SEVA-LYAP: SOLVER-AWARE DATA EVOLUTION AND VERIFICATION-GUIDED TEST-TIME ADAPTATION FOR SYMBOLIC LYAPUNOV FUNCTION DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Lyapunov functions provide fundamental certificates for the stability of dynamical systems, yet discovering explicit symbolic Lyapunov functions remains a long-standing computational challenge due to the vast search space and strict mathematical constraints. Recently, symbolic transformers have advanced this direction by formulating Lyapunov discovery as a sequence-to-sequence translation task, directly predicting interpretable Lyapunov functions from system dynamics. Despite this pioneering advance, these models face two critical bottlenecks: they rely on static synthetic datasets fixed prior to training, and they lack mechanisms to adapt at test time. Consequently, rare regions of the system distribution can remain underrepresented, making novel or hard systems difficult to solve. To address these limitations, we introduce SEVA-Lyap, a unified framework integrating solver-aware data evolution with verification-guided test-time adaptation. During training, SEVA-Lyap continuously evolves verified examples around the weaknesses of the current solver, creating a progressively challenging curriculum that strengthens reusable symbolic solving capability. At inference, SEVA-Lyap allocates additional test-time compute through targeted risk-seeking adaptation when direct prediction fails. Extensive evaluations demonstrate that SEVA-Lyap raises system-level verification success from 70.75% to 92.00% on polynomial systems and from 60.50% to 74.50% on non-polynomial systems, while reliably solving representative challenging dynamics and demonstrating notable transferability. Anonymous code and data are available at `https://anonymous.4open.science/r/SEVA-review-DBD6-0921`.

1 INTRODUCTION

Lyapunov functions are a cornerstone of dynamical-systems analysis and control, supporting stability certification, region-of-attraction estimation, and controller design (Khalil, 2002; Kellett, 2015; Vannelli & Vidyasagar, 1985; Sontag, 1989). Across robotics, autonomous vehicles, power networks, and other safety-critical applications, these functions provide mathematical guarantees that systems remain near or converge toward desired operating states, without requiring exhaustive trajectory simulation (Dawson et al., 2023; Dai et al., 2021; Ames et al., 2017; Cui & Zhang, 2022). Explicit symbolic Lyapunov functions are especially valuable because their symbolic form is interpretable, can reveal structural properties of the underlying dynamics, and can be independently checked under explicit verification criteria. However, automatically discovering such functions remains challenging because it requires searching a vast combinatorial space of symbolic expressions while enforcing strict mathematical constraints over continuous state spaces (Ahmadi, 2012).

Existing approaches have advanced symbolic Lyapunov discovery from different directions. Classical methods typically rely on structured candidate families or explicit computational constraints, providing strong mathematical guarantees in practice, but with limited scalability and expressiveness for higher-dimensional systems and complex non-polynomial dynamics (Prajna et al., 2002; Papachristodoulou & Prajna, 2005a;b; Ahmadi & Parrilo, 2011; Giesl, 2007; Peet, 2009; Ahmadi et al., 2011; Ahmadi & Khadir, 2020). Learning-based approaches relax these restrictions by representing Lyapunov candidates with neural networks (Richards et al., 2018; Chang et al., 2019; Boffi et al., 2021; Grüne, 2021), offering substantially greater flexibility. However, the resulting Lyapunov

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.

functions are typically represented implicitly by neural networks, obscuring their analytical structure and potentially increasing verification cost (Dai et al., 2021; Zhou et al., 2022; Grande et al., 2023; Wu et al., 2023; Yang et al., 2024). These limitations have motivated a shift toward explicit symbolic learning, supported by advances in symbolic mathematics and symbolic regression (Lample & Charton, 2020; Petersen et al., 2021; Kamienny et al., 2022; 2023). In particular, symbolic Transformers use mathematical construction rules to generate large-scale valid system–Lyapunov training pairs and learn a reusable mapping from system dynamics to interpretable Lyapunov candidates (Alfarano et al., 2024). However, these training examples are generated before solver learning and remain fixed throughout training, preventing the data from adapting to the solver's evolving weaknesses and potentially leaving difficult regions underrepresented. Moreover, once direct prediction fails, verification feedback is not used to further refine the unresolved prediction.

To address these challenges, we introduce SEVA-Lyap (**S**olver-Aware Data **E**volution and **V**erification-Guided Test-Time **A**daptation), a unified framework for symbolic Lyapunov function discovery. Instead of relying on a fixed synthetic dataset, SEVA-Lyap couples a Generator with the current Solver in a closed loop: the Generator is continually pushed toward valid systems that remain difficult for the Solver, while the Solver learns from these hard examples and, as it improves, reshapes the Generator's subsequent search. This interaction drives the training distribution toward the Solver's evolving weaknesses over successive training rounds and progressively strengthens a reusable cross-system Solver. When the resulting Solver still fails on a new system, SEVA-Lyap reuses its learned parameters and verification feedback to refine the prediction for that system, rather than restarting the search from scratch. The framework thus combines reusable knowledge accumulated across systems with targeted adaptation on difficult individual cases.

Experiments show that SEVA-Lyap consistently outperforms all baselines on polynomial and non-polynomial benchmarks. Specifically, solver-aware data evolution improves over the strongest Symbolic Transformer baseline, raising system-level verification success from 70.75% to 76.75% on polynomial systems and from 60.50% to 65.50% on non-polynomial systems. Verification-guided test-time adaptation further improves these rates to 92.00% and 74.50%, respectively, by refining systems that remain unresolved after direct prediction. Additional experiments show that the generated systems become progressively more difficult for a fixed reference solver over successive rounds of data evolution, confirming the intended increase in training-data difficulty. Tests on independently generated systems and representative challenging dynamics demonstrate meaningful transfer beyond the training distribution. Overall, SEVA-Lyap closes the loop between data generation, solver learning, and test-time refinement, turning solver failures into actionable feedback and enabling symbolic Lyapunov discovery to improve through cross-system learning and instance-level adaptation.

