
space-separated, most likely first) and osd. PyTorch 2.11, torchvision,
timm, numpy, pandas, Pillow and scikit-learn are installed and a GPU is
available.
```

and on identify-contrails:

```
The script runs in a folder that holds train/ and test/, one folder per
record_id with band_11.npy, band_14.npy and band_15.npy (256 x 256 x 8
frames, the labeled frame is index 4) and, under train/ only,
human_pixel_masks.npy (256 x 256 x 1), plus sample_submission.csv. It
must print the line 'Validation Dice Score: <number>' for its validation
split of train/ and write submission.csv with a run-length-encoded mask
per test record. PyTorch 2.11, torchvision, timm, numpy and pandas are
installed and a GPU is available.
```

The rule of the editable section says on fathomnet that the section runs after the images and labels are loaded and split, and that the fixed part after it prints the scores and produces `submission.csv` from `predict_proba` on the test images. On identify-contrails it says that the section runs after the record folders are listed and split into `train_dataset` and `valid_dataset`, and that the fixed part prints the validation Dice score of `model` and produces `submission.csv` from its predictions on the test records. The contract of the section asks on fathomnet for `model`, a PyTorch module trained on the training images and their multi-hot labels, `predict_proba`, which maps images to the probabilities of the 290 categories, and the mean average precision at 20 on the training and validation splits. On identify-contrails it asks for `model`, a module that maps a padded image batch to two-class logits of the same height and width, trained on the training records. The starter of each section is the benchmark's own model in PyTorch, a small convolutional network for fathomnet and a single 1×1 convolution for identify-contrails, with the training left to fill in. The evaluator's and the inspector's questions are the same words on every task, `predict the`

`unseen rows of test.csv better`, and on identify-contrails, whose held-out records are stored under `test/`, the models interpret the phrase as referring to those records.

I EXPERIMENTS IN DETAIL

I.1 THE STANDARD ERROR

Every quantity reported with a plus or minus sign is a mean over the seeds of a family together with the standard error of that mean. A seed is the unit on every family, and the MLAgentBench-Kaggle family, with three seeds, has the coarsest spread. A seed's value is the mean over every trajectory that it holds, so a seed of the cart-pole family contributes the mean over its 10 scenarios, a seed of the ARC-AGI family contributes the mean over its 6 scenarios and 400 tasks, and a seed of the MLAgentBench-Kaggle family contributes the mean over its 4 scenarios and 6 tasks. The rounds of the best budget per scenario carry a standard error on ARC-AGI alone. There a seed's scenarios hold different numbers of loops, since the loops whose first attempt gave no grid are left out, so a seed's mean round varies with those counts. For k seeds with values $v_1, \dots, v_k$ and mean $\bar{v}$,

$$\mathrm{se} \;=\; \sqrt{\frac{1}{k(k-1)}\sum_{i=1}^{k}(v_i-\bar{v})^2} \;=\; \frac{s}{\sqrt{k}}, \qquad s^2 = \frac{1}{k-1}\sum_{i=1}^{k}(v_i-\bar{v})^2. \tag{I.1}$$

The seeds of a family are independent, so the mean of k of them has variance σ^2/k where σ is the standard deviation of a single seed, and (I.1) estimates $\sigma/\sqrt{k}$ by putting the sample standard deviation s in place of σ. The standard error therefore measures how much the reported mean would change if the family were run again with fresh seeds, and it decreases as $1/\sqrt{k}$. The standard deviation s measures something else, namely how far single seeds lie from each other, and it does not decrease as seeds are added. A difference between two rules is formed seed by seed, since both rules stop the same trajectories, and its standard error is computed from those k differences in the same way. A family with few seeds gives a poor estimate of σ, since (I.1) then rests on $k-1$ degrees of freedom. That leaves 2 degrees of freedom on the 3 seeds of the MLAgentBench-Kaggle family, 4 degrees of freedom on the 5 seeds of BigCodeBench-Hard, and 19 degrees of freedom on the 20 seeds of the two controller families and of ARC-AGI.

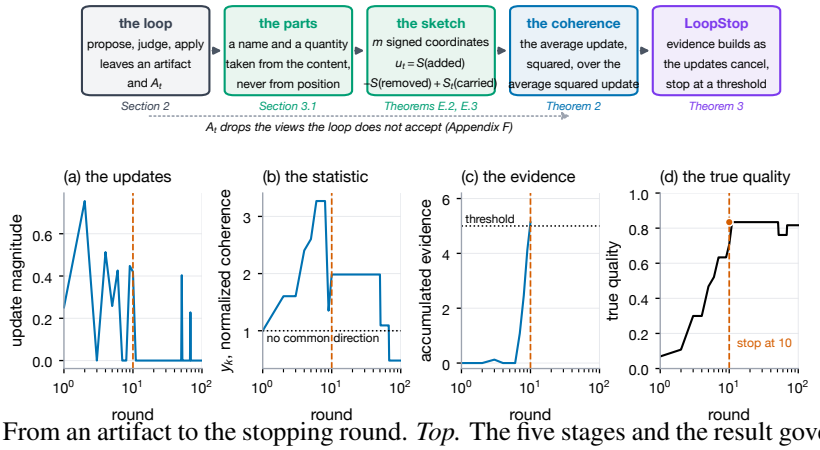

Figure I.1: From an artifact to the stopping round. *Top.* The five stages and the result governing each stage. *Bottom.* A single cruise-control loop, with the update magnitudes, y_k, the sum Λ_t against its threshold, and the true quality, with the round on a logarithmic axis.

I.2 THE LOOPS IN DETAIL

Table I.1 defines the true quality of each family, which the scoring holds out from every role, and Figure I.1 follows a single cruise-control loop from its artifact to the round at which LOOPSTOP stops it. Table I.2 lists every scenario of the five families with the model in every role, the rounds per trajectory, and the seeds per scenario, and Appendix I.18 repeats Figure 2 for every scenario. The cart-pole and ARC-AGI loops ran on a cluster of GH200 nodes with Qwen3.5 served by vLLM, with thinking off and the model's own sampling settings. The two models ran on one or two GPUs of a single node, with every loop of a family running at once.

ARC-AGI. The family runs all 400 tasks of the benchmark's public evaluation set, with no screen on their difficulty, so that the loops whose first grid is already perfect stay in the family, where every rule scores the same on them. A loop whose first attempt gives no grid ends at round 0 and is left

out. The generator produced no usable grid at round zero on 3,166 of the 48,000 loops, which the rows record as a failed first attempt, and those loops carry no trajectory, which leaves the 44,834 loops of Table I.7. The count is highest, 678 of 8,000, on the no-evaluator scenario of the smaller generator.

BigCodeBench-Hard. The family runs every task of the hard split of BigCodeBench whose test suite runs in the sandbox with the benchmark's own solution, 112 of its 148 tasks. The hard split allows improvement without a screen on difficulty, since a quarter of the loops start at a perfect score and a third start at zero, and the loops that start perfect stay in the family, where every rule scores the same on them.

MLAgentBench-Kaggle. The family runs the 6 Kaggle tasks of that benchmark, namely two classic tasks, house-price and spaceship-titanic, two tasks whose data are tables or text, parkinsons-disease and feedback, and two image tasks. The image tasks are fathomnet, which labels sea-floor animals in underwater images over 290 categories, and identify-contrails, which segments aviation contrails in satellite images. The benchmark's 7 tasks outside its Kaggle group, which train a language, vision or graph model or ask for faster code, are left out. Each seed holds out a fifth of a task's training rows as the hidden rows on which the quality is scored, drawn afresh per seed, by patient on parkinsons-disease, by image on fathomnet and by record on identify-contrails. The script sees the other four fifths as its training files. A tabular script scores in seconds on a CPU, and an image script trains its model on a GPU within a limit of 20 minutes on fathomnet and 30 minutes on identify-contrails, after which it counts as a script that did not run. Its artifact is the editable section of the benchmark's starter script, which must define the feature function, the fitted model and the two printed scores. The scaffold around it loads the data, prints the scores and produces the submission file from the feature function applied to the hidden rows. Claude Haiku 4.5 fills every role through its command-line interface, with a thinking budget of 1,024 tokens and no tools. Under no evaluator and under the inspector every change is applied, and the harness records whether the composed script ran to completion, which it sees without the hidden rows. The artifact that it holds at a round, and therefore the artifact that any rule delivers, is the last script that ran, as a deployed harness keeps its last working version. The quality curves of this family are the quality of that script. The validation check and the model evaluator do not keep a script that did not run, so the policy changes nothing there. Without it, 13 of LOOPSTOP's 72 stops would deliver a script that did not run. A form of this family in which the loop regenerated the whole script was run first with the Qwen3.5 generator. Of its proposals, 6 percent ran to a submission, so its quality measured whole-file generation instead of the loop, and it is not reported.

Table I.1: The true quality of each family, which is held out from every role of the loop.

family	true quality
cruise control	the best achievable cost of a drive over the cost the tuned controller achieves, on a held-out drive that no role sees, so one is the best possible controller
cart-pole	the best achievable cost of a balancing episode over the achieved cost, on a held-out episode that no role sees
ARC-AGI	the share of cells of the output grid that match the hidden answer
BigCodeBench-Hard	the share of the hidden tests that the repaired file passes, of which the loop sees at most the visible two fifths
MLAgentBench-Kaggle	the score of the script's predictions on the hidden fifth of the rows, namely the accuracy on spaceship-titanic, one minus the share that remains of a constant predictor's error on house-price (absolute error against the median) and feedback (root mean squared error against the mean over the six scores), and one minus the SMAPE share on parkinsons-disease, each floored at zero, and on the hidden images or records the mean average precision at twenty of the predicted category list on fathomnet and the mean Dice coefficient of the predicted mask on identify-contrails

Table I.2: The loops and the model in every role, every Qwen3.5 call made with thinking off. The true quality is held out from every role. Rounds are per trajectory and seeds are per scenario, and a row with two generators holds a scenario per generator. A trajectory whose first attempt leaves nothing to revise, as on ARC-AGI when the model's first reply holds no grid, ends at round 0 and is left out of every count and score.

