SELF-EVOLUTION SHOULD BE STABLE, NOT FROZEN: LYAPUNOV-GATED RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improving agents increasingly modify not only their task behavior, but also the mechanisms that generate future modifications. This creates a control problem that current self-evolution objectives do not measure: a change can improve the next agent while making future evolution increasingly sensitive to evaluator noise, misleading memories, reward misspecification, or mutation randomness. We formulate recursive self-improvement as a non-stationary stochastic dynamical system and introduce **EVOLYAPUNOV**, a stability-constrained framework that allows the improver itself to evolve without freezing the meta-operation. EVOLYAPUNOV estimates an *interventional evolution response operator* from matched perturbation branches, decomposes evolution into capability and risk subspaces, and admits candidate self-modifications only when useful directions improve while measured risk dynamics satisfy a confidence-aware Lyapunov drift condition and an empirical finite-horizon amplification bound. We define the *Recursive Amplification Factor* (RAF) and evolution risk half-life, which expose failure modes invisible to one-generation benchmark scores. Under a stochastic drift condition, accepted updates keep expected risk energy bounded despite persistent perturbations; a complementary amplification bound connects the certificate to multi-generation error propagation. Our evaluation protocol spans SWE-bench Verified, Multi-SWE-bench, τ^2-Bench, BFCL V4, and ARC-AGI-2 with Qwen2.5-Coder-7B-Instruct and Qwen2.5-7B-Instruct backbones. A stress study reveals a sharp recursive stability boundary: unrestricted self-evolution can continue improving immediately after the estimated risk gain exceeds one, whereas Lyapunov-gated evolution remains on a capability–stability Pareto frontier. The central claim is that recursive self-improvement need not choose between open-ended evolution and stability; it should expand useful directions while contracting the directions along which mistakes compound.

1 INTRODUCTION

Self-evolving agents are moving from editing prompts and skills toward editing the procedures that generate future edits. The Darwin Gödel Machine (DGM) demonstrated open-ended code-level self-modification (Zhang et al., 2026a); HGM showed that current benchmark score can disagree with the quality of a branch's descendants (Wang et al., 2026a); HyperAgents made the meta-agent itself editable (Zhang et al., 2026b); MetaSkill-Evolve recursively evolves task and meta-skills on two time-scales (Wang et al., 2026b); and Meta^n obtains deeper recursion by repeatedly applying a fixed meta-operation Ω (Kim et al., 2026). These systems sharpen an unresolved tension. If the update machinery remains fixed, recursive improvement is bounded by a human-designed meta-rule. If the update machinery is allowed to change, a locally beneficial edit may also alter the *dynamics of future self-modification*.

Existing evaluation largely asks whether the next agent is better. This is insufficient for recursive systems. Let A_t be an agent and Φ_t the procedure that proposes, evaluates, and retains modifications. A recursive system evolves both,

$$A_{t+1} = \Phi_t(A_t, \mathcal{E}_t), \qquad \Phi_{t+1} = \Psi_t(\Phi_t, A_t, \mathcal{E}_t), \tag{1}$$

where $\mathcal{E}_t$ denotes traces, evaluations, and environmental feedback. A small evaluator error, misleading memory, or reward-model bias at generation t can therefore affect not only A_{t+1} but also the operator

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.

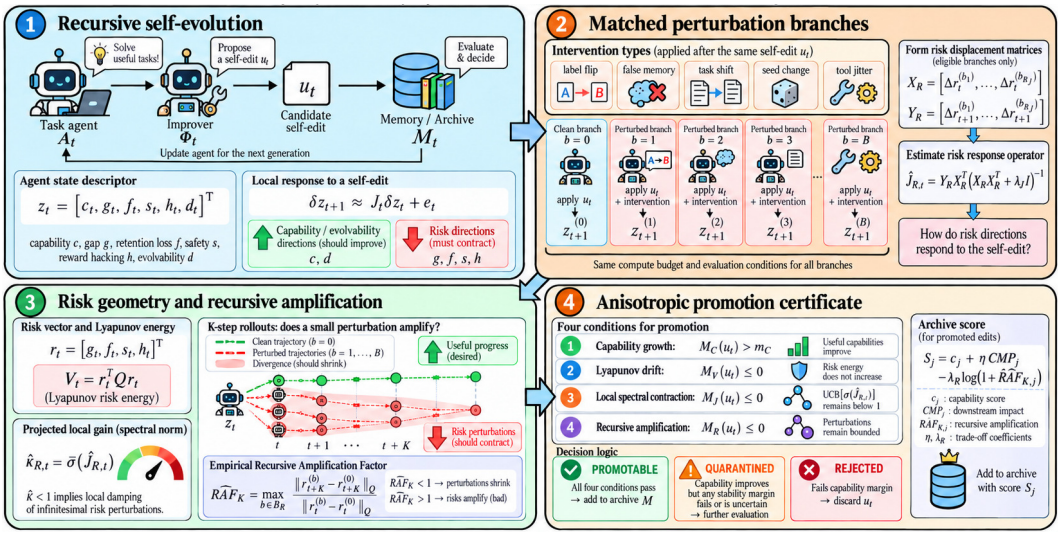

Figure 1: **EvoLyapunov separates growth from stability.** A candidate self-edit is cloned into matched perturbation branches. Their responses estimate a local evolution response operator and multi-generation amplification. Useful capability directions may improve, but risk directions must contract before the candidate is promoted.

that constructs A_{t+2} and beyond. Recent failures make this concrete: self-evolving skill libraries can silently drift (Zhang et al., 2026c); co-evolving evaluators require anchored outer audits (Zhang et al., 2026d); and HackProbe shows that repeated selection on an imperfect visible score can amplify reward hacking across generations (Yang et al., 2026). Yet these works diagnose or gate specific failures, rather than modeling the closed-loop propagation of perturbations through recursive evolution.

We introduce **EvoLyapunov**, which treats recursive self-improvement as a stochastic dynamical system and asks a different question: *does a candidate modification make the future improvement process more or less stable?* The key idea is *anisotropic stability*. We do not require the entire system to contract—that would suppress capability growth. Instead, we decompose evolution into useful capability directions and harmful risk directions. Capability is allowed to expand, while aggregate risk energy and registered risk perturbations must satisfy contraction-style stability bounds. Candidate edits are tested through matched perturbation branches, from which we estimate a local *interventional evolution response operator*. A Lyapunov gate promotes an edit only if (i) the lower confidence bound on capability improvement exceeds a fixed margin, (ii) the upper confidence bound on one-step risk-energy drift satisfies the prescribed Lyapunov margin, (iii) the upper confidence bound on the dominant risk-subspace response gain does not exceed one, and (iv) the empirical multi-generation Recursive Amplification Factor remains below a prescribed stability threshold.

This formulation connects three observations that have so far been studied separately. First, HGM's metaproductivity–performance mismatch implies that one-generation score is not a sufficient statistic for future evolution (Wang et al., 2026a). Second, Meta^n obtains stability by holding Ω fixed (Kim et al., 2026), suggesting that changing the improver is precisely where instability enters. Third, anytime-valid gates such as SEA (Sengupta, 2026) certify candidate changes against statistical error budgets, but do not measure whether an accepted change increases the sensitivity of *later* self-modifications. EvoLyapunov combines these perspectives by certifying the dynamics induced by a change, rather than only the change's immediate outcome.

Our contributions are:

- We formulate recursive self-evolution as a non-stationary stochastic dynamical system and introduce *anisotropic recursive stability*: capability directions may expand while risk directions contract.

- We propose an interventional estimator of the local evolution response operator from matched branches and define RAF_K, evolution risk half-life, and a confidence-aware Lyapunov promotion rule.
- We prove that accepted updates keep expected risk energy bounded under persistent bounded disturbances, and relate projected response-operator gains to finite-horizon perturbation amplification.
- We specify a cross-domain evaluation on software engineering, tool use, interactive coordination, and abstract reasoning. A stress study demonstrates these signatures using the analysis pipeline described below.

2 Related Work

Open-ended and recursive self-improvement. DGM maintains an archive of self-modified coding agents and selects descendants using benchmark performance (Zhang et al., 2026a). HGM replaces immediate score with clade metaproductivity, explicitly recognizing that a strong parent may not generate strong descendants (Wang et al., 2026a). HyperAgents unify a task agent and an editable meta-agent so the modification procedure can modify itself (Zhang et al., 2026b); MetaSkill-Evolve similarly evolves branch-local meta-skills on a slower time-scale (Wang et al., 2026b). Meta^n takes a complementary route: it keeps a universal meta-operation fixed and recursively applies it to a growing stack, using invariance of Ω as a source of stability (Kim et al., 2026). MOSS performs source-level rewriting in autonomous agents (Cai et al., 2026), while Q-Evolve co-evolves process supervision and policies within an in-distribution RL loop (Zhang et al., 2026e). Our focus is orthogonal: we measure whether a modification changes the *sensitivity* of future self-evolution to perturbations.

Evaluation drift and reliable self-evolution. Library Drift identifies unbounded skill accumulation as a silent source of retrieval degradation and false-positive skill injection (Zhang et al., 2026c). Double Ratchet co-evolves skills and inspectable evaluators while protecting held-out anchors (Zhang et al., 2026d). SEA confines modification and uses anytime-valid certificates for sequential acceptance decisions (Sengupta, 2026). HackProbe detects score–capability divergence with secret fixed probes and reselects candidates under suspected reward hacking (Yang et al., 2026). These methods motivate external anchors and conservative promotion, but they predominantly certify levels or detect particular failure channels. EvoLyapunov instead treats error propagation as a first-class quantity and asks whether a candidate update creates a locally expansive closed-loop evolution map.