2 Related Work

Constraint-Based and Classical Lyapunov Synthesis. Classical computational approaches typically restrict Lyapunov synthesis to a parameterized candidate family and search for parameters satisfying the required stability inequalities. For polynomial dynamics, sum-of-squares (SOS) methods encode positivity and decrease conditions as polynomial nonnegativity constraints and solve the resulting semidefinite programs (Prajna et al., 2002; Papachristodoulou & Prajna, 2005a;b; Ahmadi & Parrilo, 2011). Nonlinear SMT solvers such as dReal provide complementary tools for checking nonlinear real-arithmetic constraints and searching for counterexamples to candidate Lyapunov functions (Gao et al., 2013). Constructive and approximation-based methods further exploit particular structural assumptions to derive or approximate Lyapunov functions (Giesl, 2007; Peet, 2009). These approaches provide strong guarantees under suitable representations, but their tractability depends on the candidate class, polynomial degree, verification domain, and system dimension; restrictive classes may also exclude valid Lyapunov functions, since globally asymptotically stable polynomial systems need not admit polynomial Lyapunov functions (Ahmadi et al., 2011).

Neural Lyapunov Functions. Learning-based approaches increase flexibility by representing Lyapunov candidates with neural networks (Richards et al., 2018; Boffi et al., 2021; Grüne, 2021). A common learner–verifier loop minimizes sampled Lyapunov-condition violations and feeds identified counterexamples back into subsequent optimization. Neural Lyapunov Control applies this paradigm to joint Lyapunov functions and controller learning (Chang et al., 2019), while FOSSIL 2.0 and LyZNet combine neural candidate construction with dedicated verification procedures

(Edwards et al., 2024; Liu et al., 2024). Subsequent work extends this framework to unknown dynamics, discrete-time settings, and broader feedback-control problems (Dai et al., 2021; Zhou et al., 2022; Grande et al., 2023; Wu et al., 2023; Yang et al., 2024; Liu et al., 2025; Dawson et al., 2023). However, these Lyapunov functions remain implicitly represented by the learned networks, obscuring their mathematical structure and making formal verification potentially costly.

Symbolic Models for Lyapunov Discovery. Recent work seeks explicit Lyapunov functions through neural symbolic generation, building on advances in symbolic mathematics and symbolic regression (Lample & Charton, 2020; Biggio et al., 2021; Petersen et al., 2021; Landajuela et al., 2022; Kamienny et al., 2022; 2023). Symbolic regression has been integrated with neural Lyapunov learning (Feng et al., 2024), while symbolic reduction has simplified polynomial Lyapunov synthesis (Liu & Fitzsimmons, 2025). Alfarano et al. (2024) exploit backward construction to generate large-scale valid system–Lyapunov pairs and train a sequence-to-sequence Transformer that amortizes Lyapunov discovery across polynomial and non-polynomial systems. However, the resulting training corpus remains fixed throughout the subsequent solver learning process. Zou et al. (2025) instead perform system-specific symbolic optimization using reinforcement learning and candidate refinement, exploiting feedback from the target system but restarting optimization for each new instance. Constrained symbolic regression provides another route by incorporating Lyapunov conditions directly into expression search (Mitrai & Tang, 2026). What therefore remains missing is a mechanism that turns feedback into verified examples for improving a reusable cross-system solver.

3 PRELIMINARIES

3.1 DYNAMICAL SYSTEMS, LYAPUNOV FUNCTIONS, AND VERIFICATION

Dynamical Systems and Lyapunov Functions We consider an autonomous continuous-time dynamical system $\dot{x} = f(x)$, where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz, with an equilibrium at the origin satisfying $f(0) = 0$. The stability of this equilibrium can be established by the existence of a Lyapunov function, a continuously differentiable scalar function $V : \mathbb{R}^n \to \mathbb{R}$ that is positive away from the origin and non-increasing along system trajectories. We center each candidate as $\bar{V}(x) = V(x) - V(0)$ and define its Lie derivative along the dynamics as $L_f V(x) = \nabla V(x)^\top f(x)$. Accordingly, a valid Lyapunov function satisfies $\bar{V}(0) = 0$, $\bar{V}(x) > 0$ for $x \neq 0$, and $L_f V(x) \leq 0$ for $x \neq 0$. The existence of such a Lyapunov function guarantees that the origin is a stable equilibrium. If, in addition, the stronger condition $L_f V(x) < 0$ holds for all $x \neq 0$, the origin is asymptotically stable (Massera, 1949; LaSalle, 1960; Khalil, 2002; Kellett, 2015).

SHGO-based Verification To assess the analytical Lyapunov conditions above at scale, we follow prior work (Alfarano et al., 2024; Zou et al., 2025) and use simplicial homology global optimization (SHGO) (Endres et al., 2018) as a bounded numerical verification criterion. Each candidate is evaluated over the bounded admissible region $\Omega_R = [-R, R]^n \cap \mathrm{Dom}(f, V)$, with $R = 5$ in all experiments. Specifically, SHGO evaluates $\min_{x \in \Omega_R \setminus B_\epsilon(0)} \bar{V}(x)$ and $\min_{x \in \Omega_R} [-L_f V(x)]$, where $B_\epsilon(0)$ excludes a small neighborhood of the equilibrium from the strict-positivity check. A candidate is accepted if neither computed minimum violates the corresponding condition beyond the prescribed numerical tolerance, indicating that no such violation is found over the bounded admissible region under this criterion. We refer to this procedure as the SHGO Verifier and apply it consistently throughout data construction, training, and evaluation across all reported experimental settings. The complete verification protocol is detailed in Appendix A, and its consistency with stronger verification procedures is further examined using dReal and global SOS verification in Appendix B.