family	scenario	generator	evaluator	rounds	seeds
cruise control	no evaluator	Qwen3.5 4B or 9B	none, every proposal applied	100	20
	simulation-based evaluator	Qwen3.5 4B or 9B	a single simulated drive scored by how fast the car recovers, how much it overshoots, and how hard it accelerates, which is the full criterion, or by the first two alone, which is the narrow criterion	100	20
	model evaluator	Qwen3.5 9B	Qwen3.5 9B or 4B, shown both settings and their simulated responses	100	20
	inspector	Qwen3.5 9B, which also applies the fix	Qwen3.5 9B or 4B, names a gain, a direction, and a magnitude	100	20
cart-pole	no evaluator	Qwen3.5 9B or 4B	none, every proposal applied	100	20
	simulation-based evaluator	Qwen3.5 9B or 4B	a simulated tilt response scored by recovery time, cart travel, and peak force, which is the full criterion, or by recovery time and cart travel alone, which is the narrow criterion	100	20
	model evaluator	Qwen3.5 9B	Qwen3.5 9B or 4B, shown both settings and their simulated responses	100	20
	inspector	Qwen3.5 9B, which also applies the fix	Qwen3.5 9B or 4B, names a gain, a direction, and a magnitude	100	20
ARC-AGI	no evaluator	Qwen3.5 9B or 4B	none	60	20
	model evaluator	Qwen3.5 9B	Qwen3.5 9B or 4B, shown the examples and both grids	60	20
	inspector	Qwen3.5 9B, which also applies the fix	Qwen3.5 9B or 4B, names a region and what is wrong there	60	20
BigCodeBench-Hard	no evaluator	Qwen3.5 9B or 4B, shown the visible tests	none, every proposal applied	60	5
	visible tests	Qwen3.5 9B or 4B, shown the visible tests	a check that keeps a change only when more of the visible two fifths of the tests pass	60	5
	model evaluator	Qwen3.5 9B	Qwen3.5 9B, shown both source files	60	5
	inspector	Qwen3.5 9B, which also applies the fix	Qwen3.5 9B, names a problem	60	5
MLAgentBench-Kaggle	no evaluator	Claude Haiku 4.5, shown what the script printed	none, every proposal applied, the last script that ran held	60	3
	validation check	Claude Haiku 4.5, shown what the script printed	a check that keeps a section only when the validation score the script prints is better than the kept one's	60	3
	model evaluator	Claude Haiku 4.5	Claude Haiku 4.5, shown both sections and what the script printed with each	60	3
	inspector	Claude Haiku 4.5, which also applies the fix	Claude Haiku 4.5, names a problem, the last script that ran held	60	3

I.3 The Two Settings and the Sketch

Table I.3 gives the memory and the tolerated delay used in every experiment of the paper. The five constants of Algorithm 1 follow from the two by (G.1), as Appendix G derives, and nothing else in the algorithm is chosen. The sketch has dimension $m = 256$, and a text artifact is split into the single tokens of the regular expression `[A-Za-z_][A-Za-z_0-9]*|\d+|[^\sA-Za-z_0-9]`, so that a token is a run of letters, digits and underscores that starts with a letter or an underscore, a run of digits, or a single other character that is not white space. The update is divided by the square root of the mean number of parts of the two artifacts, as Theorem E.3 suggests. A numeric coordinate v enters the sketch as $\operatorname{sign}(v)\log(1+|v|/s)$ with s its magnitude at the start, so that a relative change changes the sketch by the same amount at every magnitude. Appendix I.7 records which of these choices were made on the data. Algorithm F.1 uses a minimum of 10 rejected rounds and a threshold of $1/2$ on φ, the midpoint of its range.

Table I.3: The two settings of LOOPSTOP, the value used in every experiment, and what fixes it.

setting	value used	fixed by
memory τ	5 changes	the number of changes that the statistic should average, a few
tolerated delay D	4 rounds	the rounds that the deployment tolerates after a loop that has demonstrated a direction settles, at the smallest value at which no single round can reach the threshold

I.4 SENSITIVITY TO THE MEMORY AND THE DELAY

Figure I.2 runs Algorithm 1 exactly as specified, with the five constants of (G.1), over memories of 2 to 10 changes and tolerated delays of 3 to 12 rounds. Both settings act in the same direction, since raising either setting raises the evidence that a loop has to supply, and both therefore spend more rounds. A long memory delays the stop by about the memory, since the updates taken before settling stay in the average. On the cruise-control loops at the delay used, the stop changes from 7.0 rounds at a memory of 2 changes to 41.2 rounds at a memory of 10 changes. Over the whole grid the cruise-control quality lies between 0.386 and 0.565 and the rounds spent run from 7.0 to 51.6, whereas the cart-pole quality peaks at the memory used and decreases beyond it. The two families favor opposite memories, since the cruise-control loops keep improving while their evaluator still accepts and the cart-pole loops reach their peak and decline within a few changes of settling. The delay was fixed first, at the smallest value at which no single round can reach the threshold, before the memory was chosen. The memory used is then the smallest in the figure at which the algorithm delivered more than the version of the algorithm that preceded it on each of the four families with repeated seeds, as they stood when the setting was fixed. It lies 0.046 below the best cruise-control cell, which is the longest memory in the figure, since that family's model-evaluator scenarios keep increasing for 30 to 40 rounds and its other scenarios decline by then. An earlier set of BigCodeBench-Hard trajectories was available when the memory was chosen, and the trajectories reported here replaced it after the setting was fixed. The two narrow-criterion scenarios of the cruise-control family were likewise added after the setting was fixed.

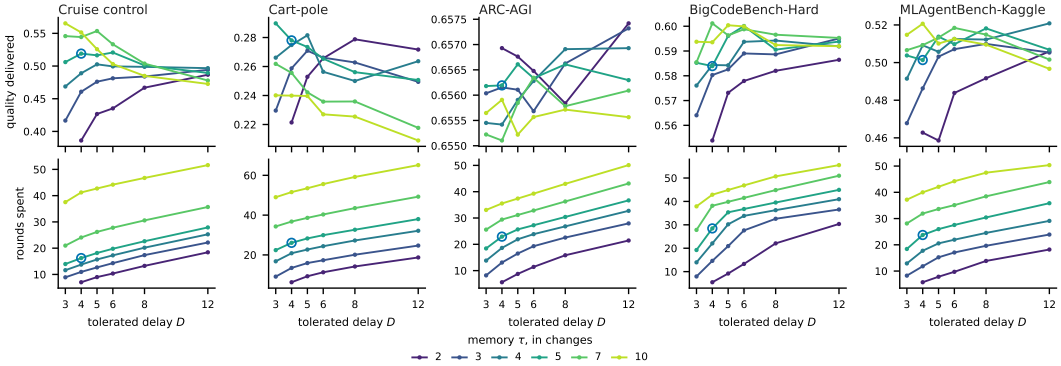

Figure I.2: Algorithm 1 over its memory τ, in changes, and tolerated delay D, in rounds. Quality delivered above and rounds spent below, mean over the trajectories of each family, a line per memory, each plot on its own vertical range. The open circle marks the setting used, $\tau = 5$ and $D = 4$. A memory of two has no point at $D = 3$, where $\Lambda^* \leq 0$.

I.5 THE SETTINGS OF THE SKETCH

Sweeping the sketch dimension from 16 to 1,024 leaves the cruise-control loops at 16.2 rounds and a quality of 0.519 at every dimension, and it changes the BigCodeBench-Hard quality between 0.584 and 0.587 at 28.4 to 28.5 rounds. Splitting a text artifact into single tokens, token pairs, or token triples, with the update divided by the square root of the part count or left undivided, gives 6 choices. They deliver a quality between 0.584 and 0.587 at 28.4 to 28.6 rounds on BigCodeBench-Hard, and the setting used is within 0.003 of the best of them. Giving the numeric coordinates the divisor one instead of their magnitude at the start takes the cruise-control quality from 0.519 to 0.521 at the same 16.2 rounds. These values are computed once by `results_sketch_settings.py` on the

loops of Table 1 and are not part of the chain that regenerates the tables, since each point recomputes every sketch of a family.

I.6 The Thresholds, Fixed Once and Cross-Fitted

Every baseline of Table 1 has a single free threshold, chosen from a grid. Table 1 gives each rule the value fixed once for every family, which is the grid value that maximizes the mean over the five families of the quality delivered, each family counted once. That is the value that a rule would carry to a new family. LoopStop's two settings are fixed by hand, as Appendix I.3 states, and Figure I.2 shows what every other grid point gives. Table I.4 instead chooses each baseline's threshold for each scenario on the trajectories of the other scenarios of the same family and scores it on the scenario held out. The value chosen there is the grid value that minimizes the mean over those trajectories of $1 - q_t + 0.002\,t$, where t is the round at which the rule stops the loop, q_t is the true quality delivered, and 0.002 is the cost of a round in units of quality. A rule that does not trigger therefore pays 0.2 for its 100 rounds. The charge enters that choice alone, since the thresholds of Table 1 and the two settings of LoopStop are chosen with no charge on the rounds. Table I.5 lists the fixed settings of every rule, its grid, the value used in Table 1, and the threshold chosen on each of the 10 folds of the cruise-control family.

Table I.4: Table 1 with every baseline's threshold chosen on the other scenarios of its family and applied to the scenario held out. Rounds used and quality delivered, mean over the loops of each family, with the standard error over its seeds. LoopStop is as in Table 1.