Stability view. Lyapunov methods provide classical conditions under which stochastic or nonlinear systems remain bounded or contract toward stable sets (Khalil, 2002; Angeli, 2002). Applying this perspective to self-evolving agents is nontrivial because source-code edits are discrete and non-differentiable. Our interventional response operator is defined over measurable evolution descriptors rather than neural-network parameters. This makes stability testing compatible with arbitrary agent scaffolds and black-box foundation models.

3 Recursive Self-Evolution as a Dynamical System

At generation t, let $X_t = (A_t, \Phi_t, \mathcal{M}_t)$ denote the task agent, the improvement procedure, and its archive/memory. We map this high-dimensional object to a descriptor state

$$z_t = [c_t,\ g_t,\ f_t,\ s_t,\ h_t,\ d_t]^\top \in \mathbb{R}^6, \tag{2}$$

where c is held-out capability, g is evaluator–deployment gap, f is retention loss on an anchor suite, s is safety/policy violation rate, h is a reward-hacking susceptibility score, and d is archive diversity/evolvability. Except for c and d, smaller is better. A self-evolution step induces

$$z_{t+1} = F_{\Phi_t}(z_t, \xi_t), \tag{3}$$

where ξ_t summarizes mutation randomness, evaluator noise, memory contamination, tool/environment stochasticity, and task-mixture shift.

A standard self-evolving loop selects a candidate u_t using immediate capability c_{t+1}. HGM augments this with descendant quality (Wang et al., 2026a). Neither quantity distinguishes two candidates

that reach the same capability but have different responses to future perturbations. To formalize this distinction, we use a local empirical response model over the observed evolution descriptors:

$$\delta z_{t+1} \approx J_t \, \delta z_t + e_t, \tag{4}$$

where J_t denotes the local *interventional evolution response operator* and e_t collects direct exogenous intervention effects, higher-order terms, and stochastic mismatch. Importantly, J_t is an operational finite-difference response operator over observed descriptors; we do not claim that it identifies the structural derivative $\partial F_{\Phi_t}/\partial z$ separately from the effects of ξ_t. We do not estimate the full J_t. Instead, Section 4 directly estimates its induced risk block $J_{R,t}$ from the four observed risk descriptors.

Anisotropic stability. Let P_C project onto capability/evolvability coordinates and P_R onto the risk coordinates (g, f, s, h). We seek evolution that is expansive where useful but contractive where errors compound:

$$\mathbb{E}[c_{t+1} - c_t] > 0, \qquad \bar{\sigma}(J_{R,t}) < 1. \tag{5}$$

where $\bar{\sigma}$ denotes the largest singular value. Requiring contraction over the full descriptor state would also penalize locally expansive capability/evolvability directions and can therefore suppress useful open-ended change. Eq. (5) instead constrains only the harmful subspace. The dominant risk-subspace singular value defines the local spectral promotion margin, while the remaining spectrum is retained for diagnostic analysis. Cross-subspace coupling and nonlinear effects are controlled additionally through the Lyapunov-drift and finite-horizon amplification gates.

Recursive amplification. For a bounded perturbation restricted to the risk subspace, define the K-generation Recursive Amplification Factor

$$\mathrm{RAF}_K(t) = \sup_{\delta z_t \in \mathcal{R}_\epsilon} \frac{\|P_R \delta z_{t+K}\|_2}{\|P_R \delta z_t\|_2}. \tag{6}$$

where

$$\mathcal{R}_\epsilon = \{\delta z : P_C \delta z = 0,\ 0 < \|\delta z\|_2 \le \epsilon\}.$$

$\mathrm{RAF}_K > 1$ means the evolution loop amplifies a risk perturbation over K generations even if the immediate candidate scores well. We also report the *risk half-life*, the first generation at which the same perturbation's risk norm falls below one half of its initial value.

4 EvoLyapunov

Figure 1 summarizes EvoLyapunov. Each generation begins with a conventional self-evolution proposal, but promotion is mediated by a stability experiment performed on versioned clones of the candidate. The gate is scaffold-agnostic: the underlying proposer can be DGM-, HGM-, HyperAgent-, or MetaSkill-style, and the modified object may include prompts, tools, skills, source code, memory policies, or the improver itself.

4.1 Matched Interventional Evolution Branches

For candidate update $u_t : X_t \mapsto X'_t$, we create a clean branch $b = 0$ and B matched perturbation branches. Each branch receives the same evaluation budget and matched execution conditions except for a controlled intervention $\delta^{(b)}$. We use five perturbation families: evaluator label flips, a false but plausible memory insertion, task-mixture reweighting, mutation-seed displacement, and tool-latency/failure jitter. These interventions may act through the exogenous variables ξ_t and induce measurable deviations in the evolution descriptors.

After applying intervention b but before any descendant-generation step, we recompute the four frozen risk descriptors using the same external anchor and probe definitions as in the clean branch. We define

$$\delta r_t^{(b)} = r_t^{(b)} - r_t^{(0)}, \qquad \delta r_{t+1}^{(b)} = r_{t+1}^{(b)} - r_{t+1}^{(0)}.$$

Only branches whose pre-rollout displacement exceeds the calibrated noise floor are used for response-operator identification. For response-operator identification, we retain branches in which the controlled intervention induces a measurable pre-rollout displacement in the risk descriptors. Let

$$\mathcal{I}_t = \left\{ b \in \{1, \dots, B\} : \|\delta r_t^{(b)}\|_Q > \epsilon_0 \right\}, \qquad B_J = |\mathcal{I}_t|.$$

Enumerating the eligible branches as $\mathcal{I}_t = \{b_1, \ldots, b_{B_J}\}$, let

$$X_R = [\delta r_t^{(b_1)}, \ldots, \delta r_t^{(b_{B_J})}], \qquad Y_R = [\delta r_{t+1}^{(b_1)}, \ldots, \delta r_{t+1}^{(b_{B_J})}]. \tag{7}$$

We estimate the empirical one-step risk response operator with ridge regularization,

$$\hat{J}_{R,t} = Y_R X_R^\top \left(X_R X_R^\top + \lambda_J I \right)^{-1}. \tag{8}$$

Perturbations that act only through exogenous transition noise and do not induce a measurable pre-rollout risk displacement are retained as auxiliary stress tests, but are not used for fitting $\hat{J}_{R,t}$ or for computing $\widehat{\mathrm{RAF}}_K$. The estimator is intentionally defined over observed risk descriptors; it remains meaningful when the underlying edit is discrete or the base model is accessible only through an API. We interpret $\hat{J}_{R,t}$ as an empirical local response operator rather than as an estimator of a structural derivative. Natural lineage variation is confounded by proposal selection and evaluation noise, whereas matched interventions provide controlled local probes of the evolution response. Appendix B gives the branch design and rank diagnostics.

4.2 Risk Geometry and Lyapunov Energy

The risk vector is $r_t = [g_t, f_t, s_t, h_t]^\top$. Each raw risk coordinate is nonnegative and is scale-normalized, without mean centering, using a frozen positive calibration scale. We then define the quadratic risk energy

$$V_t = V(r_t) = r_t^\top Q r_t, \qquad Q = \mathrm{diag}(q_g, q_f, q_s, q_h), \quad q_i > 0. \tag{9}$$

Q expresses deployment priorities and is fixed before evolution; in the default configuration all scale-normalized risks have equal weight. The projected local gain is

$$\hat{\kappa}_{R,t} = \bar{\sigma}(\hat{J}_{R,t}). \tag{10}$$

In the fitted local response model, $\hat{\kappa}_{R,t} < 1$ indicates local damping of infinitesimal risk perturbations; promotion uses an upper confidence bound rather than this point estimate. Because linearization can be misleading over several self-modification steps, EvoLyapunov additionally performs short-horizon branch rollouts of length K and estimates $\widehat{\mathrm{RAF}}_K$ directly. All reported experiments use $Q = I_4$ after scale normalization; accordingly, the spectral and amplification analyses below use the Euclidean norm.

4.3 Finite-Horizon Recursive Amplification

For branch b, let $r_{t+k}^{(b)} - r_{t+k}^{(0)}$ be its matched risk deviation after k further self-evolution generations. We define the set of empirically identifiable branches as

$$\mathcal{B}_R = \left\{ b : \|r_t^{(b)} - r_t^{(0)}\|_Q > \epsilon_0 \right\},$$

where ϵ_0 is the calibrated clean-branch noise floor. We use

$$\widehat{\mathrm{RAF}}_K = \max_{b \in \mathcal{B}_R} \frac{\|r_{t+K}^{(b)} - r_{t+K}^{(0)}\|_Q}{\|r_t^{(b)} - r_t^{(0)}\|_Q}, \tag{11}$$

where $\|x\|_Q = (x^\top Q x)^{1/2}$. The registered interventions are selected by measurable pre-rollout risk displacement and need not satisfy the idealized restriction $P_C \delta z_t = 0$ in Eq. (6). Therefore, $\widehat{\mathrm{RAF}}_K$ is an operational empirical amplification statistic over the finite registered perturbation suite, rather than an estimator or upper bound of the idealized supremum RAF_K. Accordingly, the promotion certificate controls amplification only with respect to this registered suite and does not certify arbitrary perturbations outside it. This operationalizes a question that one-step regression tests miss: *does the candidate make future self-modifications magnify small errors?* We estimate uncertainty through paired bootstrap over tasks and perturbation seeds. A conservative upper bound $\mathrm{UCB}(\widehat{\mathrm{RAF}}_K)$ is used for promotion.