3.2 BACKWARD CONSTRUCTION OF LYAPUNOV TRAINING PAIRS

Training a symbolic solver requires paired examples (f, V) in which V is valid for the corresponding dynamics f. Constructing such pairs directly is difficult: randomly sampled dynamics need not be stable, while recovering an explicit Lyapunov function for a given stable system is itself the target problem. Backward construction (Alfarano et al., 2024) reverses this process by first constructing a Lyapunov function and then generating dynamics that satisfy its Lyapunov conditions.

The Lyapunov function is constructed as $V = V_{\text{proper}} + V_{\text{cross}}$, where

$$V_{\text{cross}}(x) = \sum_{k=1}^{m_q} q_k^2(x), \quad q_k(0) = 0, \qquad V_{\text{proper}}(x) = \sum_{a,b=1}^{n} \alpha_{ab} x_a^{\beta_a} x_b^{\beta_b}, \quad A = (\alpha_{ab}) \succ 0. \tag{1}$$

Here, V_{proper} provides the positive-definite base of the Lyapunov function, while the squared terms in V_{cross} increase its symbolic complexity without destroying positive definiteness.

Once V is fixed, backward construction further constructs the dynamics f so that the Lyapunov decrease condition $L_f V(x) \leq 0$ is satisfied by construction. Specifically, the dynamics combine a dissipative component with a gradient-orthogonal component:

$$f(x) = -\big(h_i^2(x)\, \partial_i V(x)\big)_{i=1}^{n} + \sum_{j=1}^{m_g} g_j(x) e_j(x). \tag{2}$$

Here, $h_i(x)$ controls the dissipative contribution along the i-th state dimension, while $e_j(x)$ denotes a direction orthogonal to $\nabla V(x)$, satisfying $e_j(x)^\top \nabla V(x) = 0$, and $g_j(x)$ controls its contribution to the dynamics. The dissipative component drives V toward smaller values, whereas the gradient-orthogonal component enriches the generated dynamics without changing V along those directions. Consequently, the orthogonal component makes no contribution to the Lie derivative, yielding

$$L_f V(x) = -\sum_{i=1}^{n} h_i^2(x) \big(\partial_i V(x)\big)^2 \leq 0. \tag{3}$$

Thus, backward construction simultaneously constructs a Lyapunov function and corresponding dynamics that satisfy the required Lyapunov conditions. Further details are provided in Appendix C.

Backward construction efficiently constructs valid training pairs but yields static datasets fixed before solver training. In this offline setting, the key symbolic components q_k, g_j, and h_i are randomly drawn from predefined distributions. Consequently, generation cannot adapt to current solver weaknesses, limiting its ability to continually produce informative and challenging training systems.

4 METHOD

SEVA-Lyap couples a learnable backward Generator G_θ with a reusable Solver S_ϕ in a closed loop. As summarized in Figure 1, both models are warm-started on a filtered corpus produced by backward construction before solver-aware data evolution, where feedback from the Solver continually reshapes verified data generation and guides updates to the Solver. At inference, direct prediction is attempted first, and only unresolved systems invoke verification-guided risk-seeking adaptation.

4.1 LEARNABLE BACKWARD CONSTRUCTION WITH OFFLINE INITIALIZATION

We first warm-start the Generator and Solver separately for the polynomial and non-polynomial settings, using a filtered corpus of valid training pairs from backward construction for each family, so that the Generator acquires basic symbolic generation capability and the Solver develops initial competence in Lyapunov-function prediction. Using teacher-forced cross-entropy for supervised initialization, the Generator learns $V_{\text{proper}} \to (Q, G, H)$, while the Solver learns $f \to V$.

SEVA-Lyap then retains the validity-preserving backward construction while making its key symbolic choices learnable. As illustrated in Figure 1(a), conditioned on V_{proper}, the Generator G_θ autoregressively produces three structured symbolic groups Q, G, and H. Specifically, $Q = \{q_k\}_{k=1}^{m_q}$ determines the squared symbolic terms that enrich the Lyapunov function; $G = \{g_j\}_{j=1}^{m_g}$ weights the gradient-orthogonal directions $e_j(x)$ that diversify the dynamics without changing V; and $H = \{h_i\}_{i=1}^{n}$ controls the dissipative contribution along each state dimension.

Given the generated symbolic components, a deterministic assembler $\mathcal{A}$ follows the backward-generation rules to assemble $(V_{\text{proper}}, Q, G, H)$ into an explicit system–Lyapunov pair (f, V_{GT}). The Generator therefore does not directly produce an unconstrained vector field, but instead selects symbolic structures within a validity-preserving construction. Meanwhile, the Solver S_ϕ maps the resulting dynamics f to a symbolic Lyapunov candidate. This division of roles makes Solver's performance reflect the difficulty of Generator-selected symbolic structures, allowing subsequent data evolution to shift generation toward structures that remain difficult for the current Solver.