	Cruise control		Cart-pole		ARC-AGI		BigCodeBench-Hard		MLAgentBench-Kaggle		Mean	
Rule	Rounds	Quality	Rounds	Quality	Rounds	Quality	Rounds	Quality	Rounds	Quality	Rounds	Quality
Run to the budget	$100_{\pm 0}$	$0.393_{\pm 0.011}$	$100_{\pm 0}$	$0.204_{\pm 0.016}$	$60.0_{\pm 0}$	$0.650_{\pm 0.001}$	$60.0_{\pm 0}$	$0.579_{\pm 0.003}$	$60.0_{\pm 0}$	$0.489_{\pm 0.018}$	$76.0_{\pm 0}$	$0.463_{\pm 0.005}$
Idealized upper bound	$18.2_{\pm 0}$	$0.597_{\pm 0.015}$	$30.1_{\pm 0}$	$0.335_{\pm 0.016}$	$9.0_{\pm 0.0093}$	$0.654_{\pm 0.001}$	$45.0_{\pm 0}$	$0.595_{\pm 0.004}$	$32.2_{\pm 0}$	$0.527_{\pm 0.016}$	$26.9_{\pm 0.0019}$	$0.542_{\pm 0.006}$
PoR	$58.0_{\pm 0.91}$	$0.396_{\pm 0.009}$	$49.5_{\pm 0.72}$	$0.193_{\pm 0.009}$	$39.8_{\pm 0.088}$	$0.649_{\pm 0.001}$	$39.7_{\pm 0.12}$	$0.563_{\pm 0.003}$	$36.3_{\pm 0.23}$	$0.465_{\pm 0.016}$	$44.7_{\pm 0.24}$	$0.453_{\pm 0.004}$
PoU	$54.4_{\pm 0.95}$	$0.375_{\pm 0.011}$	$43.2_{\pm 0.78}$	$0.209_{\pm 0.013}$	$32.8_{\pm 0.44}$	$\mathbf{0.657}_{\pm 0.001}$	$34.0_{\pm 0.27}$	$\mathbf{0.588}_{\pm 0.004}$	$27.6_{\pm 0.34}$	$0.496_{\pm 0.013}$	$38.4_{\pm 0.27}$	$0.465_{\pm 0.004}$
RLF	$21.1_{\pm 1.4}$	$0.353_{\pm 0.016}$	$6.2_{\pm 0.45}$	$0.207_{\pm 0.012}$	$5.0_{\pm 0.00043}$	$0.652_{\pm 0.001}$	$5.1_{\pm 0.0067}$	$0.546_{\pm 0.007}$	$6.1_{\pm 0.12}$	$0.437_{\pm 0.014}$	$8.7_{\pm 0.30}$	$0.439_{\pm 0.005}$
LoopStop (ours)	$16.2_{\pm 0.36}$	$\mathbf{0.519}_{\pm 0.016}$	$26.1_{\pm 1.0}$	$\mathbf{0.278}_{\pm 0.017}$	$22.9_{\pm 0.052}$	$0.656_{\pm 0.001}$	$28.5_{\pm 0.16}$	$0.584_{\pm 0.004}$	$23.7_{\pm 0.70}$	$\mathbf{0.501}_{\pm 0.017}$	$23.5_{\pm 0.26}$	$\mathbf{0.508}_{\pm 0.006}$

I.7 The Selection Effect

The algorithm was designed on the cruise-control, cart-pole and ARC-AGI families and is scored on the same trajectories. The baselines' thresholds of Table 1 are likewise chosen on the trajectories that they are scored on, over all five families, and Table I.4 chooses them with the scenario held out. Three choices were made on the data, namely the demonstrated peak as the quantity to use, the form of the cap in the reference value κ_k after two alternatives were measured, and the memory, as the paragraph on the sweep above states. The convention $\rho = \tau/2$ was fixed after the form of the cap had been chosen. The delay and the constants λ, κ and Λ^* derived in Appendix G were not chosen on the data. The BigCodeBench-Hard trajectories reported here arrived after the setting was fixed and played no part in it.

Table I.5: Every rule of Tables 1, I.7 and I.12, with its settings, its grid, the threshold used in Table 1, and the threshold chosen on each fold of Table I.4, which holds out the scenario named. The columns are the 10 cruise-control scenarios of Table I.2, namely none, the simulation-based evaluator on the full and on the narrow criterion, the model evaluator, and the inspector, where 4B or 9B names the generator on the first three and the evaluator on the last two.

				threshold chosen with the scenario held out										
rule	fixed settings	grid	used	none 4B	none 9B	full 4B	full 9B	narrow 4B	narrow 9B	model 4B	model 9B	inspector 4B	inspector 9B
patience on rejections (OpenHands, 2026)	none	consecutive rejected rounds, 3, 5, 8, 10, 15, 20, 30, or 50	10	8	8	8	8	8	10	8	8	8	8
patience on update magnitude (OpenHands, 2026; Shrivastava, 2026)	an update is small when its magnitude is at most 0.05 of the largest so far	consecutive small rounds, 3, 5, 8, 10, 15, 20, or 30	8	8	8	8	8	8	10	5	8	8	8
run-length filter on y_t (Adams & MacKay, 2007)	the coherence $y_t = c_t/w_t$ of Section 3.2 formed over every round with $\lambda = 1/2$, a Gamma emission whose shape is estimated from the update lengths, a hazard of $1/50$ per round, a warm-up of three rounds, run lengths kept up to 300, and a stop once the run lengths below 15 hold the posterior mass and their mean y_t is below one	posterior mass of 0.5, 0.7, 0.8, 0.9, or 0.95	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
CUSUM on the accept decisions (Page, 1954)	the accept rate ν_0 estimated over the first five rounds, the alternative rate $\nu_1 = 0.3\,\nu_0$, and Page's sum of the log-likelihood ratio of each decision under ν_1 against ν_0	threshold on the sum, 1, 2, 4, 6, 8, or 12	12	12	12	12	12	12	12	6	12	12	12
LOOPSTOP (ours), Algorithm 1	$\tau = 5$ changes and $D = 4$ rounds, so $\lambda = 2/3$, $\kappa = 3$, $\rho = 5/2$ and $\Lambda^* = 5$	none, fixed	as left						fixed				
LOOPSTOP (ours) over a family of views, Algorithm F.1	as LOOPSTOP, with $n_0 = 10$ and the threshold $1/2$ on φ	none, fixed						fixed					

I.8 TWO RULES UNDER A SENTENCE EMBEDDING

Semantic early stopping thresholds the distance between consecutive drafts under a neural embedding, and Table 1 runs that rule with the sketch in its place. Table I.6 runs it as published, with every artifact embedded by the sentence-transformer all-MiniLM-L6-v2, on all five families, with a text as itself, a grid as its rows of digits and a parameter vector as its entries, and it runs LOOPSTOP on the same embeddings. The patience rule gains a small amount from the embedding on *BigCodeBench-Hard*, where it then passes LOOPSTOP on the sketch by 0.006, and it gains 0.001 on *ARC-AGI*, where it then passes LOOPSTOP on the sketch by 0.002. The larger part of both differences comes from the threshold, which Table I.6 chooses with the scenario held out as in Table I.4. With the sketch and that threshold the patience rule already delivers 0.588 on *BigCodeBench-Hard* and 0.657 on *ARC-AGI*, so the embedding adds 0.002 and 0.001, and at the threshold fixed once for every family in Table 1 the rule delivers less than LOOPSTOP on both families. LOOPSTOP itself delivers 0.001 less on the sketch than on the embedding on *BigCodeBench-Hard* and the same quality on *ARC-AGI*. On *MLAgentBench-Kaggle* the embedding gives patience on update magnitude 0.004 less at 18 more rounds, and it gives LOOPSTOP 0.001 less at about twice the rounds. On the two controller families the embedding gives patience on update magnitude 0.023 and 0.014 more at a few more rounds, and it gives LOOPSTOP 0.011 less on *cruise control* and 0.005 more on *cart-pole*, both within one standard error. The sketch therefore gives up a small amount of quality to a learned representation while needing no model. On the same CPU thread the embedding costs between $4\times$

and $500\times$ what our sketch costs (Table 2), the least on MLAgentBench-Kaggle, whose section is replaced whole each round. The embedding model, all-MiniLM-L6-v2 with 23 million parameters and 6 layers, is among the smallest in common use. A smaller model would still run a forward pass over the whole artifact at every round, which is a cost that grows with the artifact, whereas the sketch takes a single pass over the edit.

Table I.6: Patience on update magnitude and LOOPSTOP, each with the sketch and with a sentence embedding. Rounds used and quality delivered, mean over the loops of each family, with the patience rule's threshold chosen on the other scenarios of its family as in Table I.4, both representations scored on the same loops.

		with the sketch		with the embedding	
family	rule	rounds	quality	rounds	quality
cruise control	patience on update magnitude	54.4	0.375	58.3	0.398
cruise control	LOOPSTOP	16.2	0.519	15.8	0.508
cart-pole	patience on update magnitude	43.2	0.209	48.8	0.223
cart-pole	LOOPSTOP	26.1	0.278	26.1	0.283
ARC-AGI	patience on update magnitude	32.8	0.657	28.5	0.658
ARC-AGI	LOOPSTOP	22.9	0.656	23.3	0.656
BigCodeBench-Hard	patience on update magnitude	34.0	0.588	29.7	0.590
BigCodeBench-Hard	LOOPSTOP	28.5	0.584	29.9	0.585
MLAgentBench-Kaggle	patience on update magnitude	27.6	0.496	45.6	0.492
MLAgentBench-Kaggle	LOOPSTOP	23.7	0.501	41.2	0.500

I.9 WHY LOOPSTOP TAKES THE ROUNDS THAT IT TAKES

Theorem 3(ii) bounds the mean number of rounds to the stop, counted from the round at which a loop has settled and its smoothed statistic is back at one, by $(\Lambda^* + \kappa)/(\kappa - 1) = 4$ at $\kappa = 3$ and $\Lambda^* = 5$. The bound applies to a loop that has demonstrated a peak of $9/5$ or above, and it counts from the fade and not from the peak. On a loop that changes at every round the smoothed statistic decreases by the factor $\lambda = 2/3$ per change toward one, so it takes about 6 changes to fade from a peak of $9/5$ to within a tenth of one. The warm-up of five changes comes before both the fade and the bound. On the two no-evaluator scenarios of the cruise-control family, which apply every proposal, the best rounds are 6 and 15 (Table I.12) and LOOPSTOP stops at rounds 12 and 13. The scenario whose best round is 6 is stopped 6 rounds after it, within the six rounds of fade and the four rounds of the bound. The other scenario, whose best round is 15, is stopped two rounds before it, on a curve that is flat there, and gives up 0.06 of quality to the best budget. The four scenarios with a simulation-based evaluator accept 9 to 13 percent of proposals. There the five changes of the warm-up and about four more before the sum could reach the threshold, 9 changes in all, would take 70 to 100 rounds. The unchanged rounds, each scoring $\kappa - \min\{\bar{y}_k, \kappa\}$ once the direction has faded, stop those loops at 12 to 18 rounds instead, which gives 16.2 rounds over the whole family.