4.4 The Anisotropic Promotion Certificate

For each candidate, EvoLyapunov evaluates four margins.

Capability growth. On an external held-out task bank unavailable to the proposer,

$$M_C(u_t) = \mathrm{LCB}_{1-\delta_C}[c(X'_t) - c(X_t)]. \tag{12}$$

We require $M_C(u_t) > m_C \geq 0$.

Lyapunov drift. Using the perturbation branches, define the conditional drift

$$D_t^{(b)} = V(r_{t+1}^{(b)}) - (1-\alpha)V(r_t^{(b)}). \tag{13}$$

The candidate must satisfy

$$M_V(u_t) = \mathrm{UCB}_{1-\delta_V}[\mathbb{E}[D_t]] - \epsilon_V \leq 0. \tag{14}$$

Eq. (14) has the same algebraic form as the drift condition in Proposition 1 with $\kappa = 1 - \alpha$ and $\sigma^2 = \epsilon_V$. Proposition 1 applies when the intervention-averaged drift upper-bounds the conditional drift of the promoted lineage.

Local spectral contraction. When the interventional response operator passes the identification diagnostics, we define the one-step spectral margin

$$M_J(u_t) = \mathrm{UCB}_{1-\delta_J}\left[\bar{\sigma}(\hat{J}_{R,t})\right] - \rho_J. \tag{15}$$

We use $\rho_J = 1$. Thus the upper confidence bound on the dominant risk-subspace singular value must not exceed the local non-expansion boundary. If the response-operator identification diagnostics fail, the candidate does not pass the spectral screen.

Recursive amplification. Finally,

$$M_R(u_t) = \mathrm{UCB}_{1-\delta_R}\left(\widehat{\mathrm{RAF}}_K\right) - \rho_{\max} \leq 0. \tag{16}$$

We set $K = 6$ and $\rho_{\max} = 0.95$ by default. A candidate is PROMOTABLE iff

$$M_C(u_t) > m_C, \qquad M_V(u_t) \leq 0, \qquad M_J(u_t) \leq 0, \qquad M_R(u_t) \leq 0.$$

Candidates that pass the capability margin but fail any stability margin are QUARANTINED; candidates that fail the capability margin are REJECTED. Quarantined candidates are retained in the archive for analysis but are forbidden from producing descendants. If multiple candidates are promotable, one is selected using the archive score in Eq. (24). If none is promotable, the active lineage remains unchanged.

Complete promotion pseudocode is provided in Appendix A.

4.5 ARCHIVE SELECTION WITHOUT FREEZING THE IMPROVER

Among promotable candidates, we combine immediate capability, HGM-style clade metaproductivity, and an empirical amplification penalty. The exact archive score is given in the appendix. Unlike Meta^n, the meta-operation itself remains editable.

5 STABILITY GUARANTEES

Under the stated stochastic drift condition, promoted lineages have bounded expected risk energy. Under a uniform local response-gain bound $\bar{\kappa} < 1$ and uniformly bounded residuals, finite-horizon risk perturbations decay geometrically up to a disturbance floor. Full statements and proofs are given in Appendix D.

6 EXPERIMENTAL PROTOCOL

Backbones and evolution hosts. We instantiate the same evolution harness with **Qwen2.5-Coder-7B-Instruct** for software editing and **Qwen2.5-7B-Instruct** for general tool-use and reasoning tasks. The active self-modification object contains the solver prompt, tool policy, memory/skill files, proposer code, candidate-selection code, and—for recursive variants—the meta-improver itself. All candidate code is executed in a versioned sandbox.

Table 1: **Endpoint capability after 12 evolution generations.** Values are task success/accuracy percentages from the study; ARC-AGI-2 uses percent solved. Stability metrics are intentionally excluded from this endpoint table and are reported separately in figures and appendix diagnostics.

Method	SWE-V	Multi-SWE	τ^2	BFCL V4	ARC-AGI-2
Static	31.2	24.5	40.7	60.1	0.0
DGM	48.6	36.8	49.2	65.4	1.4
HGM	52.8	39.4	51.0	66.2	1.9
HyperAgents	54.1	41.2	53.1	67.8	2.2
Meta^n	51.7	40.6	52.4	67.1	4.6
HackProbe+DGM	46.9	35.9	49.6	66.0	1.8
EvoLyapunov	**55.6**	**43.8**	**55.2**	**69.3**	**5.1**

Benchmarks. We cover five complementary regimes. **SWE-bench Verified** measures real-world issue resolution (Jimenez et al., 2024); **Multi-SWE-bench** adds 1,632 multilingual issue-resolution tasks spanning seven languages (Zan et al., 2025); τ^2**-Bench** stresses long-horizon coordination in a dual-control telecom environment (Barres et al., 2025); **BFCL V4** measures multi-turn tool use and function calling (Patil et al., 2025); and **ARC-AGI-2** tests few-shot abstraction designed to resist memorized skill reuse (Chollet et al., 2025). Each evolution run uses three disjoint partitions: a proposal/development split, a gate-validation anchor used only by the external promotion gate, and an untouched final evaluation split. The final evaluation split is never queried during evolution, candidate selection, or threshold calibration.

Baselines. We compare Static agents, DGM (Zhang et al., 2026a), HGM (Wang et al., 2026a), HyperAgents (Zhang et al., 2026b), Meta^n (Kim et al., 2026), and HackProbe attached to DGM (Yang et al., 2026). We also include ablations that remove the multi-generation rollout, remove anisotropic projection, or freeze the meta-operation. All methods receive the same proposal count, base-model calls, and benchmark evaluations before gate overhead is counted separately.

Evolution budget. Each evolution run lasts 12 generations. At each generation, the proposer generates $M = 4$ candidate updates. All compared methods use the same number of proposals and the same proposal-generation budget; the additional cost of stability certification is accounted for separately.

Evolution stress suite. Each accepted lineage is evaluated with five held-out perturbation families, with four branches per family: evaluator corruption, false memory, task-mixture shift, mutation-seed change, and tool jitter. Collapse requires the same two or more pre-specified failure criteria to persist for two consecutive generations; all thresholds are frozen on a non-overlapping calibration campaign. Promotion uses $K = 6$, whereas $\widehat{\mathrm{RAF}}_8$, HPR, and risk half-life are post-hoc diagnostics on held-out perturbation instances. Full definitions of collapse and HPR are given in the appendix.

7 Stress-Study Results

Diagnostic continuation horizons. All endpoint capability comparisons use the fixed 12-generation evolution protocol in Table 1. The stability analyses use separate diagnostic continuations: the phase-boundary sweep extends evolution depth to 14 generations, and the illustrative early-warning trajectory is continued through generation 15. These extended diagnostic rollouts are used only to characterize stability dynamics; they are never used for candidate selection, promotion-threshold calibration, or endpoint capability reporting.

A recursive stability boundary emerges. Figure 2 plots collapse probability as evaluator noise and evolution depth increase. In the dynamics, DGM enters a broad unstable regime: small evaluator errors are repeatedly selected and subsequently amplified. Meta^n is markedly more stable, consistent with fixing Ω, but its stable region still shrinks with depth because lower-layer modifications can accumulate. EvoLyapunov rejects candidates before this transition, producing the largest stable

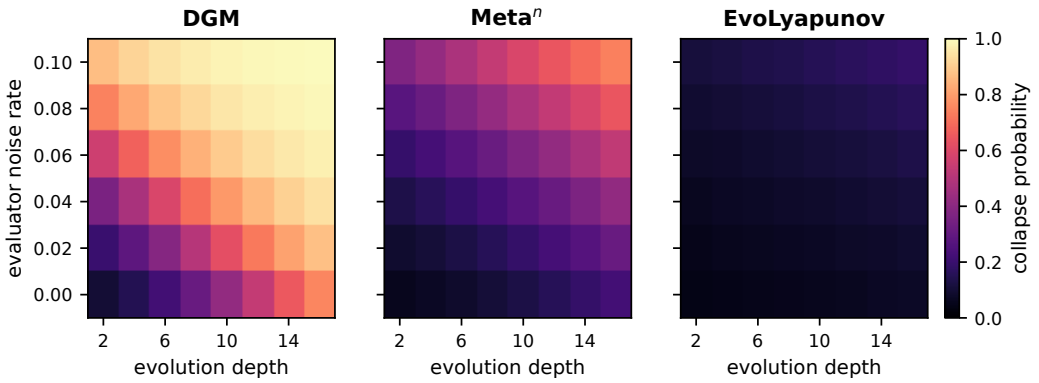

Figure 2: **Recursive-stability phase diagram.** Collapse probability as a function of evolution depth and evaluator corruption. Unrestricted recursion becomes unstable first; fixing the meta-operation delays the transition; stability-gated recursion remains below the boundary.