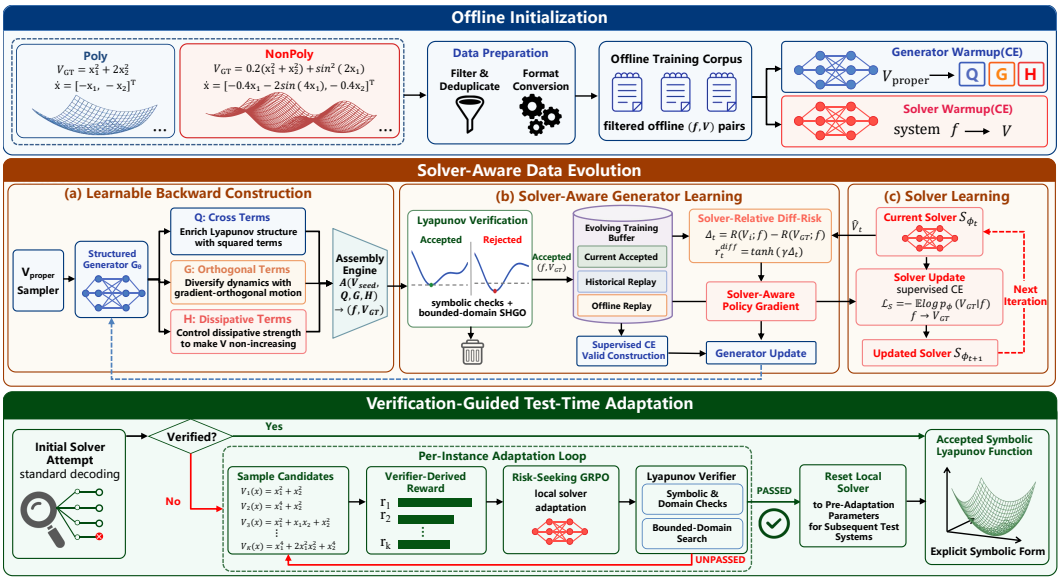

Figure 1: Overview of **SEVA-Lyap**. Offline initialization warm-starts the Generator and Solver on a filtered corpus produced by backward construction. During solver-aware data evolution, learnable backward construction produces newly verified training pairs, solver-relative feedback adapts the Generator, and the evolving buffer iteratively updates the Solver. At test time, unresolved systems trigger verifier-guided risk-seeking GRPO on a local Solver, whose parameters are reset before subsequent test systems. Solid and dashed arrows denote data flow and parameter updates, respectively.

4.2 Generator and Solver Co-Evolution

After offline initialization, the Generator uses the learnable construction above to produce system–Lyapunov pairs and then enters an iterative coupled coevolution process with the Solver. Each assembled pair (f, V_{GT}) is checked by the SHGO Verifier; invalid pairs are discarded, while verified pairs proceed to solver-relative evaluation against the current Solver before entering the evolving training buffer. The shared training buffer $\mathcal{B}_t$ combines newly accepted construction records with historical replay from earlier evolution rounds and offline replay from the warmup corpus, providing balanced training data for Generator adaptation and updates to the Solver.

The Generator is first trained via supervised learning to preserve its ability to produce valid outputs:

$$\mathcal{L}_G^{\mathrm{sup}} = -\mathbb{E}_{(V_{\mathrm{proper}},Q,G,H)\sim\mathcal{B}_t}\left[\log p_\theta\big(Q,G,H \mid V_{\mathrm{proper}}\big)\right]. \tag{4}$$

However, validity alone does not indicate whether a generated system provides a sufficiently informative learning signal for the current Solver. For each accepted pair (f, V_{GT}), the current Solver S_{ϕ_t} produces the candidate prediction $\widehat{V}_t = S_{\phi_t}(f)$. We then quantify its violations of the Lyapunov conditions $L_f V(x) \le 0$ and $\bar{V}(x) > 0$ using the following empirical violation score:

$$\mathcal{R}(V; f) = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}[L_f V(x)]_+ + \frac{1}{|\mathcal{X}\setminus B_\epsilon(0)|}\sum_{x\in\mathcal{X}\setminus B_\epsilon(0)}[-\bar{V}(x)]_+, \tag{5}$$

where $[u]_+ = \max(u, 0)$ and $\mathcal{X} \subset \Omega_R$ denotes the finite state set sampled from the bounded verification region. We then define the solver-relative difficulty as:

$$\Delta_t(f) = \mathcal{R}(\widehat{V}_t; f) - \mathcal{R}(V_{\mathrm{GT}}; f), \qquad r_t^{\mathrm{diff}} = \tanh\big(\gamma\Delta_t(f)\big), \tag{6}$$

where $\gamma > 0$ controls the scale. A larger positive $\Delta_t(f)$ indicates a wider violation gap between the verified target and the current Solver candidate, and hence a more challenging system.

We use the solver-relative difficulty as the reward to further update the Generator (Williams, 1992):

$$\mathcal{L}_G^{\mathrm{PG}} = -\mathbb{E}_{(Q,G,H)\sim p_\theta(\cdot|V_{\mathrm{proper}})}\left[\tilde{r}_t^{\mathrm{diff}}\log p_\theta\big(Q,G,H \mid V_{\mathrm{proper}}\big)\right], \tag{7}$$

where $\widetilde{r}_t^{\text{diff}}$ denotes the normalized form of r_t^{diff}. This policy-gradient update progressively steers the generation distribution toward verified systems that remain challenging for the current Solver.

Verified records in the shared buffer are then used to further update the reusable Solver through teacher-forced cross-entropy, enabling it to continually learn from newly evolved training systems:

$$\mathcal{L}_S(\phi) = -\mathbb{E}_{(f,V_{\text{GT}})\sim\mathcal{B}_t}\left[\log p_\phi\big(V_{\text{GT}} \mid f\big)\right]. \tag{8}$$

Newly accepted systems provide targeted supervision on weaknesses exposed by the current Solver, while replay preserves prior symbolic competence as the generation distribution continues to evolve. As the Solver improves, previously difficult systems yield weaker rewards, continuously driving the Generator toward the Solver's evolving capability frontier and creating a dynamic curriculum across successive rounds that progressively strengthens cross-system solving competence.