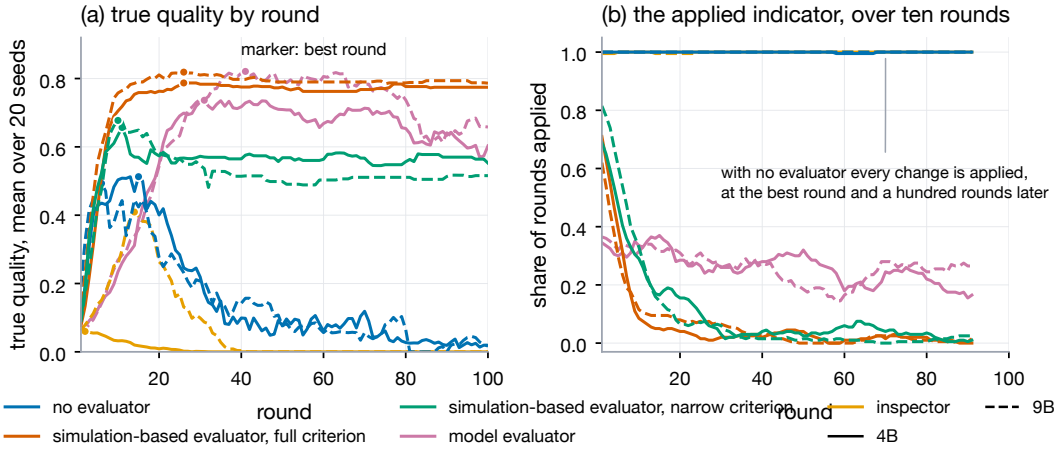

Figure I.3: Loops settle early, and on the no-evaluator scenarios the accept indicator does not say so. *(a)* True quality by round, mean over 20 seeds, best round marked. *(b)* Share of rounds applied, averaged over 10 rounds.

I.10 The Quality Remaining past the Best Round

Past its best round every cruise-control scenario holds, declines, or collapses, as Figure I.3 draws. The two scenarios with a simulation-based evaluator on the full criterion hold their level, going from 0.787 after 26 rounds to 0.774 after 100 rounds for the smaller generator and from 0.819 to 0.786 for the larger. Both scenarios on the narrow criterion decline further, from 0.657 after 11 rounds to 0.553 and from 0.677 after 10 rounds to 0.516. The two model-evaluator scenarios increase longest and then decline, from 0.736 after 31 rounds to 0.603 and from 0.821 after 41 rounds to 0.658. Collapse to 0.000 is the outcome of the two inspector scenarios, from 0.060 after 2 rounds and from 0.409 after 14 rounds, and of the two no-evaluator scenarios, from 0.513 after 15 rounds to 0.019 and from 0.493 after 6 rounds to 0.017. The scenarios with a simulation-based evaluator admit 0.36 to 0.55 of proposals over the first 20 rounds and 0.010 to 0.033 over the last 50.

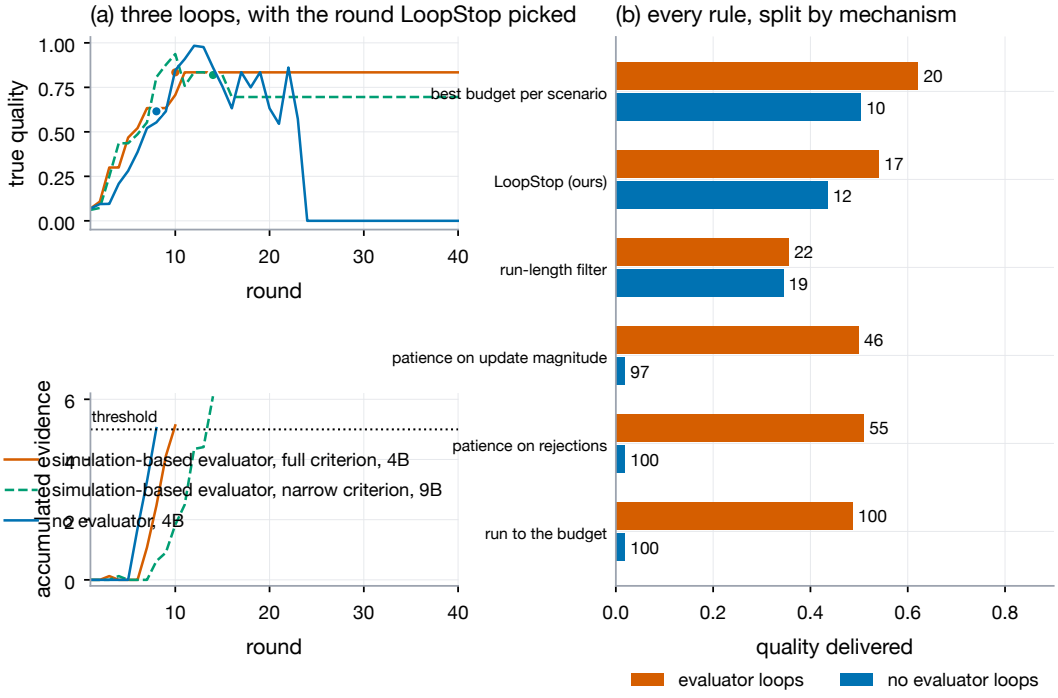

Figure I.4: LoopStop on three loops and every rule. *(a)* True quality by round, the chosen round marked, Λ_t below. *(b)* Quality every rule delivers on the cruise-control loops with and without an evaluator, rounds beside each bar.

I.11 The Rules Split by Mechanism

Figure I.4 draws the sum Λ_t against the round on three loops and the quality that every rule of Tables 1 and I.7 delivers split by mechanism.

Table I.7: LoopStop with the unchanged rounds left out of the algorithm, so that they are not scored after the warm-up, do not fade the smoothed statistic and add nothing before the warm-up, on the five families, scored by the protocol of Table 1. Mean and standard error over the seeds of a family.

	cruise control, 200 loops		cart-pole, 200 loops		ARC-AGI, 44834 loops		BigCodeBench-Hard, 3360 loops		MLAgentBench-Kaggle, 72 loops	
rule	rounds	quality	rounds	quality	rounds	quality	rounds	quality	rounds	quality
LoopStop (ours)	$16.2_{\pm 0.36}$	$0.519_{\pm 0.016}$	$26.1_{\pm 1.0}$	$0.278_{\pm 0.017}$	$22.9_{\pm 0.052}$	$0.656_{\pm 0.001}$	$28.5_{\pm 0.16}$	$0.584_{\pm 0.004}$	$23.7_{\pm 0.70}$	$0.501_{\pm 0.017}$
rounds that changed nothing left unscored	$46.3_{\pm 1.8}$	$0.556_{\pm 0.012}$	$53.9_{\pm 1.0}$	$0.209_{\pm 0.016}$	$35.0_{\pm 0.051}$	$0.654_{\pm 0.000}$	$40.3_{\pm 0.12}$	$0.582_{\pm 0.004}$	$34.8_{\pm 0.72}$	$0.502_{\pm 0.010}$

I.12 LoopStop with the Unchanged Rounds Left Unscored

Table I.7 runs Algorithm 1 on the five families with the unchanged rounds left out of the algorithm. Such a round is then not scored after the warm-up, it does not fade the smoothed statistic toward one,

and it adds nothing before the warm-up, so that only the changing rounds count. Leaving them out changes the result in both directions. The ablation spends 46.3 rounds on cruise control against 16.2 rounds for LOOPSTOP, and 53.9 rounds on cart-pole against 26.1 rounds. The extra rounds gain 0.037 ± 0.011 of quality on cruise control, whose model-evaluator scenarios keep increasing for 30 to 40 rounds, and give up 0.069 ± 0.020 on cart-pole, where a loop whose evaluator has stopped accepting cannot reach the threshold.

I.13 THE RULES COMPARED

Run to the budget, which is the first of the two references, spends every round that the harness allows, which is how deployed agent frameworks behave, since OpenHands, AutoGen and SWE-agent each stop at a limit on iterations, replies or cost set by the framework (Wang et al., 2025; Wu et al., 2024; Yang et al., 2024). *Patience on rejections* stops after a tuned number of consecutive rejected rounds, and *patience on update magnitude* stops after a tuned number of consecutive rounds whose update magnitude is at most a fixed fraction of the largest update magnitude up to that round. Both rules are modeled on the stuck detector that OpenHands runs by default, which stops on repeated actions and errors (OpenHands, 2026). The second is also semantic early stopping (Shrivastava, 2026) with the sketch in place of a model embedding, since that rule thresholds the distance between consecutive drafts. *Run-length filter* is the rule of Adams & MacKay (2007) on the same coherence, which keeps a posterior over the length of the current segment instead of a two-state latent variable. *Rounds that changed nothing left unscored* of Table I.7 is Algorithm 1 with the unchanged rounds left out of the algorithm, as the paragraph above states, so that only the rounds that changed the artifact carry evidence, at the same threshold.

Table I.8: Three views of the same cruise-control loop. *removes* is the share of rounds on which some part's quantity is lower than at the round before. The last column is the share of loops in which the first check of Algorithm F.1 drops the view at its fifth change, because the view has removed nothing by then. The artifact is the only view that the loop delivers.