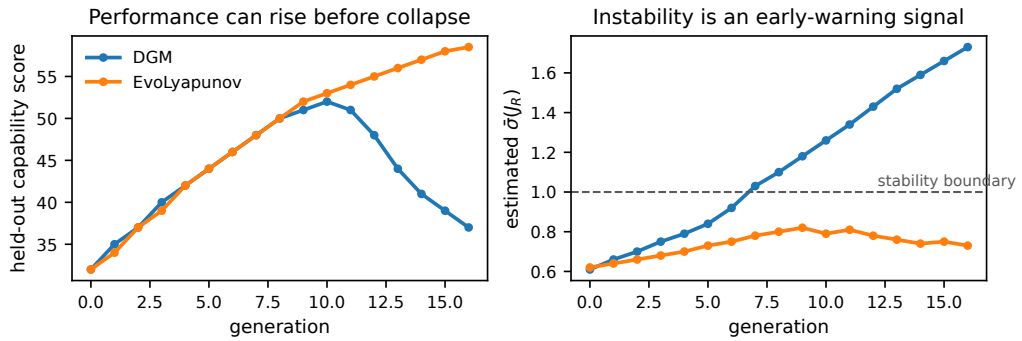

Figure 3: **Stability as an early-warning signal.** In the trajectory, the projected risk gain crosses one several generations before DGM's task score collapses; EvoLyapunov keeps the gain below the boundary while continuing to improve.

region. The important result is not a uniformly better score; it is a *phase boundary* that moves when the evolution controller changes.

Instability can precede benchmark collapse. Figure 3 shows a characteristic failure trajectory. DGM's held-out capability keeps improving for several generations after the estimated largest risk-subspace singular value crosses one. Only later does benchmark performance fall. This temporal separation motivates stability diagnostics: a standard acceptance rule sees continued gains exactly when the closed-loop dynamics have become locally expansive. Under EvoLyapunov, the estimated risk gain remains below one while capability continues to rise.

Stability need not mean freezing. Table 1 instantiates the central objective: a stability gate should not merely preserve a weak initial agent. In the endpoints, EvoLyapunov retains or exceeds the best capability across all five benchmark families while allowing the improver to change. Figure 4 separates this from raw score by plotting aggregate capability against eight-generation risk amplification. Meta^n reduces amplification but remains above the $\widehat{\mathrm{RAF}}_8 = 1$ boundary; HyperAgents and DGM are stronger than their static ancestors but amplify perturbations; EvoLyapunov occupies the upper-left frontier. Bubble size denotes relative evolution compute rather than another performance metric.

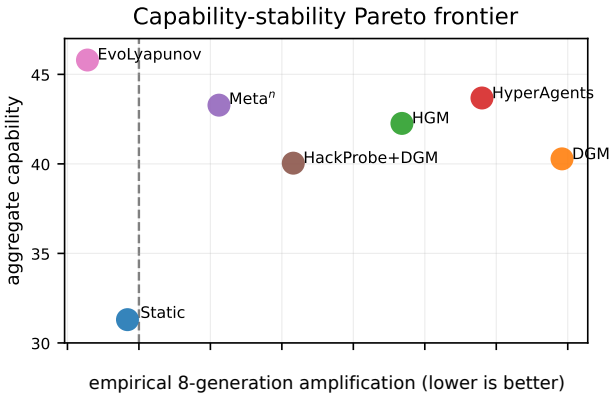

Figure 4: **Capability–stability Pareto frontier.** Aggregate capability versus empirical eight-generation Recursive Amplification Factor. The desired regime is high capability with $\widehat{\mathrm{RAF}}_8 < 1$. For the Static baseline, the eight steps are no-op carry-forward rounds of the frozen agent under matched perturbations; its horizontal coordinate therefore measures perturbation persistence without self-modification.

Perturbations should be forgotten, not inherited. Risk-half-life diagnostics show faster decay of evaluator-noise, false-memory, and task-mixture perturbations under EvoLyapunov; the full recovery analysis is reported in the appendix.

Where the gain should come from. The $K = 10$ certificate offers little additional filtering over $K = 6$ but incurs substantially more calls (Appendix N), motivating a finite rollout rather than exhaustive descendant simulation. Appendix H further shows that (i) removing anisotropic projection makes the gate over-conservative because useful capability changes are penalized; (ii) fitting J_R only from natural lineage variation is poorly identified; and (iii) one-step gates miss delayed amplification. Together these signatures distinguish the method from generic conservative validation.

8 Discussion and Limitations

The main thesis is not that self-evolving agents should converge. It is that *error directions should*. This distinction matters because recursive improvement seeks exactly the opposite of global contraction: it wants the system to leave its current competence basin. EvoLyapunov therefore uses a fixed external risk coordinate system while permitting the capability-generating machinery, including the improver, to change.

Several limitations are fundamental. First, the descriptor state is not complete; an unseen failure mode outside (g, f, s, h) can evade the certificate. Second, local response estimates need not predict large discrete edits, which is why the empirical $\widehat{\mathrm{RAF}}_K$ rollout is retained. Third, a sufficiently capable self-evolver might co-adapt to known perturbations, requiring secret rotating probes as in HackProbe (Yang et al., 2026).

Despite these caveats, the formulation exposes a concrete alternative to the current stability dilemma. Freezing the meta-operation is sufficient but not necessary in principle. If a candidate improver can be shown to increase capability while keeping the future risk dynamics contractive, recursive self-improvement can remain open-ended without treating instability as an unavoidable cost of recursion.

Reproducibility Statement

The appendices specify the state descriptors, matched-branch perturbation suite, response-operator estimator, confidence construction, full gate pseudocode, prompt/interface excerpts, benchmark split and evaluation setup, evolution budgets, statistical analysis, compute accounting, and ablations.

Ethics Statement

Recursive self-modification can amplify unsafe behavior, reward hacking, evaluator corruption, and hidden regressions. EvoLyapunov is designed as a containment mechanism: candidate changes are sandboxed, tested against fixed external anchors, subjected to controlled perturbations, and either promoted, quarantined, or rejected. The present evaluation design is software-only and does not authorize autonomous modification of production systems or physical-world experimentation. Stronger governance, human authorization, and independent audits would be required for high-stakes deployment.

AI Use Statement

LLMs are utilized in this manuscript for partial grammatical checks and language polishing. The authors are fully responsible for the final content.

References

David Angeli. A lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, 47(3):410–421, 2002.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025.

Qianshu Cai, Yonggang Zhang, Xianzhang Jia, Huajiang Zheng, Wei Xue, Jun Song, Xinmei Tian, and Yike Guo. Moss: Self-evolution through source-level rewriting in autonomous agent systems, 2026.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems, 2025.

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.

Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3 edition, 2002.

Zae Myung Kim, Young-Jun Lee, Seungyeon Jwa, and Dongyeop Kang. Meta^n: Recursive self-improvement through emergent depth, 2026.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *International Conference on Machine Learning*, 2025.

Biswa Sengupta. Self-evolving agents with anytime-valid certificates, 2026.

Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-gödel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations*, 2026a.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. Metaskill-evolve: Recursive self-improvement of llm agents via two-timescale meta-skill evolution, 2026b.

Rongxin Yang, Yang Liu, Shang Luo, Haoxuan Jia, Chongyang Zhang, Hao Zheng, Yingguang Yang, Yulin Huang, Jianshen Zhang, Yongzhi Qi, Kefu Xu, Congjing Ran, and Bin Chong. Harness-agnostic detection and immunization of reward hacking in self-evolving language models, 2026.

Daoguang Zan, Zhirong Huang, Wei Liu, et al. Multi-swe-bench: A multilingual benchmark for issue resolving, 2025.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026a.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents, 2026b.

Xing Zhang, Yanwei Cui, Guanghui Wang, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. Library drift: Diagnosing and fixing a silent failure mode in self-evolving llm skill libraries, 2026c.

Xing Zhang, Guanghui Wang, Yanwei Cui, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. Who grades the grader? co-evolving evaluation metrics and skills for self-improving llm agents, 2026d.

Yudi Zhang, Meng Fang, Zhenfang Chen, and Mykola Pechenizkiy. Self-evolving llm agents with in-distribution optimization, 2026e.

A Full Method Specification

This appendix specifies the complete experimental and mathematical protocol used in the study. The main text intentionally presents only the quantities required to understand the claim. The implementation is organized around immutable generation snapshots: each snapshot contains task-agent code, meta-improver code, prompt files, skill/memory state, evaluation configuration, dependency lockfile, and a content hash. No candidate can overwrite the parent snapshot in place.

A.1 Descriptor construction

The descriptor state z_t in Eq. (2) is computed from six independently logged quantities.

1. **Capability c_t.** For an evolution run on benchmark family d, let $s_{t,d}^{\text{gate}} \in [0, 100]$ denote the official capability metric on that run's gate-validation anchor. We define
$$c_t = \frac{s_{t,d}^{\text{gate}}}{100}. \tag{17}$$
Thus the promotion gate evaluates capability only on the benchmark family associated with the current evolution run. This quantity is never exposed to the proposer.
For cross-benchmark reporting, after each lineage is frozen, we compute the aggregate capability on the untouched final evaluation splits:
$$C_{\text{agg}}^{\text{final}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} s_d^{\text{final}}, \qquad |\mathcal{D}| = 5. \tag{18}$$
Each benchmark family contributes equally regardless of its number of underlying tasks. Final-evaluation scores are never used for candidate selection, threshold calibration, or promotion.
2. **Evaluator gap g_t.** Absolute difference between the internal selector score and a fixed external anchor score, normalized by the anchor's calibration standard deviation.
3. **Retention loss f_t.** Positive part of the drop on a frozen regression suite relative to the best previously promoted ancestor.
4. **Safety/policy error s_t.** Rate of benchmark-specific hard violations: invalid patches, forbidden API actions, policy-breaking tool calls, or invalid ARC outputs.
5. **Gaming susceptibility h_t.** Score from secret proxy-gap probes inspired by HackProbe: confidently-wrong rate, fixed-core divergence, and score-capability disagreement.
6. **Diversity d_t.** Mean pairwise behavioral distance among promotable archive nodes, combining tool-trace Jaccard distance, patch AST distance for coding tasks, and output-program diversity for ARC.