4.3 Verification-Guided Test-Time Adaptation

After Generator–Solver coevolution, the resulting reusable Solver $S_{\phi^\star}$ serves as the shared initialization and inference model across test systems. For each test system $f^{(i)}$, it first generates symbolic candidates for verification using the SHGO Verifier. If no candidate is accepted, a local Solver copy initialized from $S_{\phi^\star}$ is adapted only to the current system. During local adaptation, K candidates $\{\tau_j\}_{j=1}^K$ are sampled, and verification feedback from the SHGO Verifier is used to construct rewards r_j according to how well each candidate satisfies the Lyapunov positivity and decrease conditions. Since solving the current system requires only one candidate to pass verification, the relevant capability is the best candidate the local Solver can produce rather than its average candidate quality. We therefore adopt a risk-seeking objective (Petersen et al., 2021; Bastiani et al., 2026) that concentrates optimization on the high-reward tail of the candidate distribution.

Rewards are clipped as $\widetilde{r}_j = \text{clip}(r_j, -c, c)$, where c limits extreme values, and b_q is their empirical q-quantile. Let $\mathcal{E}_q = \{j : \widetilde{r}_j - b_q > \epsilon_A\}$ denote candidates above this threshold, with $\epsilon_A > 0$ a small numerical tolerance. When $\mathcal{E}_q \neq \varnothing$, we define the risk-seeking group-relative advantage as

$$A_j^{\text{RS}} = \frac{[\widetilde{r}_j - b_q]_+}{\frac{1}{|\mathcal{E}_q|}\sum_{\ell\in\mathcal{E}_q}[\widetilde{r}_\ell - b_q]_+ + \epsilon_A}, \qquad [z]_+ = \max(z, 0). \tag{9}$$

Candidates below the threshold receive zero policy-gradient weight, while upper-tail candidates receive larger advantages. If $\mathcal{E}_q = \varnothing$, the update is skipped. Reward r_j is detailed in Appendix D.3.

The upper-tail advantages provide focused learning signals for Solver updates through risk-seeking GRPO (Shao et al., 2024), emphasizing high-reward candidates during local adaptation. Let

$$\varrho_j = \exp\big(\bar{\ell}_j(\phi) - \bar{\ell}_j(\phi_{\text{old}})\big), \qquad \bar{\varrho}_j = \text{clip}\big(\varrho_j, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\big), \tag{10}$$

where $\bar{\ell}_j(\phi)$ is the average token log-probability of candidate τ_j, and ϕ_{old} denotes the local policy before the update. Since $A_j^{\text{RS}} \geq 0$, the clipped surrogate reduces to $A_j^{\text{RS}}\min(\varrho_j, \bar{\varrho}_j)$, yielding

$$\mathcal{L}_{\text{RS-GRPO}}(\phi) = -\frac{1}{K}\sum_{j=1}^{K} A_j^{\text{RS}}\min(\varrho_j, \bar{\varrho}_j) + \frac{\beta}{K}\sum_{j=1}^{K} D_{\text{KL}}^{\text{tok}}\left(\pi_\phi \,\|\, \pi_{\phi^\star}\right). \tag{11}$$

The token-level KL term is evaluated on sampled candidates conditioned on $f^{(i)}$, with ϵ_{clip} limiting abrupt local policy changes (Schulman et al., 2017) and β regularizing adaptation toward the fixed reusable Solver policy $\pi_{\phi^\star}$. This group-relative formulation requires no separate value model, while upper-tail advantages focus optimization on producing at least one verifiable symbolic candidate.

Sampling, verification, and local optimization repeat until a candidate passes the SHGO Verifier or the test-time budget is exhausted, after which the local Solver is reset to $S_{\phi^\star}$ before tackling the next system. This instance-wise refinement broadens the Solver's exploration capacity for unresolved systems at inference and tightly integrates solver-aware coevolutionary training with verification-guided test-time adaptation. Further training and inference details are provided in Appendix D.

5 EXPERIMENTS

5.1 BENCHMARKS AND DATASETS

Poly and NonPoly benchmarks. We evaluate polynomial (*Poly*) and non-polynomial (*NonPoly*) autonomous systems as primary benchmarks. Poly spans dimensions 2–5, whereas NonPoly includes systems in dimensions 2 and 3 with non-polynomial operators. Each benchmark contains equal numbers of held-out systems from conventional backward construction and solver-aware generation, all of which are disjoint from training, fixed before evaluation, and shared across methods.

Transfer and challenging systems. To assess transfer to independently generated systems, we further evaluate SEVA-Lyap's transfer capabilities using an Into-the-Wild (ITW) benchmark, where system dynamics f are sampled directly without first constructing a reference Lyapunov function. Following a basic filtering step removing invalid symbolic systems and those that are clearly unstable near the origin, ITW contains 2,000 Poly and 2,000 NonPoly systems. We additionally evaluate representative challenging systems reported in established Lyapunov studies complementing the ITW benchmark. Details of the ITW construction and filtering procedure are provided in Appendix F.

5.2 BASELINES

We compare SEVA-Lyap with four Lyapunov-synthesis and two symbolic learning baselines.

Lyapunov-synthesis baselines. LyZNet learns Lyapunov functions from sampled states and verifies them with SMT procedures (Liu et al., 2024), whereas ANLC and FOSSIL 2.0 iteratively refine candidates via falsification or verifier feedback (Grande et al., 2023; Edwards et al., 2024). SOS synthesis instead seeks polynomial Lyapunov functions by relaxing polynomial nonnegativity conditions into semidefinite programs (Prajna et al., 2002; Papachristodoulou & Prajna, 2005b).

Symbolic learning baselines. The Symbolic Transformer is trained on synthetic system–Lyapunov pairs and generates symbolic Lyapunov candidates for unseen systems through beam-search decoding (Alfarano et al., 2024). By contrast, the RL-based Approach performs system-specific symbolic optimization from scratch using numerical Lyapunov-condition violations and iterative counterexample feedback, followed by genetic-programming refinement (Zou et al., 2025).