view	what it holds	how it changes	removes	dropped at change 5
artifact	the gains the loop holds	a rejected round leaves it alone	0.204	50%
proposal	the gains just offered	replaced every round, accepted or not	0.478	31%
transcript	every reply so far	appended to, not revised	0.000	100%

I.14 A FAMILY OF VIEWS

The family form protects against a view that does not settle, and its cost is a check that cannot distinguish an artifact whose parts have only grown up to a round from a view that only ever grows. A family of a single view is the special case that the rows of Table 1 use, and Table I.8 shows three views of the same cruise-control loop. Algorithm F.1 was offered four candidate views of every cruise-control loop, namely the artifact, the artifact with the divisor one, the proposal and the transcript, with the two admission checks switched off. It stops at 27.4 ± 0.6 rounds and delivers a quality of 0.501 ± 0.014, against 16.2 ± 0.4 rounds at a quality of 0.519 ± 0.016 for the artifact alone, as mean and standard error over the 20 seeds. A view that still shows a direction delays the algorithm, since a changing round contributes the smallest score over the views that changed, and the transcript, which only grows, is that view. With the two checks run online, the transcript is dropped at its fifth change in every loop, and the proposal, which changes whether or not the loop accepts, is dropped in 86 percent of the loops once 10 rounds have been rejected. The family then stops at 15.1 ± 0.4 rounds for a quality of 0.477 ± 0.016, with 2.7 of the four views admitted at the stop on average. That is earlier than the artifact alone and lower by 0.042. The reason is that the first check also drops the artifact in half of the loops at its fifth change, when its two parameters have only increased up to that change, until the first round on which either parameter decreases. In those rounds the family is the proposal view, which alone stops at 12.7 ± 0.2 rounds for a quality of 0.462 ± 0.016. A family of the artifact and the transcript under the same checks stops at 18.8 ± 0.5 rounds for a quality of 0.530 ± 0.013. The single view chosen by hand remains the setting that the paper reports.

I.15 The Cruise-Control Family by Evaluator

Table I.9 splits the cruise-control column of Table 1 into the 8 scenarios with an evaluator and the two scenarios without an evaluator. With an evaluator LoopStop stops at 17.2 rounds and delivers a quality of 0.540, below the best budget per scenario at 0.621. Patience on rejections stops after 55 rounds there and delivers 0.509, and patience on update magnitude stops after 46 rounds and delivers 0.498. On the two no-evaluator scenarios the patience rules rarely trigger and spend the budget. LoopStop stops those loops at 12.2 rounds, two rounds past the best budget per scenario at 10.5 rounds, and delivers 0.434 against 0.503.

Table I.9: The cruise-control family, all scenarios and split by evaluator. Rounds are the mean over the 20 seeds, and quality is their mean and standard error, each seed averaged over the column's scenarios. The blocks hold the rules, the ablation of Table I.7, and the two fixed budgets.

	all ten scenarios		eight with an evaluator		two without	
rule	rounds	quality	rounds	quality	rounds	quality
PoR (OpenHands, 2026)	64.1	0.411 ± 0.010	55.2	0.509 ± 0.011	100.0	0.018 ± 0.012
PoU (OpenHands, 2026; Shrivastava, 2026)	56.5	0.402 ± 0.010	46.4	0.498 ± 0.012	97.2	0.018 ± 0.012
RLF (Adams & MacKay, 2007)	21.1	0.353 ± 0.016	21.7	0.355 ± 0.013	18.9	0.344 ± 0.053
LoopStop (ours)	16.2	0.519 ± 0.016	17.2	0.540 ± 0.014	12.2	0.434 ± 0.045
rounds that changed nothing left unscored	46.3	0.556 ± 0.012	54.8	0.587 ± 0.010	12.2	0.434 ± 0.045
run to the budget (Wang et al., 2025; Wu et al., 2024; Yang et al., 2024)	100.0	0.393 ± 0.011	100.0	0.486 ± 0.013	100.0	0.018 ± 0.012
best budget per scenario	18.2	0.597 ± 0.015	20.1	0.621 ± 0.015	10.5	0.503 ± 0.034

I.16 Paired Differences by Seed

Table I.10 gives the difference in delivered quality between LoopStop and every other rule, formed within a seed and then averaged, so the seed-to-seed variation that both rules share cancels. A positive value means that LoopStop delivers more. Against leaving the unchanged rounds unscored, LoopStop is 0.069 ± 0.020 above on cart-pole and 0.037 ± 0.011 below on cruise control, whose model-evaluator scenarios keep increasing. It is within a standard error of zero on *ARC-AGI*, BigCodeBench-Hard and MLAgentBench-Kaggle, where the unchanged rounds are few.

Table I.10: Delivered quality of LoopStop minus that of each rule, paired by seed, mean and standard error over seeds. A positive value favors LoopStop. The blocks hold the rules of Table 1, the ablation of Table I.7, and the two fixed budgets.

rule	cruise control	cart-pole	ARC-AGI	BigCodeBench-Hard	MLAgentBench-Kaggle
PoR (OpenHands, 2026)	$+0.108 \pm 0.021$	$+0.068 \pm 0.019$	$+0.006 \pm 0.001$	$+0.019 \pm 0.002$	$+0.023 \pm 0.010$
PoU (OpenHands, 2026; Shrivastava, 2026)	$+0.117 \pm 0.017$	$+0.055 \pm 0.019$	$+0.003 \pm 0.001$	$+0.006 \pm 0.004$	$+0.012 \pm 0.011$
RLF (Adams & MacKay, 2007)	$+0.166 \pm 0.016$	$+0.071 \pm 0.021$	$+0.004 \pm 0.001$	$+0.037 \pm 0.007$	$+0.065 \pm 0.031$
rounds that changed nothing left unscored	-0.037 ± 0.011	$+0.069 \pm 0.020$	$+0.002 \pm 0.001$	$+0.002 \pm 0.003$	-0.001 ± 0.007
run to the budget (Wang et al., 2025; Wu et al., 2024; Yang et al., 2024)	$+0.126 \pm 0.022$	$+0.074 \pm 0.023$	$+0.006 \pm 0.001$	$+0.005 \pm 0.003$	$+0.013 \pm 0.012$
best budget per scenario	-0.078 ± 0.012	-0.057 ± 0.017	$+0.002 \pm 0.001$	-0.011 ± 0.003	-0.026 ± 0.016

I.17 The Best Budget per Scenario, Cross-Fitted

The last row of Table I.10 compares LoopStop with budgets taken from the same trajectories on which it is scored, a single budget per scenario. Each budget is the maximum of a curve whose standard error over seeds at that round runs from 0.001 on *ARC-AGI* to 0.073 on *cart-pole*, so part of that reference is noise selected by the maximum. Table I.11 chooses every scenario's budget on the other seeds and scores it on the seed held out. The reference decreases by 0.068 on cruise control, by 0.078 on cart-pole, by 0.002 on ARC-AGI, by 0.013 on BigCodeBench-Hard and by 0.022 on MLAgentBench-Kaggle. Against the cross-fitted reference LoopStop delivers more on cart-pole and ARC-AGI and is within a standard error of it on cruise control, where it is 0.011 ± 0.014 below, on BigCodeBench-Hard, where it is 0.001 ± 0.003 above, and on MLAgentBench-Kaggle, where it is 0.004 ± 0.010 below. Per scenario, the ARC-AGI quality curves are flat after round three, so a budget chosen for a single scenario is largely the round at which the noise peaked.

Table I.11: The best budget per scenario, taken from the same trajectories and cross-fitted over seeds. Mean and standard error over seeds, and the paired difference of LOOPSTOP against the cross-fitted reference.

family	in-sample	cross-fitted	LOOPSTOP minus cross-fitted
cruise control	0.597 ± 0.015	0.530 ± 0.015	-0.011 ± 0.014
cart-pole	0.335 ± 0.016	0.257 ± 0.016	$+0.021 \pm 0.014$
ARC-AGI	0.654 ± 0.001	0.653 ± 0.001	$+0.003 \pm 0.001$
BigCodeBench-Hard	0.595 ± 0.004	0.582 ± 0.003	$+0.001 \pm 0.003$
MLAgentBench-Kaggle	0.527 ± 0.016	0.505 ± 0.011	-0.004 ± 0.010

I.18 EVERY SCENARIO ON ITS OWN

Table I.12 gives Table 1 for every scenario on its own, with rounds used and quality delivered by every rule, each baseline at its threshold fixed once for every family, and with the standard error over the seeds of the scenario. It adds CUSUM, which is Page's cumulative sum on the accept decisions with the settings of Table I.5. The scenarios differ in where a rule stops. Without an evaluator every proposal is applied and the artifact keeps changing, so a rule that waits for the changes to stop spends the budget, whereas under an evaluator the accepted changes thin out and the same rule stops early.

Table I.12: Every rule of Table 1 on every scenario, and CUSUM, Page's cumulative sum on the accept decisions, whose settings Table I.5 gives. Rounds used and quality delivered, mean over the loops of the scenario, with the standard error over its seeds. Each baseline runs at its threshold fixed once for every family.