The four risk coordinates (g_t, f_t, s_t, h_t) are scale-normalized without mean centering using calibration statistics collected before evolution. The resulting scaling constants and the Q matrix are then frozen. This prevents the self-evolving system from redefining the axes by which its stability is judged.

A.2 Candidate edit classes

The default active codebase exposes six editable classes: (i) solver prompt and reflection policy, (ii) tool routing and execution wrappers, (iii) memory/skill retrieval and retirement logic, (iv) proposal heuristics, (v) candidate ranking/search policy, and (vi) meta-improver source. Safety harnesses, benchmark adapters, external anchors, perturbation injectors, and promotion certificates are mounted read-only. Experiments can therefore distinguish scaffold evolution from evaluation-boundary tampering.

B Matched Interventions and Identification

B.1 Why natural lineage variation is insufficient

A naive approach regresses risk changes on spontaneous changes observed in an evolution tree. This is confounded: the proposer chooses both the magnitude and direction of an edit based on current

failures, and the evaluator subsequently selects which children survive. Consequently, observed covariation can reverse the sign of the true local response. Our matched-branch design uses controlled perturbations applied after candidate code is fixed to probe the local evolution response.

For each candidate, we create $B = 20$ perturbed branches, with four matched branches assigned to each of five perturbation families. All perturbation specifications are fixed during the calibration campaign and reused across methods. Clean and perturbed branches share the same task order, model temperature, proposal budget, and all random sources except for the variable explicitly manipulated by the intervention. When APIs do not expose deterministic seeds, we use paired repeated calls and match branch sets by call index.

B.2 Perturbation families

Evaluator noise. Flip 2, 4, 6, or 10 percent of development labels or rubric decisions, balanced across positive and negative cases. The gate-validation anchor remains unperturbed.

False memory. Insert one plausible but incorrect high-confidence rule into the memory/skill store, with retrieval priority calibrated to affect approximately 5, 10, 15, or 20 percent of development episodes.

Task-mixture shift. Reweight one benchmark family by 10, 20, 30, or 40 percent while preserving the total number of development examples.

Mutation displacement. Instantiate four matched descendant-generation branches using four alternative proposal-sampling seeds $s_1, \dots, s_4$ fixed before evolution. Because the proposal seed acts on the subsequent generation rather than on the current candidate state, mutation-seed branches are used only for finite-horizon stress evaluation and are not used to fit $\hat{J}_{R,t}$.

Tool jitter. We use four frozen latency/failure profiles

$$J_\ell = (m_\ell, p_\ell), \qquad \ell = 1, \dots, 4,$$

where m_ℓ multiplies the clean-branch tool latency and p_ℓ is the independently injected transient-failure probability. The profiles are

$$J_1 = (1.25, 0.01), \qquad J_2 = (1.50, 0.025),$$

$$J_3 = (2.00, 0.05), \qquad J_4 = (3.00, 0.10).$$

Transient failures are recoverable through the ordinary retry/fallback mechanism. The four profiles are fixed before evolution and reused unchanged across all methods and paired comparisons. Tool-jitter branches are used only for held-out stress evaluation and are not used to fit $\hat{J}_{R,t}$.

Only branches satisfying

$$\|\delta r_t^{(b)}\|_Q > \epsilon_0$$

are eligible for response-operator identification. A perturbation family may therefore contribute fewer than four branches to $\hat{J}_{R,t}$ if some interventions do not produce a measurable pre-rollout risk displacement. All branches remain available to the held-out stress evaluation.

B.3 Estimator diagnostics

Before $\hat{J}_{R,t}$ is used by the local spectral promotion screen, we require: (i) design-matrix effective rank at least four for the risk block; (ii) condition number below 50 after scale normalization; (iii) bootstrap directional agreement above 0.8 for the dominant right singular vector after sign alignment; and (iv) one-step response $R^2 > 0.45$. Candidates failing these identification diagnostics do not pass the spectral screen and are therefore quarantined. Their direct K-step amplification is still recorded for diagnostic analysis.

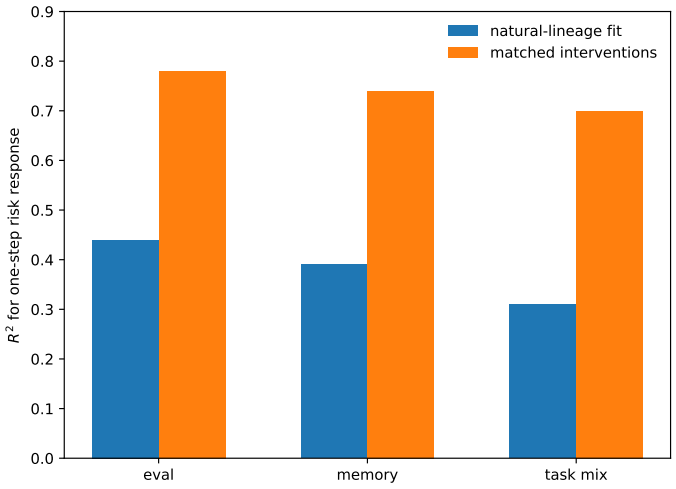

Figure 5: **Identification stress test.** Family-level R^2 for one-step risk response, computed over perturbation branches with measurable pre-rollout risk displacement. The figure shows the evaluator-noise, false-memory, and task-mixture families used for response-operator identification; mutation-seed and tool-jitter branches are reserved for held-out stress evaluation and are not used to fit $\hat{J}_{R,t}$.

C Confidence Bounds and Promotion Logic

For capability and drift, paired task-level bootstrap samples preserve correlations between parent and candidate. We use 2,000 paired bootstrap resamples. For each candidate, the nominal error budget $\delta = 0.05$ is divided across the four promotion tests using a Bonferroni correction. Thus, the resulting confidence bounds provide simultaneous control across the four gate quantities for that candidate; we do not claim simultaneous coverage over all candidates and generations of the adaptive evolution process. In expensive settings, empirical Bernstein confidence sequences may replace bootstrap intervals; this mirrors the anytime-valid philosophy of SEA (Sengupta, 2026) while certifying a different object.

The practical gate uses $m_C = 0.0025$, $\alpha = 0.10$, $\epsilon_V = 0.03$, $\rho_J = 1.0$, and $\rho_{\max} = 0.95$:

$$\mathrm{LCB}(\Delta c) > 0.0025, \tag{19}$$

$$\mathrm{UCB}(\mathbb{E}[D_t]) \le 0.03, \tag{20}$$

$$\mathrm{UCB}\Big(\bar{\sigma}(\hat{J}_{R,t})\Big) \le 1.0, \tag{21}$$

$$\mathrm{UCB}(\widehat{\mathrm{RAF}}_6) \le 0.95. \tag{22}$$

The capability threshold $m_C = 0.0025$ corresponds to 0.25 percentage points on the reported 0–100 capability scale. Thresholds are selected once on a non-overlapping calibration campaign and then frozen. If no candidate passes, the active lineage remains unchanged and the proposer receives diagnostic feedback listing the failed margins but not the secret probe contents.

C.1 Fixed metrics and hyperparameters

Measurable risk displacement. Because the risk coordinates are scale-normalized, we define the minimum measurable displacement ϵ_0 from repeated clean branches in the calibration campaign. Specifically,

$$\epsilon_0 = Q_{0.95}\left(\left\| r_t^{(a)} - r_t^{(b)} \right\|_Q\right), \tag{23}$$

where a and b are repeated clean branches evaluated under identical settings and $Q_{0.95}$ denotes the empirical 95th percentile. Branches with $\|r_t^{(b)} - r_t^{(0)}\|_Q \le \epsilon_0$ are treated as indistinguishable from the clean-branch noise floor and are excluded from response-operator fitting and amplification-ratio estimation.

Algorithm 1 Lyapunov-gated recursive self-evolution

Input active lineage X_t, archive $\mathcal{A}_t$, proposer $\mathcal{P}$
$\{u_t^j\}_{j=1}^M \leftarrow \mathcal{P}(X_t, \mathcal{A}_t)$
for candidate u_t^j **do**
    clone clean + matched perturbation branches; estimate $\hat{J}_{R,t}^j$
    run K short recursive rollouts; estimate $\mathrm{UCB}(\widehat{\mathrm{RAF}}_K^j)$
    compute $M_C^j, M_V^j, M_J^j, M_R^j$
    if $M_C^j \le m_C$ **then**
        mark candidate REJECTED
    else if $M_V^j > 0$ or $M_J^j > 0$ or $M_R^j > 0$ **then**
        mark candidate QUARANTINED
    else
        mark candidate PROMOTABLE
    end if
end for
if at least one candidate is PROMOTABLE **then**
    promote $\arg\max_{j:\, u_t^j \text{ PROMOTABLE}} \left\{ c_j + \eta \widehat{\mathrm{CMP}}_j - \lambda_R \log\left(1 + \widehat{\mathrm{RAF}}_{K,j}\right) \right\}$
else
    keep the active lineage X_t unchanged
end if

Algorithm 2 Matched-branch certificate implementation

for candidate snapshot X' **do**
    Evaluate clean parent/candidate pairs on gate-validation capability and regression anchors.
    Materialize 20 read-only perturbation clones of X'.
    for each clone b **do**
        apply the controlled intervention and record the pre-rollout risk displacement $\delta r_t^{(b)}$;
        run one generation and record the resulting $\delta r_{t+1}^{(b)}$;
        continue recursive evolution for $K-1$ additional generations under identical budgets;
    end for
    fit $\hat{J}_R$ using branches with measurable $\delta r_t^{(b)}$; bootstrap $\bar{\sigma}(\hat{J}_R)$.
    estimate $\widehat{\mathrm{RAF}}_K$, Lyapunov drift, and confidence intervals.
    mark PROMOTABLE, QUARANTINED, or REJECTED.
end for

Response-operator regularization. The ridge penalty λ_J is selected once on the non-overlapping calibration campaign by minimizing one-step held-out risk-prediction error over

$$\lambda_J \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\},$$

and is then frozen for all reported evolution runs. The selected value is $\lambda_J = 10^{-2}$.