5.3 EXPERIMENTAL SETUP

We report verification success rate, counting a system as solved when at least one candidate generated within the prescribed computational budget passes the SHGO Verifier. The Symbolic Transformer (Alfarano et al., 2024) follows its original data-generation and training configuration reported in prior work. For a controlled comparison, SEVA-Lyap and the Symbolic Transformer use the same solver architecture, parameter count, and decoding configuration, with the solver supervision scale matched to 500K system–Lyapunov pairs. The RL-based Approach (Zou et al., 2025) follows its original configuration with up to $18{,}000$ seconds of optimization per test system. SEVA-Lyap performs test-time adaptation only on unresolved systems, capped at 600 seconds per instance, and the default risk-seeking update retains the top 25% of sampled candidates in each group. Detailed SEVA-Lyap architectures, optimization settings, runtime and hardware specifications are provided in Appendix E, with additional implementation details for all evaluated baselines in Appendix H.

6 RESULTS AND ANALYSIS

Our evaluation examines overall performance, how generated-system difficulty evolves during training, whether the learned Solver generalizes beyond training distributions, and how the main training and test-time adaptation components contribute to the final performance.

6.1 OVERALL PERFORMANCE

Table 1 reports system-level verification success on Poly and NonPoly. SEVA-Lyap (Direct) achieves 76.75% and 65.50% on Poly and NonPoly, respectively, outperforming all baselines on

Table 1: System-level verification success rate (%) on Poly and NonPoly. *Direct* uses the Solver obtained after solver-aware data evolution, while *Full* additionally applies verification-guided test-time adaptation to unresolved systems. SOS synthesis is evaluated on Poly only.

Method	Poly ↑	NonPoly ↑
LyZNet (Liu et al., 2024)	15.50	21.75
ANLC (Grande et al., 2023)	26.75	41.00
FOSSIL 2.0 (Edwards et al., 2024)	52.75	36.75
SOS synthesis (Prajna et al., 2002)	29.75	–
Symbolic Transformer (Alfarano et al., 2024)	70.75	60.50
RL-based Approach (Zou et al., 2025)	30.50	32.25
SEVA-Lyap (Direct)	76.75	65.50
SEVA-Lyap (Full)	**92.00**	**74.50**

both families. Relative to the Symbolic Transformer, the evolved Solver gains 6.00 and 5.00 percentage points, showing that data evolution strengthens the reusable Solver itself. Verification-guided test-time adaptation on unresolved systems yields SEVA-Lyap (Full), raising success to 92.00% and 74.50%, with gains of 15.25 and 9.00 percentage points over direct inference. Together, these results show a coherent progression within SEVA-Lyap: the shared Solver is strengthened across systems and selectively refined when direct prediction remains insufficient. Results across training scales and random seeds are reported in Appendix G.1 and Appendix E.6, respectively.

6.2 Evolution of Generated-System Difficulty

We next examine whether solver-aware data evolution progressively produces more challenging training systems. For each family, accepted systems are ordered by generation time and divided into ten consecutive 50K intervals. To keep difficulty assessment independent of the evolving Solver, we sample 200 systems from each interval and evaluate them using the fixed Symbolic Transformer baseline, repeating the analysis over three random seeds. Figure 2 reports mean verification success, with error bars showing the across-seed variation. As generation progresses, success decreases substantially for both Poly and NonPoly, from above 80% early in evolution to the mid-50% range near the end.

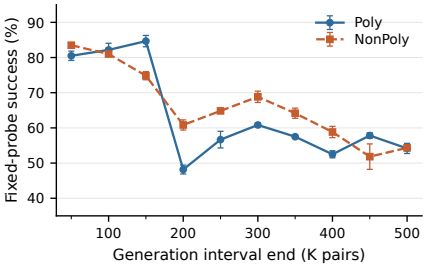

Figure 2: Evolution of generated-system difficulty throughout the solver-aware data evolution process.

Despite local fluctuations, both families exhibit a clear overall downward trend, providing evidence that solver-aware data evolution increasingly concentrates on systems that remain difficult for the fixed Symbolic Transformer baseline. Complexity-controlled analyses in Appendix G.1 further show that this trend cannot be explained solely by increasing system dimension or expression size.

6.3 Generalization to Independently Constructed Systems

Beyond the primary Poly and NonPoly benchmarks, we further evaluate SEVA-Lyap on the Into-the-Wild (ITW) benchmark to assess generalization to independently constructed systems. Table 2 reports the ITW results. Unlike the main benchmarks, ITW applies only a basic filter that removes invalid systems and those that are clearly unstable near the origin; the retained systems are not guaranteed to be stable and may still include many unstable cases that do not admit a valid Lyapunov function. This makes ITW a substantially broader and more challenging distribution, and explains the much lower absolute success rates. SEVA-Lyap (Direct) improves success from 3.0 to $4.5‰$ on Poly and from 3.5 to $4.5‰$ on NonPoly. Test-time adaptation further improves SEVA-Lyap (Full) to $18.0‰$ and $21.5‰$, respectively. These results show that the reusable Solver retains transfer capability on independently constructed systems, while verification-guided local adaptation provides substantial additional gains when direct prediction fails under this broader distribution.

Table 2: ITW verification success (‰). ITW systems are not stability-certified, so the reported rates are computed over the full retained system pool rather than a known-stable subset.

Method	Poly ↑	NonPoly ↑
Symbolic Transformer	3.0	3.5
RL-based Approach	2.5	4.0
SEVA-Lyap (Direct)	4.5	4.5
SEVA-Lyap (Full)	**18.0**	**21.5**

Table 3: Ablation of SEVA-Lyap on Poly.