scenario	run to the budget	PoR	CUSUM	PoU	RLF	LOOPSTOP	best budget
cruise control							
no evaluator, 4B	$100, 0.019_{\pm 0.019}$	$100, 0.019_{\pm 0.019}$	$100, 0.019_{\pm 0.019}$	$99, 0.019_{\pm 0.019}$	$24, 0.343_{\pm 0.063}$	$13, 0.448_{\pm 0.063}$	$15, 0.513_{\pm 0.058}$
no evaluator, 9B	$100, 0.017_{\pm 0.017}$	$100, 0.017_{\pm 0.017}$	$100, 0.017_{\pm 0.017}$	$95, 0.017_{\pm 0.017}$	$14, 0.345_{\pm 0.074}$	$12, 0.420_{\pm 0.071}$	$6, 0.493_{\pm 0.058}$
simulation-based evaluator, full criterion, 4B	$100, 0.774_{\pm 0.027}$	$22, 0.777_{\pm 0.031}$	$15, 0.720_{\pm 0.034}$	$19, 0.763_{\pm 0.031}$	$11, 0.683_{\pm 0.041}$	$12, 0.743_{\pm 0.032}$	$26, 0.787_{\pm 0.029}$
simulation-based evaluator, full criterion, 9B	$100, 0.786_{\pm 0.024}$	$23, 0.814_{\pm 0.025}$	$14, 0.768_{\pm 0.041}$	$19, 0.809_{\pm 0.025}$	$9, 0.705_{\pm 0.046}$	$18, 0.774_{\pm 0.038}$	$26, 0.819_{\pm 0.024}$
simulation-based evaluator, narrow criterion, 4B	$100, 0.553_{\pm 0.030}$	$29, 0.560_{\pm 0.027}$	$16, 0.592_{\pm 0.039}$	$25, 0.545_{\pm 0.028}$	$13, 0.566_{\pm 0.037}$	$17, 0.624_{\pm 0.037}$	$11, 0.657_{\pm 0.043}$
simulation-based evaluator, narrow criterion, 9B	$100, 0.516_{\pm 0.055}$	$26, 0.522_{\pm 0.063}$	$19, 0.605_{\pm 0.057}$	$23, 0.562_{\pm 0.058}$	$16, 0.555_{\pm 0.062}$	$15, 0.617_{\pm 0.068}$	$10, 0.677_{\pm 0.064}$
model evaluator, 4B	$100, 0.603_{\pm 0.063}$	$66, 0.683_{\pm 0.046}$	$80, 0.666_{\pm 0.049}$	$48, 0.646_{\pm 0.056}$	$9, 0.129_{\pm 0.022}$	$23, 0.585_{\pm 0.046}$	$31, 0.736_{\pm 0.037}$
model evaluator, 9B	$100, 0.658_{\pm 0.048}$	$76, 0.719_{\pm 0.036}$	$86, 0.711_{\pm 0.034}$	$46, 0.660_{\pm 0.058}$	$6, 0.167_{\pm 0.056}$	$23, 0.609_{\pm 0.049}$	$41, 0.821_{\pm 0.042}$
inspector, 4B	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$92, 0.000_{\pm 0.000}$	$20, 0.030_{\pm 0.007}$	$14, 0.020_{\pm 0.007}$	$2, 0.060_{\pm 0.006}$
inspector, 9B	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$88, 0.005_{\pm 0.005}$	$15, 0.350_{\pm 0.056}$	$14, 0.409_{\pm 0.064}$
cart-pole							
model evaluator, 4B	$100, 0.374_{\pm 0.065}$	$34, 0.312_{\pm 0.047}$	$93, 0.386_{\pm 0.065}$	$27, 0.371_{\pm 0.055}$	$5, 0.205_{\pm 0.031}$	$33, 0.417_{\pm 0.056}$	$23, 0.438_{\pm 0.062}$
model evaluator, 9B	$100, 0.375_{\pm 0.057}$	$34, 0.348_{\pm 0.061}$	$95, 0.353_{\pm 0.059}$	$32, 0.374_{\pm 0.063}$	$5, 0.182_{\pm 0.026}$	$32, 0.483_{\pm 0.052}$	$23, 0.489_{\pm 0.056}$
inspector, 4B	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$100, 0.000_{\pm 0.000}$	$82, 0.000_{\pm 0.000}$	$6, 0.074_{\pm 0.038}$	$17, 0.040_{\pm 0.040}$	$2, 0.120_{\pm 0.043}$
inspector, 9B	$100, 0.051_{\pm 0.040}$	$96, 0.051_{\pm 0.040}$	$99, 0.051_{\pm 0.040}$	$100, 0.051_{\pm 0.040}$	$6, 0.257_{\pm 0.061}$	$17, 0.096_{\pm 0.042}$	$10, 0.299_{\pm 0.073}$
no evaluator, 4B	$100, 0.022_{\pm 0.022}$	$100, 0.022_{\pm 0.022}$	$100, 0.022_{\pm 0.022}$	$100, 0.022_{\pm 0.022}$	$11, 0.036_{\pm 0.025}$	$15, 0.002_{\pm 0.002}$	$3, 0.073_{\pm 0.034}$
no evaluator, 9B	$100, 0.075_{\pm 0.041}$	$99, 0.075_{\pm 0.041}$	$99, 0.075_{\pm 0.041}$	$98, 0.075_{\pm 0.041}$	$6, 0.010_{\pm 0.010}$	$15, 0.020_{\pm 0.014}$	$88, 0.078_{\pm 0.045}$
simulation-based evaluator, full criterion, 4B	$100, 0.309_{\pm 0.037}$	$26, 0.236_{\pm 0.041}$	$81, 0.271_{\pm 0.043}$	$23, 0.241_{\pm 0.032}$	$5, 0.278_{\pm 0.042}$	$42, 0.338_{\pm 0.052}$	$74, 0.384_{\pm 0.045}$
simulation-based evaluator, full criterion, 9B	$100, 0.232_{\pm 0.041}$	$28, 0.376_{\pm 0.051}$	$64, 0.364_{\pm 0.053}$	$21, 0.364_{\pm 0.055}$	$5, 0.369_{\pm 0.053}$	$36, 0.476_{\pm 0.058}$	$17, 0.475_{\pm 0.060}$
simulation-based evaluator, narrow criterion, 4B	$100, 0.423_{\pm 0.049}$	$27, 0.390_{\pm 0.051}$	$83, 0.376_{\pm 0.040}$	$22, 0.397_{\pm 0.048}$	$5, 0.286_{\pm 0.035}$	$32, 0.410_{\pm 0.061}$	$43, 0.475_{\pm 0.062}$
simulation-based evaluator, narrow criterion, 9B	$100, 0.178_{\pm 0.022}$	$38, 0.294_{\pm 0.048}$	$77, 0.199_{\pm 0.022}$	$30, 0.336_{\pm 0.048}$	$6, 0.373_{\pm 0.048}$	$22, 0.499_{\pm 0.050}$	$18, 0.519_{\pm 0.055}$
ARC-AGI							
model evaluator, 4B	$60, 0.660_{\pm 0.002}$	$29, 0.655_{\pm 0.002}$	$57, 0.659_{\pm 0.002}$	$37, 0.660_{\pm 0.002}$	$5, 0.656_{\pm 0.002}$	$42, 0.660_{\pm 0.002}$	$27, 0.660_{\pm 0.002}$
model evaluator, 9B	$60, 0.660_{\pm 0.002}$	$44, 0.659_{\pm 0.001}$	$55, 0.660_{\pm 0.002}$	$29, 0.665_{\pm 0.002}$	$5, 0.661_{\pm 0.002}$	$34, 0.661_{\pm 0.002}$	$19, 0.663_{\pm 0.001}$
inspector, 4B	$60, 0.646_{\pm 0.002}$	$60, 0.645_{\pm 0.002}$	$57, 0.644_{\pm 0.002}$	$59, 0.646_{\pm 0.002}$	$5, 0.647_{\pm 0.001}$	$14, 0.656_{\pm 0.002}$	$2, 0.651_{\pm 0.002}$
inspector, 9B	$60, 0.651_{\pm 0.001}$	$60, 0.651_{\pm 0.001}$	$57, 0.650_{\pm 0.001}$	$59, 0.652_{\pm 0.002}$	$5, 0.651_{\pm 0.002}$	$14, 0.658_{\pm 0.001}$	$2, 0.654_{\pm 0.002}$
no evaluator, 4B	$60, 0.637_{\pm 0.001}$	$60, 0.637_{\pm 0.001}$	$53, 0.635_{\pm 0.001}$	$50, 0.643_{\pm 0.001}$	$5, 0.642_{\pm 0.001}$	$18, 0.641_{\pm 0.002}$	$2, 0.643_{\pm 0.002}$
no evaluator, 9B	$60, 0.650_{\pm 0.001}$	$60, 0.650_{\pm 0.001}$	$57, 0.650_{\pm 0.001}$	$55, 0.655_{\pm 0.001}$	$5, 0.653_{\pm 0.001}$	$15, 0.660_{\pm 0.001}$	$2, 0.655_{\pm 0.001}$
BigCodeBench-Hard							
no evaluator, 4B	$60, 0.529_{\pm 0.014}$	$55, 0.529_{\pm 0.014}$	$56, 0.530_{\pm 0.014}$	$53, 0.543_{\pm 0.013}$	$5, 0.506_{\pm 0.010}$	$16, 0.538_{\pm 0.007}$	$56, 0.547_{\pm 0.012}$
no evaluator, 9B	$60, 0.602_{\pm 0.009}$	$56, 0.602_{\pm 0.009}$	$55, 0.602_{\pm 0.008}$	$53, 0.602_{\pm 0.007}$	$5, 0.598_{\pm 0.010}$	$15, 0.615_{\pm 0.007}$	$18, 0.620_{\pm 0.016}$
visible tests, 4B	$60, 0.625_{\pm 0.008}$	$12, 0.579_{\pm 0.006}$	$51, 0.622_{\pm 0.008}$	$37, 0.620_{\pm 0.007}$	$5, 0.548_{\pm 0.004}$	$45, 0.608_{\pm 0.005}$	$57, 0.625_{\pm 0.008}$
visible tests, 9B	$60, 0.675_{\pm 0.004}$	$12, 0.636_{\pm 0.010}$	$52, 0.675_{\pm 0.003}$	$39, 0.671_{\pm 0.005}$	$5, 0.609_{\pm 0.011}$	$45, 0.666_{\pm 0.004}$	$57, 0.675_{\pm 0.004}$
model evaluator, 9B	$60, 0.602_{\pm 0.008}$	$28, 0.606_{\pm 0.010}$	$51, 0.595_{\pm 0.010}$	$24, 0.588_{\pm 0.004}$	$5, 0.544_{\pm 0.014}$	$35, 0.595_{\pm 0.010}$	$49, 0.612_{\pm 0.010}$
inspector, 9B	$60, 0.439_{\pm 0.007}$	$60, 0.439_{\pm 0.007}$	$60, 0.438_{\pm 0.006}$	$59, 0.441_{\pm 0.008}$	$5, 0.475_{\pm 0.015}$	$15, 0.481_{\pm 0.012}$	$33, 0.492_{\pm 0.009}$
MLAgentBench-Kaggle							
no evaluator, Haiku	$60, 0.484_{\pm 0.017}$	$60, 0.484_{\pm 0.017}$	$60, 0.484_{\pm 0.017}$	$60, 0.484_{\pm 0.017}$	$7, 0.438_{\pm 0.036}$	$13, 0.468_{\pm 0.031}$	$9, 0.528_{\pm 0.015}$
validation check, Haiku	$60, 0.550_{\pm 0.010}$	$18, 0.537_{\pm 0.014}$	$44, 0.548_{\pm 0.009}$	$19, 0.542_{\pm 0.009}$	$5, 0.488_{\pm 0.014}$	$41, 0.548_{\pm 0.010}$	$38, 0.551_{\pm 0.009}$
model evaluator, Haiku	$60, 0.503_{\pm 0.033}$	$25, 0.475_{\pm 0.025}$	$30, 0.488_{\pm 0.030}$	$19, 0.468_{\pm 0.032}$	$6, 0.417_{\pm 0.030}$	$26, 0.501_{\pm 0.040}$	$45, 0.507_{\pm 0.035}$
inspector, Haiku	$60, 0.418_{\pm 0.044}$	$60, 0.418_{\pm 0.044}$	$60, 0.418_{\pm 0.044}$	$53, 0.465_{\pm 0.021}$	$6, 0.404_{\pm 0.028}$	$15, 0.488_{\pm 0.011}$	$37, 0.522_{\pm 0.021}$

The scenarios of a family differ enough that their average hides where each rule stops. Figures I.5, I.6, I.7, I.8 and I.9 therefore repeat Figure 2 scenario by scenario, with LOOPSTOP's stop marked on each and its evidence against the threshold, which shows where the stop comes from.