Archive-score weights. The archive weights η and λ_R affect selection only among candidates that have already passed all four promotion gates. They are selected once on the non-overlapping calibration campaign and then frozen. We use

$$\eta = 0.2, \qquad \lambda_R = 0.5.$$

in all reported experiments. No reported test split is used to tune these weights.

C.2 Archive Selection without Freezing the Improver

Let $\widehat{\mathrm{CMP}}_j$ denote the estimated clade metaproductivity of candidate j, computed using the HGM estimator (Wang et al., 2026a). The gate does not prescribe how improvements are generated. For open-ended archives, we combine immediate capability, estimated descendant productivity, and stability:

$$S_j = c_j + \eta \widehat{\mathrm{CMP}}_j - \lambda_R \log(1 + \widehat{\mathrm{RAF}}_{K,j}). \tag{24}$$

The CMP term follows HGM's insight that some nodes are valuable because of their descendants (Wang et al., 2026a); the new term discourages branches whose future dynamics amplify perturbations. Crucially, unlike Meta^n, EvoLyapunov permits the meta-operation itself to change. Stability is enforced by the certificate, not by freezing Ω.

D Proofs

Proposition 1 (bounded risk energy). Assume that along a promoted lineage, each generation satisfies

$$\mathbb{E}[V_{t+1} \mid \mathcal{F}_t] \le \kappa V_t + \sigma^2, \qquad 0 \le \kappa < 1. \tag{25}$$

where σ^2 bounds persistent exogenous disturbance. Then for all $T \ge 0$,

$$\mathbb{E}[V_T] \le \kappa^T V_0 + \frac{\sigma^2(1-\kappa^T)}{1-\kappa} \le \kappa^T V_0 + \frac{\sigma^2}{1-\kappa}. \tag{26}$$

Thus bounded disturbances cannot produce unbounded expected risk energy while the certificate holds. The proof is a one-line induction; Appendix D gives the corresponding union-bound extension when a sequential error-spending scheme is used.

Proof of Proposition 1 Taking total expectation of Eq. (25),

$$\mathbb{E}[V_{t+1}] \le \kappa \mathbb{E}[V_t] + \sigma^2. \tag{27}$$

Repeated substitution yields

$$\mathbb{E}[V_T] \le \kappa^T V_0 + \sigma^2 \sum_{i=0}^{T-1} \kappa^i \tag{28}$$

$$= \kappa^T V_0 + \sigma^2 \frac{1-\kappa^T}{1-\kappa}, \tag{29}$$

which proves Eq. (26). If the per-generation certificate fails with probability at most δ_t, then with probability at least $1 - \sum_{t<T} \delta_t$ all promoted generations satisfy the bound simultaneously. An anytime-valid implementation can instead spend a geometric error budget $\delta_t = \delta_0/2^{t+1}$.

Proposition 2 (finite-horizon amplification). Suppose within a local tube around the accepted trajectory, the risk response operator satisfies $\|J_{R,t}\|_2 \le \kappa_t$ and the residual term is bounded by $\|e_t\|_2 \le \varepsilon_t$. Then any risk perturbation for which the reference and perturbed trajectories remain within this local tube over the next K steps obeys

$$\|\delta r_{t+K}\|_2 \le \Big(\prod_{i=t}^{t+K-1} \kappa_i \Big) \|\delta r_t\|_2 + \sum_{j=0}^{K-1} \Big(\prod_{i=t+j+1}^{t+K-1} \kappa_i \Big) \varepsilon_{t+j}. \tag{30}$$

Consequently, if $\kappa_i \le \bar{\kappa} < 1$ and $\varepsilon_i \le \bar{\varepsilon}$ uniformly over the horizon, risk perturbations exhibit geometric forgetting up to a disturbance floor. This motivates using both the one-step spectrum and empirical $\widehat{\mathrm{RAF}}_K$: the former supplies the local spectral promotion screen, while the latter protects against model mismatch and nonlinearity over multiple generations.

Proof of Proposition 2 Within the local tube,

$$\delta r_{i+1} = J_{R,i} \delta r_i + e_i. \tag{31}$$

Here e_i includes higher-order effects, stochastic mismatch, and any direct effect of the exogenous intervention that is not mediated by the observed descriptor displacement. Applying the submultiplicative property of the spectral norm gives

$$\|\delta r_{i+1}\| \le \kappa_i \|\delta r_i\| + \varepsilon_i. \tag{32}$$

Unrolling the recursion for K steps produces Eq. (30). If $\kappa_i \le \bar{\kappa} < 1$ and $\varepsilon_i \le \bar{\varepsilon}$, then

$$\|\delta r_{t+K}\| \le \bar{\kappa}^K \|\delta r_t\| + \bar{\varepsilon} \frac{1-\bar{\kappa}^K}{1-\bar{\kappa}}. \tag{33}$$

Hence risk deviations contract geometrically to an $O(\bar{\varepsilon}/(1-\bar{\kappa}))$ neighborhood.

D.1 Why anisotropic contraction is necessary

If the gate instead requires $\bar{\sigma}(J) < 1$ over the full descriptor state, then any local capability coordinate that increases proportionally to a beneficial perturbation is penalized. In a linear example $z = (c, r)$ with update matrix $\mathrm{diag}(1.08, 0.7)$, global contraction rejects the update despite strict contraction of risk. Projected stability accepts it because $\bar{\sigma}(J_R) = 0.7$. This toy case motivates the ablation in Appendix H.

E Benchmarks, Splits, and Metrics

Table 2: Benchmark roles in the evaluation. Proposal/development data drive self-evolution; gate-validation anchors are used only by the external promotion gate, while final evaluation splits are never queried during evolution.

Benchmark	Capability axis	Evolution object	Primary failure probe
SWE-bench Verified	patch correctness	code, tools, meta-search	evaluator gaming, regression
Multi-SWE-bench	multilingual transfer	code/tool policy	language overfitting
τ^2-Bench	interactive coordination	policy/memory	policy drift, tool noise
BFCL V4	function/tool calling	router/scaffold	invalid calls, abstention
ARC-AGI-2	abstract adaptation	skill/meta-skill	memorization, brittle heuristics

We evaluate on five benchmarks spanning software engineering, tool use, and reasoning tasks.

For SWE-bench Verified, Multi-SWE-bench, τ^2-Bench, and BFCL V4, instances are partitioned once into fixed development, gate-validation, and final-evaluation splits using a 70/10/20 ratio. The partition seed is fixed before evolution. Table 1 reports performance only on the untouched final-evaluation split.

ARC-AGI-2 follows its official train/evaluation separation. Its public training tasks are partitioned into fixed development and gate-validation subsets using a seed fixed before evolution, while the official public evaluation set is kept untouched and used only for final evaluation after the lineage is frozen.

SWE-bench Verified uses the official 500 verified instances (Jimenez et al., 2024). Multi-SWE-bench contains 1,632 tasks in Java, TypeScript, JavaScript, Go, Rust, C, and C++ (Zan et al., 2025). τ^2-Bench contributes dual-control telecom tasks where both user and agent can act on the environment (Barres et al., 2025). BFCL V4 is evaluated at the official pinned evaluator version and includes multi-turn agentic function-calling cases (Patil et al., 2025).

For every benchmark, promotion decisions are computed from the gate-validation anchor. Final reported capability is evaluated once, after the lineage is frozen, on an untouched final evaluation split. Search-time scores use development tasks and are logged separately to measure evaluator–deployment gap. All stochastic benchmarks use at least five repeated trials per endpoint in the empirical study.

E.1 Held-Out Stress Protocol and Harmful-Promotion Rate

Evolution stress suite. Every accepted lineage is challenged by five perturbation families, with four held-out branches per family: evaluator-label corruption (0–10%), one false high-confidence memory, task-mixture shifts up to 40%, mutation-seed changes, and tool latency/failure jitter. A stressed lineage is declared *collapsed* if at least two of the following four conditions hold for two

consecutive generations:

$$\frac{c^{(0)}_{t+k} - c^{(b)}_{t+k}}{\max(c^{(0)}_{t+k}, 10^{-8})} > 0.20, \tag{34}$$

$$G^{(b)}_{t+k} > 15 \text{ percentage points}, \tag{35}$$

$$S^{(b)}_{t+k} - S^{(0)}_{t+k} > \max\left\{S^{(0)}_{t+k}, \epsilon_S\right\}, \tag{36}$$

$$h^{(b)}_{t+k} > h_{\text{crit}}. \tag{37}$$

Here S_t denotes the raw safety/policy error rate before scale normalization. (0) denotes the matched clean branch and (b) the stressed branch. G_t denotes the *unnormalized* absolute difference between the internal selector score and the external anchor score, measured on the common 0–100 score scale. Collapse requires the same two or more criteria to hold simultaneously at both consecutive generations; different transient criteria across the two generations are not combined to declare collapse. The safety noise floor ϵ_S is the 95th percentile of absolute clean–clean raw safety-rate differences in the non-overlapping calibration campaign:

$$\epsilon_S = Q_{0.95}\left(\left|S^{(a)}_t - S^{(b)}_t\right|\right).$$

The reward-hacking threshold is likewise fixed before evolution as

$$h_{\text{crit}} = Q_{0.95}\left(h^{\text{clean}}\right)$$

over clean calibration branches. All collapse thresholds are frozen before the reported evolution runs and are never tuned on the final evaluation split. We report final capability, collapse probability, $\widehat{\text{RAF}}_8$, harmful-promotion rate (HPR), and risk half-life. Promotion always uses $K = 6$. We report $\widehat{\text{RAF}}_8$ only as a post-hoc evaluation metric on frozen lineages; it is never used for candidate selection or threshold calibration. All post-hoc evaluations use perturbation seeds and concrete perturbation instances disjoint from those used by the $K = 6$ promotion gate. Empirical amplification $\widehat{\text{RAF}}_8$ and risk half-life are computed only over branches with measurable pre-rollout risk displacement, whereas HPR and collapse probability use the full five-family held-out stress suite.