Variant	Success ↑	Δ
SEVA-Lyap (Direct)	76.75	–
No Generator RL	67.75	−9.00
Binary validity reward	65.75	−11.00
No replay	47.50	−29.25
SEVA-Lyap (Full, RS-25)	92.00	–
RS-50	90.00	−2.00
No risk-seeking (RS-100)	88.50	−3.50

To further assess SEVA-Lyap's generality beyond the main evaluation families, we conduct cross-family transfer experiments in Appendix G.2. We also evaluate its ability to solve externally defined challenging systems using representative cases from prior Lyapunov studies in Appendix I.

6.4 Ablation Analysis

To assess the role of individual design choices, we conduct ablations of the main design mechanisms in SEVA-Lyap on the Poly benchmark, with the corresponding results reported in Table 3.

The ablation results highlight the main contributions of both training and test-time adaptation. For solver-aware data evolution, removing the Generator policy-gradient update reduces success from 76.75% to 67.75%, indicating that actively shaping the difficulty distribution of the training data is crucial for strengthening the reusable Solver. Replacing the graded solver-relative reward with binary validity feedback further lowers success to 65.75%, showing that the relative difficulty signal provides more informative guidance than simply distinguishing valid from invalid constructions. The largest drop occurs without replay, where success falls to 47.50%, underscoring its role in preserving previously learned competence as the training distribution evolves across successive rounds.

We denote by RS-p the risk-seeking setting that retains the top $p\%$ of sampled candidates ranked by reward for each policy updates, with SEVA-Lyap (Full) adopting RS-25. For test-time adaptation, increasing the retained fraction from RS-25 to RS-50 reduces success from 92.00% to 90.00%, while removing risk-seeking altogether (RS-100) further lowers it to 88.50%. This trend suggests that concentrating updates on the high-reward tail is more effective than broadly optimizing over all sampled candidates, which can dilute the learning signal from the most promising solutions.

7 Conclusion and Discussion

Conclusion. We presented SEVA-Lyap, a framework for symbolic Lyapunov discovery in which the training distribution evolves together with the Solver. Through solver-aware data evolution, the Generator continually produces valid systems that remain challenging for the current Solver, allowing the reusable Solver to improve as training progresses. When direct inference is insufficient, verification-guided test-time adaptation further refines the Solver for unresolved systems and extends its effective solving capability. Experiments demonstrate that SEVA-Lyap progressively steers training toward verified systems that remain difficult for the current Solver, thereby strengthening reusable cross-system prediction and enabling further recovery through targeted test-time adaptation. More broadly, SEVA-Lyap demonstrates how data generation, solver learning, and verification can be integrated into a unified learning process for symbolic Lyapunov discovery.

Discussion and limitations. While our present study focuses on Lyapunov function discovery, extending the broader applicability of SEVA-Lyap represents an important direction for future work. Its underlying principle is not tied strictly to Lyapunov functions and requires only explicit criteria for evaluating candidate solutions, allowing the framework to generalize naturally to related problems. This makes barrier functions (Ames et al., 2017), control Lyapunov functions (Sontag, 1989), first integrals (Zhong et al., 2026), and conserved quantities natural extensions of the framework. Exploring these settings would further establish solver-aware data evolution and verification-guided adaptation as a broader approach to verification-driven symbolic scientific discovery.

AI USE STATEMENT

Generative AI tools were used to assist with method implementation, code development, manuscript polishing, and the refinement of figures and presentation. Suggestions provided by generative AI were used only as supporting material rather than as authoritative scientific conclusions, with all substantive research decisions made by the authors. All methodological designs, mathematical formulations, implementations, experimental results, and interpretations reported in this work were independently examined and verified by the authors. The authors take full responsibility for the final content of the paper and for all material produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies the automated discovery and verification of symbolic Lyapunov functions and does not involve human subjects, personal data, or sensitive user information. The experimental systems are either synthetically generated or drawn from publicly available benchmarks and the literature. By producing explicit and independently verifiable symbolic Lyapunov functions, this work may contribute to more transparent and interpretable stability analysis in scientific and control applications. We do not identify any specific ethical concerns arising from the data, methodology, or experimental procedures used in this study under standard academic research practices.

REPRODUCIBILITY STATEMENT

We provide detailed documentation and an anonymous repository containing implementation code and model files to support reproduction of the reported results. To complement these resources, the appendix specifies the verification protocol and independent verification audits, symbolic representation and structured assembly, training and inference procedures, model and optimization settings, computational environment, benchmark construction, and baseline configurations. Additional analyses of training scale, multi-seed variation, difficulty evolution, cross-family transfer, and representative challenging systems are also reported in the appendix. Together, these materials document the main algorithmic, experimental, and evaluation choices required to reproduce the study.

REFERENCES

Amir Ali Ahmadi. On the difficulty of deciding asymptotic stability of cubic homogeneous vector fields. In *Proceedings of the 2012 American Control Conference*, pp. 3334–3339, 2012. doi: 10.1109/ACC.2012.6315687.

Amir Ali Ahmadi and Bachir El Khadir. On algebraic proofs of stability for homogeneous vector fields. *IEEE Transactions on Automatic Control*, 65(1):325–332, 2020. doi: 10.1109/TAC.2019.2914968.

Amir Ali Ahmadi and Pablo A. Parrilo. Converse results on existence of sum of squares lyapunov functions. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 6516–6521, 2011. doi: 10.1109/CDC.2011.6161493.

Amir Ali Ahmadi, Miroslav Krstic, and Pablo A. Parrilo. A globally asymptotically stable polynomial vector field with no polynomial lyapunov function. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 7579–7580, 2011. doi: 10.1109/CDC.2011.6161499.