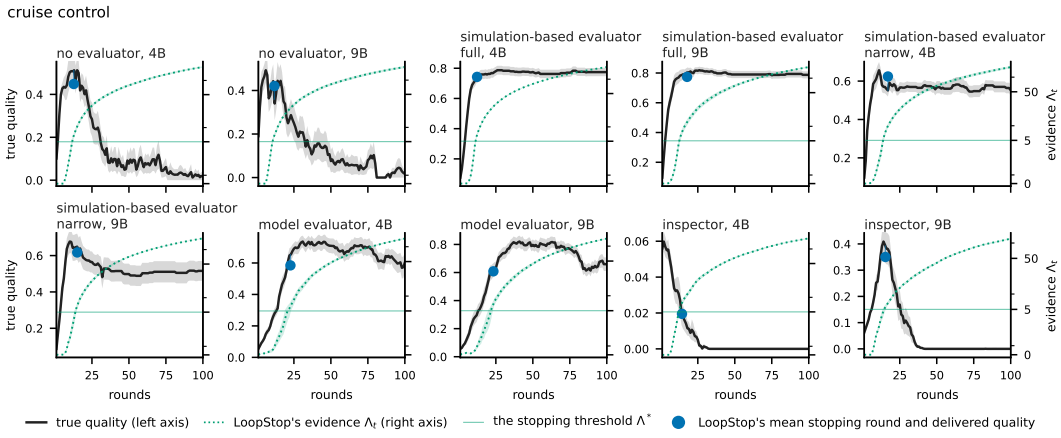

Figure I.5: Every cruise-control scenario by round, drawn as in Figure 2, with the true quality on the left axis and LOOPSTOP's evidence Λ_t on the right, each a mean over the loops with a band of one standard error, the threshold Λ^*, and LOOPSTOP's mean stopping round and delivered quality. The short vertical segment joins the marker to the quality curve.

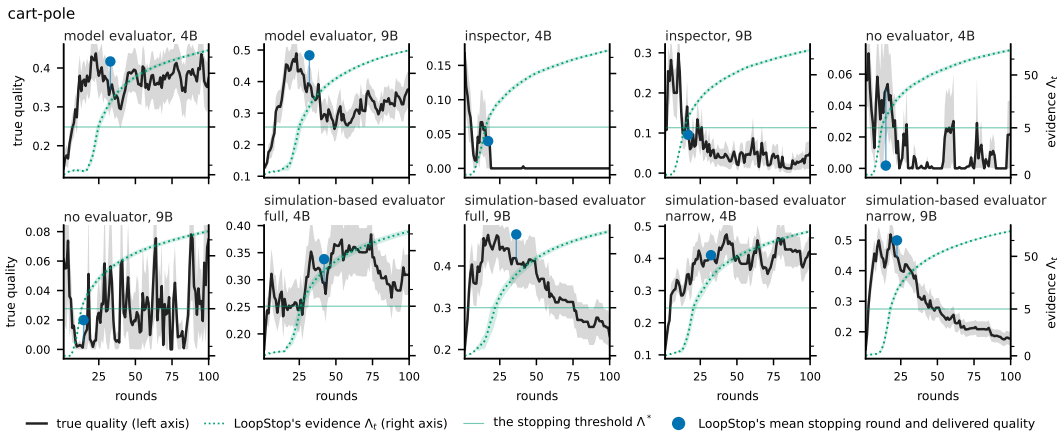

Figure I.6: Every cart-pole scenario by round, drawn as in Figure 2, with the true quality on the left axis and LOOPSTOP's evidence Λ_t on the right, each a mean over the loops with a band of one standard error, the threshold Λ^*, and LOOPSTOP's mean stopping round and delivered quality.

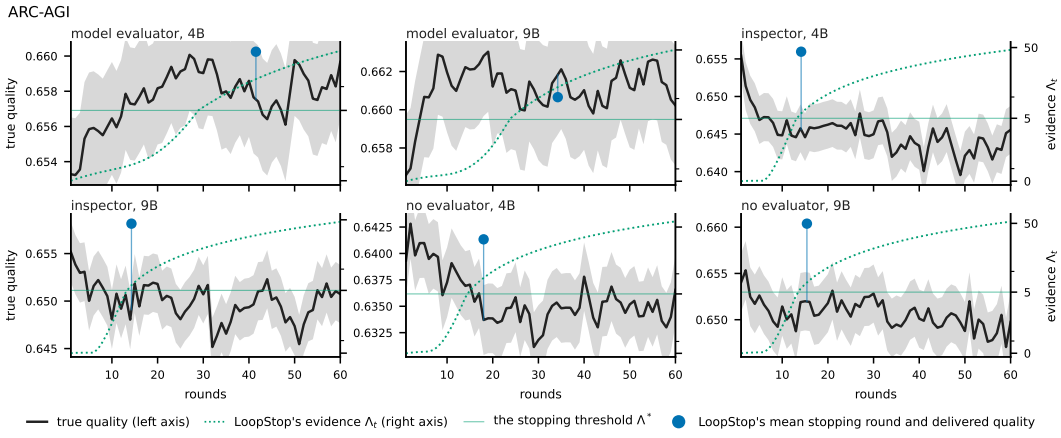

Figure I.7: Every ARC-AGI scenario by round, drawn as in Figure 2, with the true quality on the left axis and LOOPSTOP's evidence Λ_t on the right, each a mean over the loops with a band of one standard error, the threshold Λ^*, and LOOPSTOP's mean stopping round and delivered quality.

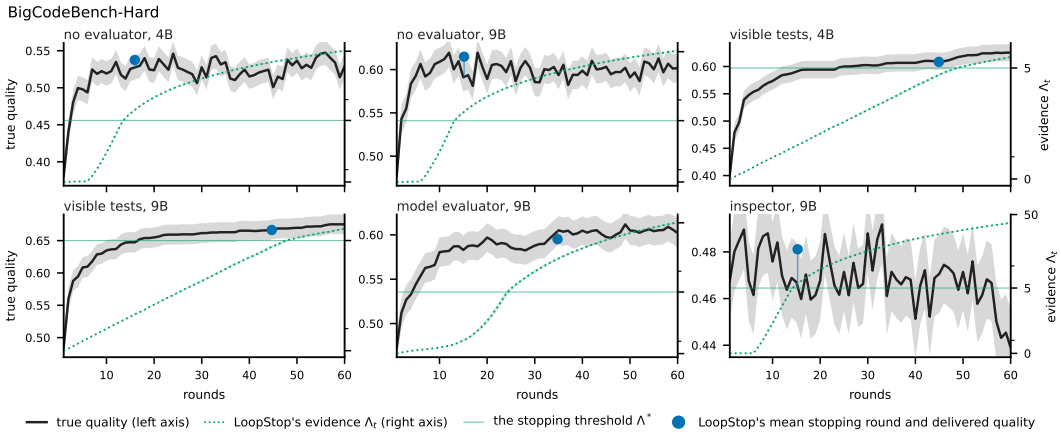

Figure I.8: Every scenario of BigCodeBench-Hard by round, drawn as in Figure 2, with the true quality on the left axis and LOOPSTOP's evidence Λ_t on the right, each a mean over the loops with a band of one standard error, the threshold Λ^*, and LOOPSTOP's mean stopping round and delivered quality.

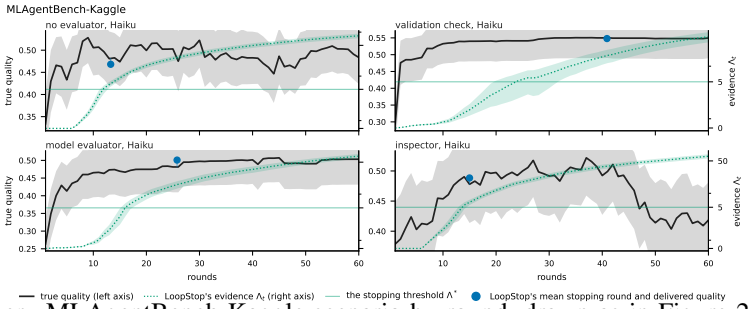

Figure I.9: Every MLAgentBench-Kaggle scenario by round, drawn as in Figure 2, with the true quality on the left axis and LOOPSTOP's evidence Λ_t on the right, each a mean over the loops with a band of one standard error, the threshold Λ^*, and LOOPSTOP's mean stopping round and delivered quality.