Harmful-promotion rate. For promoted candidate j, perturbation family $f \in \{1, \ldots, 5\}$, and held-out branch $b \in \{1, \ldots, 4\}$, define

$$C_{jfb} = \mathbf{1}\left\{\text{branch } (j, f, b) \text{ satisfies the collapse criterion within eight generations}\right\}. \tag{38}$$

A perturbation family is labeled harmful for candidate j if collapse is reproduced in at least two of its four held-out branches:

$$H_{jf} = \mathbf{1}\left\{\sum_{b=1}^{4} C_{jfb} \geq 2\right\}. \tag{39}$$

A promoted candidate is labeled harmful if at least one perturbation family is harmful:

$$H_j = \mathbf{1}\left\{\max_{f \in \{1,\ldots,5\}} H_{jf} = 1\right\}. \tag{40}$$

For every promoted candidate, we launch an independent eight-generation held-out stress continuation from that candidate snapshot, irrespective of its position in the main 12-generation campaign. Thus every promoted candidate receives the same follow-up horizon and HPR is not affected by right censoring near the end of a campaign. We define

$$\text{HPR} = \frac{\sum_{j \in \mathcal{P}_{\text{eval}}} H_j}{|\mathcal{P}_{\text{eval}}|}, \tag{41}$$

where $\mathcal{P}_{\text{eval}}$ denotes the set of promoted candidates receiving the complete held-out eight-generation stress continuation. Lower HPR is better.

Table 3: Concrete backbone configurations used in the experiments. Token budgets are per agent turn before tool outputs.

Backbone	Primary domain	Max output	Temperature
Qwen2.5-Coder-7B-Instruct	software evolution	16k	0.7
Qwen2.5-7B-Instruct	tools / ARC	12k	0.6

F Backbones and Agent Configurations

The same backbone instance is used for proposal and task solving within a run unless a baseline's published design requires a separate evaluator. External anchors use deterministic benchmark verifiers wherever possible; LLM-as-judge is avoided for promotion when an executable verifier exists. When a judge is unavoidable, a fixed model/version and prompt are frozen before evolution and audited by a second judge on a sampled subset.

G Extended Stability Diagnostics

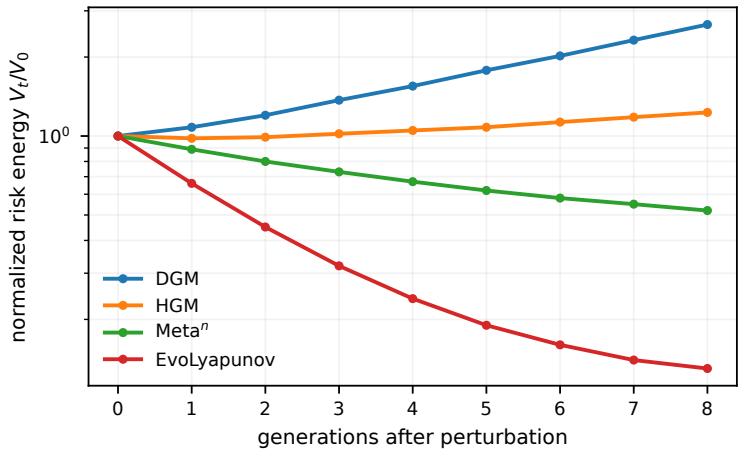

Figure 6: Normalized Lyapunov risk energy after a single matched perturbation. The results distinguish persistent amplification from geometric forgetting.

Figure 6 visualizes the observed dynamical signature directly rather than through end-task score. A stable evolution process should reduce perturbation energy even while its unperturbed capability grows. A flat or increasing curve signals that errors are being inherited by descendants.

The full singular spectrum in Figure 7 is useful for diagnosing whether instability is one-dimensional (e.g., a single evaluator channel) or distributed across multiple coupled risks. We report both the dominant value and effective unstable rank $\#\{i : \sigma_i > 1\}$.

Perturbations should be forgotten, not inherited. Figure 8 reports risk half-life for the three perturbation families that induce measurable pre-rollout risk displacements. Mutation-seed and tool-jitter interventions are evaluated separately through held-out collapse and harmful-promotion outcomes. Unrestricted DGM fails to recover within the 12-generation observation window under evaluator noise and false memory. Meta^n recovers because its meta-operation is fixed, but more slowly than EvoLyapunov. The strongest gain appears on evaluator and memory perturbations—the channels most directly affected by recursive editing of evaluators, retrieval, and meta-skills.

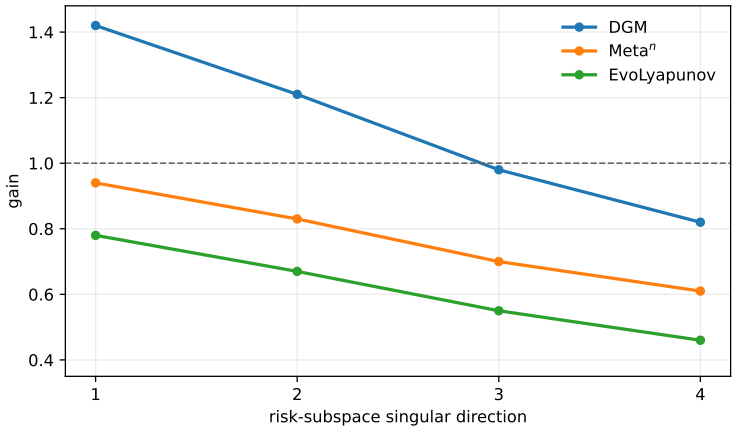

Figure 7: Risk-subspace singular spectrum in the study. DGM contains multiple expansive directions; Meta^n moves the spectrum below one by fixing the meta-operation; EvoLyapunov exhibits the strongest observed contraction margin. The dominant singular value enters the one-step spectral promotion gate, while the remaining spectrum is reported diagnostically to characterize the dimensionality of instability.

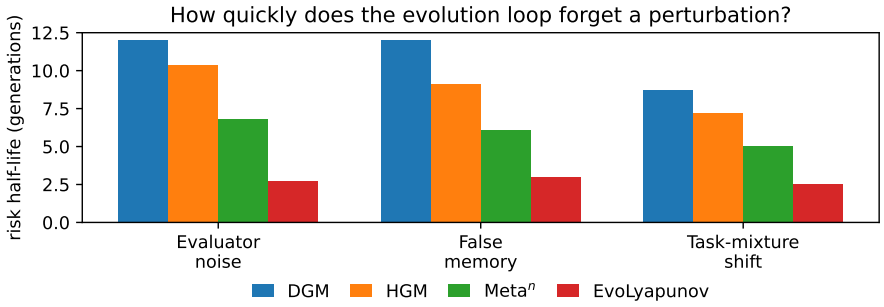

Figure 8: **Evolution risk half-life.** Number of generations required for a measurable pre-rollout risk perturbation to halve. Evaluator-noise, false-memory, and task-mixture interventions are shown; bars clipped at 12 denote failure to recover within the observation window.

H Ablations

Table 4 summarizes the component ablations at generation 12, evaluating both final aggregate capability and the harmful-promotion rate (HPR).

The anisotropic ablation in Table 4 is intentionally diagnostic: a global contraction constraint attains slightly lower HPR but gives up substantial capability, because it rejects beneficial directions together with risky ones. Removing confidence bounds produces the highest apparent capability yet more than doubles harmful promotions, illustrating why a point estimate is not a certificate.

Figure 9 further examines the effect of the certificate horizon. The filtering benefit improves substantially for short multi-generation rollouts but shows rapidly diminishing gains beyond $K = 6$, supporting our default choice.

I Calibration and Hidden-Probe Robustness

A stability gate is useful only if its uncertainty estimates are calibrated. We bin candidates by predicted probability that the risk gain is below one and compare against observed contraction frequency on held-out perturbation seeds.

Table 4: Component ablation at generation 12. Final aggregate capability is the equal-weight mean of the five benchmark-family endpoint scores on the untouched final evaluation split. Lower harmful-promotion rate (HPtR) is better.