I.19 THE SOURCES OF THE EVIDENCE

The evidence of Algorithm 1 has two sources. A round that changed nothing scores $\kappa - \min\{\bar{y}_k, \kappa\}$, which is zero while the recent changes still share a direction and $\kappa - 1$ once that direction has faded. A round that changed the artifact scores $\kappa_k - y_k$, which is negative while the recent changes still share a direction and positive once that direction has faded. Figures I.10, I.11, I.12, I.13 and I.14 accumulate the two scores separately for every scenario, continued past the stop as if the loop had continued. They draw the evidence Λ_t beside the two, which is their sum floored at zero, on axes that are linear up to the threshold and logarithmic beyond it, together with the share of loops whose proposal was rejected at each round. Under an evaluator that rejects most proposals, the unchanged rounds supply most of the evidence, and the loop is stopped a few rounds after its changes stop sharing a direction. The changing rounds supply all of the evidence when there is no evaluator and every proposal is applied, and the sum then grows only once the changes stop sharing a direction. On the visible-tests scenarios of BigCodeBench-Hard the changes contribute nothing, since the warm-up of τ changes does not complete. The unchanged rounds contribute at the pace of each loop's own gaps, so the loops whose first change came within the first few rounds are stopped near round 40.

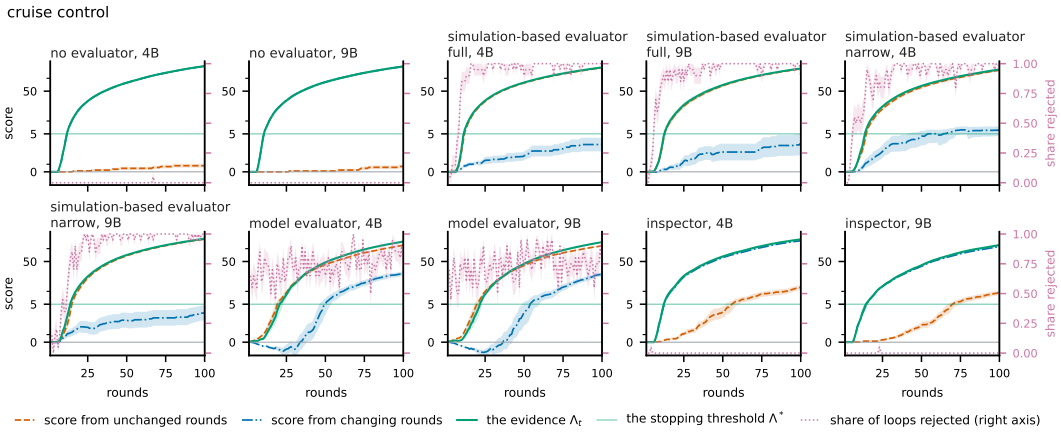

Figure I.10: The sources of the evidence on every cruise-control scenario, namely the score accumulated by unchanged rounds, the score accumulated by changing rounds, and the evidence Λ_t against the threshold Λ^*, each a mean over the loops with a band of one standard error, with the share of loops rejected at each round on the right axis.

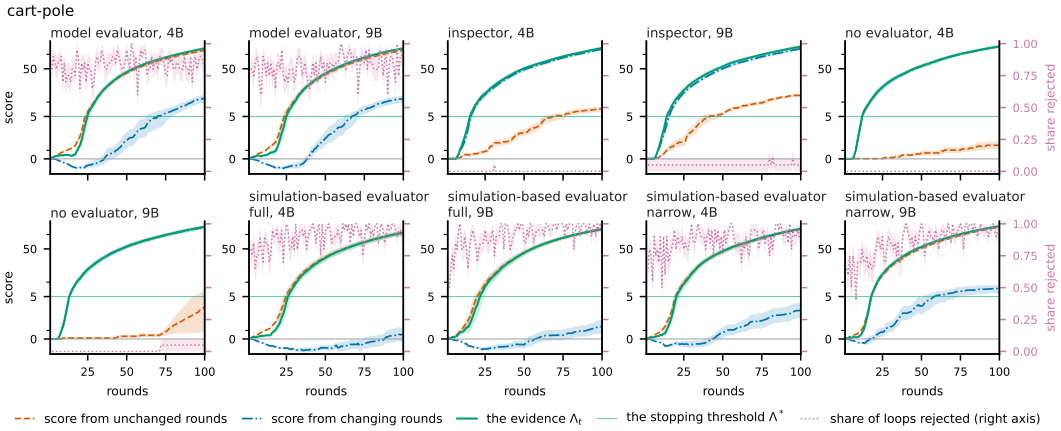

Figure I.11: The sources of the evidence on every cart-pole scenario, namely the score accumulated by unchanged rounds, the score accumulated by changing rounds, and the evidence Λ_t against the threshold Λ^*, each a mean over the loops with a band of one standard error, with the share of loops rejected at each round on the right axis.

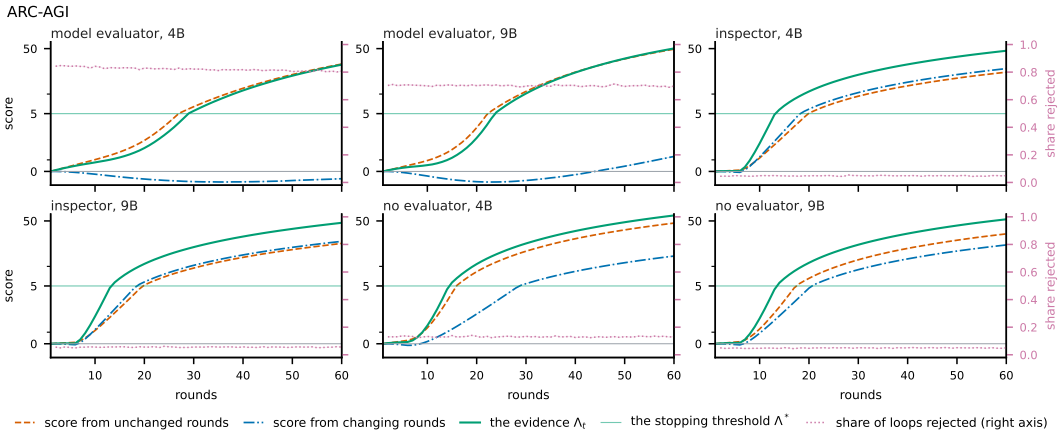

Figure I.12: The sources of the evidence on every ARC-AGI scenario, namely the score accumulated by unchanged rounds, the score accumulated by changing rounds, and the evidence Λ_t against the threshold Λ^*, each a mean over the loops with a band of one standard error, with the share of loops rejected at each round on the right axis.

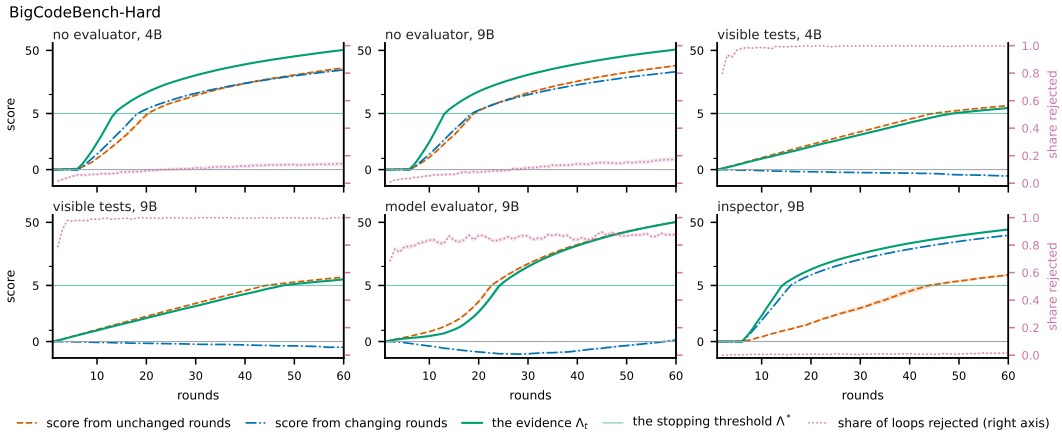

Figure I.13: The sources of the evidence on every scenario of BigCodeBench-Hard, namely the score accumulated by unchanged rounds, the score accumulated by changing rounds, and the evidence Λ_t against the threshold Λ^*, each a mean over the loops with a band of one standard error, with the share of loops rejected at each round on the right axis.

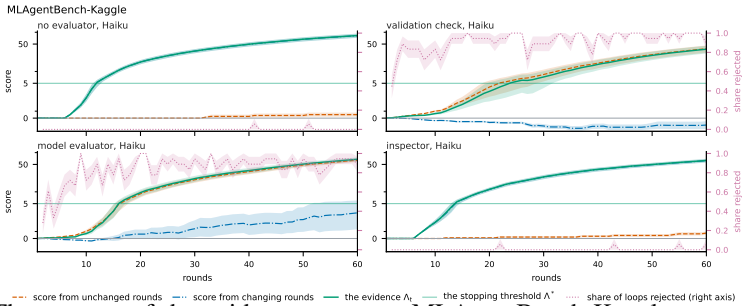

Figure I.14: The sources of the evidence on every MLAgentBench-Kaggle scenario, namely the score accumulated by unchanged rounds, the score accumulated by changing rounds, and the evidence Λ_t against the threshold Λ^*, each a mean over the loops with a band of one standard error, with the share of loops rejected at each round on the right axis.