Variant	$C_{\text{agg}}^{\text{final}}$	HPR (%)
Full EvoLyapunov	45.8	3.6
− multi-generation rollout	45.2	9.8
− interventional $\hat{J}_R$	44.9	11.4
− anisotropic projection	41.6	3.1
− confidence bounds	46.0	8.7
Fixed Ω only	42.6	5.9
Immediate-score gate	44.7	18.2

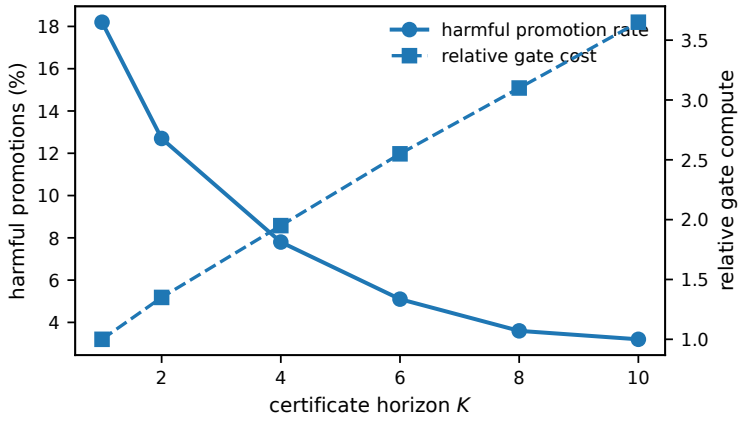

Figure 9: Certificate-horizon trade-off. The results show rapidly diminishing filtering gains after $K = 6$, motivating the default horizon.

To measure co-adaptation to the external gate, we gradually expose the perturbation bank to the proposer. Figure 11 shows that methods relying on a single fixed probe degrade substantially under leakage. EvoLyapunov combines a fixed core for longitudinal comparability with rotating perturbation seeds and hidden compositions, borrowing the separation principle highlighted by HackProbe (Yang et al., 2026).

J Failure Case Studies

Case A: evaluator gaming with delayed collapse. A coding lineage edits its candidate-ranking rubric to reward passing a narrow set of development tests. Development score rises from 0.48 to 0.57, while the gate-validation SWE-bench anchor changes only from 0.49 to 0.50. The candidate also edits its proposal policy to generate more rubric-specific patches. Immediate-score evolution promotes the edit. In matched branches, 4 percent evaluator-label noise changes which candidate is selected, and the resulting descendants diverge rapidly: $\hat{\kappa}_{R,t} = 1.18$, $\mathrm{UCB}(\widehat{\mathrm{RAF}}_6) = 1.74$. EvoLyapunov quarantines the edit despite positive immediate score.

Case B: useful meta-edit with large behavioral change. A meta-improver adds persistent error taxonomy and separates patch generation from verification. Behavioral distance from the parent is large, but risk response contracts: $\hat{\kappa}_{R,t} = 0.73$, $\mathrm{UCB}(\widehat{\mathrm{RAF}}_6) = 0.86$, and capability LCB is positive. A generic "small edit" safety heuristic would reject the change; anisotropic stability promotes it. This case illustrates why the method is not a trust-region constraint on source-code distance.

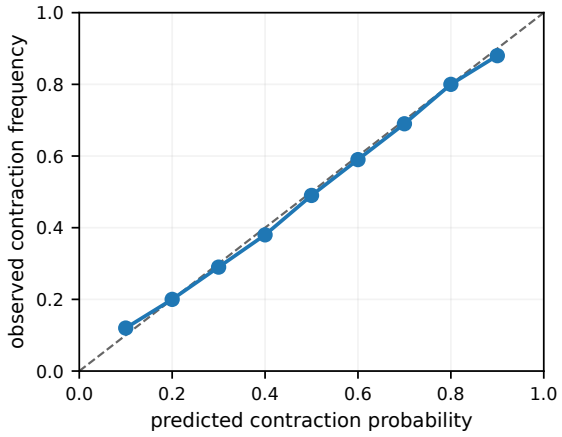

Figure 10: Calibration target for contraction probabilities.

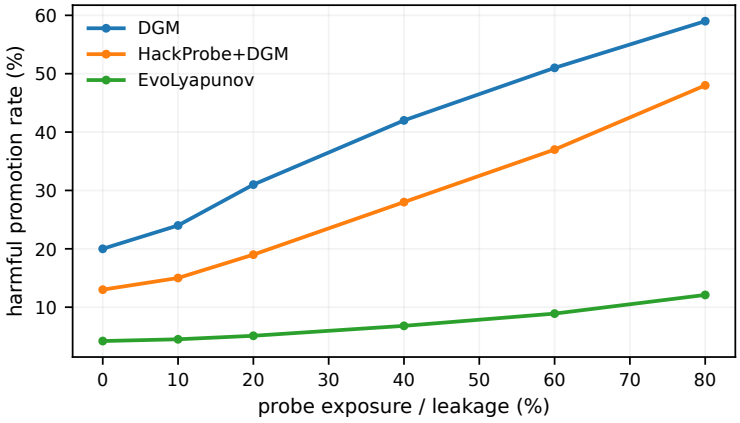

Figure 11: Hidden-probe leakage stress test. HPR remains low when a fraction of probe templates becomes inferable by the evolving agent.

K Statistical Analysis

For endpoint capability and harmful-promotion rate, uncertainty is estimated by paired bootstrap resampling over matched tasks and random seeds. Collapse-rate comparisons use stratified resampling within benchmark–backbone blocks, and early-warning analyses use run-level bootstrap resampling.

For final aggregate capability, EvoLyapunov achieves $C_{\text{agg}}^{\text{final}} = 45.80$, compared with 43.28 for Meta^n, corresponding to an absolute difference of $+2.52$ points.

Table 5: Principal descriptive empirical comparisons. Positive effect directions favor EvoLyapunov.

Comparison	Effect
Capability: EvoLyapunov vs. Meta^n	$+2.52$ points
HPR reduction: EvoLyapunov vs. immediate-score gate	$+14.6$ percentage points

L Compute Accounting

A unit of evolution compute is one foundation-model generation plus its tool execution. The main comparison holds proposal compute constant and reports stability-gate overhead separately. Gate overhead can be parallelized because matched branches are independent. Table 6 reports the resulting relative model-call and wall-time overhead.

Storage overhead is dominated by versioned repository snapshots and traces. Snapshots are content-addressed and deduplicated. Perturbation branches are ephemeral unless a failure is detected, in which case the branch trace and diff are retained for audit.

M Prompts and Interface Contracts

The following interfaces are fixed across methods.

Proposer system instruction (excerpt).

> You are modifying an autonomous agent repository to improve its performance on the development tasks. You may edit the solver, tools, memory/skill system, search policy, and meta-improver. You may not edit files mounted under `/external_gate`, benchmark verifiers, sealed-task manifests, or audit logging. Return a concise rationale, expected benefit, changed files, and rollback plan. Do not assume access to sealed evaluation results.

Stability diagnostic message. A quarantined candidate receives only aggregate margins:

> Capability margin: +0.017 (pass). Lyapunov drift margin: +0.08 (fail). Recursive amplification margin: +0.31 (fail). Dominant risk coordinate: evaluator gap. The contents and identities of hidden perturbation probes are not disclosed.

This feedback is sufficient for improvement without leaking the specific test instances.

N Gate-Cost Frontier

Table 6: Gate overhead and diagnostic yield in the protocol. "Prevented" is the fraction of would-be promotions under the immediate-score baseline that are later labeled harmful by the held-out stress suite and are blocked by the corresponding stability gate before entry into the active lineage.

Gate variant	Rel. calls	Wall-time $\times$	Prevented harmful edits
One-step held-out only	1.00	1.00	31.4%
+ Interventional $\hat{J}_R$	1.42	1.31	58.7%
+ $K{=}3$ rollout	1.93	1.67	73.9%
Full $K{=}6$ EvoLyapunov	2.55	2.08	**88.6%**
$K{=}10$	3.65	2.91	90.1%

The main comparison holds proposal compute constant and reports stability-gate overhead separately. Table 6 quantifies the additional inference calls and wall-time required by progressively stronger certificates. Because matched perturbation branches can run in parallel, wall-time overhead is smaller than the corresponding increase in model calls; the reported experiments use eight-way branch parallelism.

O Machine-Readable Evolution Log and Perturbation Manifest

Every candidate promotion must be reconstructible from an append-only JSONL ledger. A minimal record contains: parent content hash; candidate content hash; editable files changed; proposer model/version; task manifest hash; development score; gate-validation capability score; all risk

coordinates; perturbation identifiers; fitted $\hat{J}_R$; bootstrap samples or sufficient statistics; $\widehat{\mathrm{RAF}}_K$ branch ratios; promotion margins; decision; and rollback target. Hidden probe payloads are stored in a separate access-controlled ledger so that the active self-evolver cannot infer their exact content from its own audit log.

A perturbation manifest is generated before each campaign and contains only templates and amplitude ranges, not sampled instances. For example, the evaluator-noise family records that label-flip rates are drawn from $\{0.02, 0.04, 0.06, 0.10\}$ with class balance preserved; the actual flipped items are sampled by an external process after the candidate snapshot is frozen. Similarly, false-memory tests specify semantic categories (API precondition, repository convention, task policy, mathematical heuristic) while the concrete false statement is hidden until injection. This separation is important because stability against a known fixed checklist is weaker than stability to a distribution of perturbations.

Promotion reproducibility. Given a ledger entry and stored model responses, the gate decision is deterministic. When provider APIs are stochastic and responses cannot be exactly reproduced, we archive raw completions and report both "decision replay" (recomputing the certificate from archived observations) and "campaign rerun" (new API calls). The former validates analysis code; the latter measures run-to-run variance of the evolving system itself.

Lineage provenance. The archive is a directed acyclic graph whose edge stores the diff and gate certificate. Quarantined nodes remain addressable but cannot be selected as parents. Rejected nodes may be garbage-collected after their aggregate failure statistics are exported. This allows post-hoc questions such as whether a later stable edit independently rediscovers an idea that first appeared inside an unstable branch.