Alberto Alfarano, François Charton, and Amaury Hayat. Global lyapunov functions: A long-standing open problem in mathematics, with symbolic transformers. In *Advances in Neural Information Processing Systems*, volume 37, pp. 93643–93670, 2024. doi: 10.52202/079017-2969. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/aa280e73c4e23e765fde232571116d3b-Abstract-Conference.html.

Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017. doi: 10.1109/TAC.2016.2638961.

Zachary Bastiani, Mike Kirby, Jacob Hochhalter, and Shandian Zhe. Complexity-aware deep symbolic regression with robust risk-seeking policy gradients. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 2098–2106. PMLR, 2026. URL https://proceedings.mlr.press/v300/bastiani26a.html.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurélien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 936–945. PMLR, 2021. URL https://proceedings.mlr.press/v139/biggio21a.html.

Nicholas Boffi, Stephen Tu, Nikolai Matni, Jean-Jacques Slotine, and Vikas Sindhwani. Learning stability certificates from data. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1341–1350. PMLR, 2021. URL https://proceedings.mlr.press/v155/boffi21a.html.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, volume 32, pp. 3240–3249, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/2647c1dba23bc0e0f9cdf75339e120d2-Abstract.html.

Wenqi Cui and Baosen Zhang. Lyapunov-regularized reinforcement learning for power system transient stability. *IEEE Control Systems Letters*, 6:974–979, 2022. doi: 10.1109/LCSYS.2021.3088068.

Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. In *Proceedings of Robotics: Science and Systems*, 2021. doi: 10.15607/RSS.2021.XVII.063.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023. doi: 10.1109/TRO.2022.3232542.

Alec Edwards, Andrea Peruffo, and Alessandro Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 2024. doi: 10.1145/3641513.3651398.

Stefan C. Endres, Carl Sandrock, and Walter W. Focke. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72(2):181–217, 2018. doi: 10.1007/s10898-018-0645-y.

Jie Feng, Haohan Zou, and Yuanyuan Shi. Combining neural networks and symbolic regression for analytical lyapunov function discovery. In *ICML 2024 Workshop on Foundations of Reinforcement Learning and Control*, 2024. URL https://openreview.net/forum?id=Knj78wY9T4.

Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *Automated Deduction – CADE-24*, pp. 208–214. Springer, 2013. doi: 10.1007/978-3-642-38574-2_14.

Peter Giesl. *Construction of Global Lyapunov Functions Using Radial Basis Functions*, volume 1904 of *Lecture Notes in Mathematics*. Springer, 2007. doi: 10.1007/978-3-540-69909-5.

Davide Grande, Andrea Peruffo, Enrico Anderlini, and Georgios Salavasidis. Augmented neural lyapunov control. *IEEE Access*, 11:67979–67986, 2023. doi: 10.1109/ACCESS.2023.3291349.

Lars Grüne. Computing lyapunov functions using deep neural networks. *Journal of Computational Dynamics*, 8(2):131–152, 2021. doi: 10.3934/jcd.2021006.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 10269–10281, 2022. doi: 10.52202/068431-0746.

Pierre-Alexandre Kamienny, Guillaume Lample, Sylvain Lamprier, and Marco Virgolin. Deep generative symbolic regression with monte-carlo-tree-search. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15655–15668. PMLR, 2023. URL https://proceedings.mlr.press/v202/kamienny23a.html.

Christopher M. Kellett. Classical converse theorems in lyapunov's second method. *Discrete and Continuous Dynamical Systems – B*, 20(8):2333–2360, 2015. doi: 10.3934/dcdsb.2015.20.2333.

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

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1eZYeHFDS.

Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P. Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K. Petersen. A unified framework for deep symbolic regression. In *Advances in Neural Information Processing Systems*, volume 35, pp. 33985–33998, 2022. doi: 10.52202/068431-2463.

Joseph P. LaSalle. Some extensions of liapunov's second method. *IRE Transactions on Circuit Theory*, 7(4):520–527, 1960. doi: 10.1109/TCT.1960.1086720.

Jun Liu and Maxwell Fitzsimmons. Symbolic reduction for formal synthesis of global lyapunov functions. In *Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems (QEST+FORMATS 2025)*, Lecture Notes in Computer Science, pp. 333–352. Springer, 2025. doi: 10.1007/978-3-032-05792-1_18.

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. LyZNet: A lightweight python tool for learning and verifying neural lyapunov functions and regions of attraction. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 25:1–25:8. ACM, 2024. doi: 10.1145/3641513.3650134.

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Physics-informed neural network lyapunov functions: PDE characterization, learning, and verification. *Automatica*, 175:112193, 2025. doi: 10.1016/j.automatica.2025.112193.

Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2): 130–141, 1963. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2.

José Luis Massera. On liapounoff's conditions of stability. *Annals of Mathematics*, 50(3):705–721, 1949. doi: 10.2307/1969558.

Ilias Mitrai and Wentao Tang. A constrained symbolic regression approach for lyapunov function discovery. *arXiv preprint arXiv:2606.10045*, 2026. URL https://arxiv.org/abs/2606.10045.

Antonis Papachristodoulou and Stephen Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. In *Positive Polynomials in Control*, pp. 23–43. Springer, 2005a. doi: 10.1007/10997703_2.

Antonis Papachristodoulou and Stephen Prajna. A tutorial on sum of squares techniques for systems analysis. In *Proceedings of the 2005 American Control Conference*, volume 4, pp. 2686–2700, 2005b. doi: 10.1109/ACC.2005.1470374.

Matthew M. Peet. Exponentially stable nonlinear systems have polynomial lyapunov functions on bounded regions. *IEEE Transactions on Automatic Control*, 54(5):979–987, 2009. doi: 10.1109/TAC.2009.2017116.

Brenden K. Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=m5Qsh0kBQG.

Stephen Prajna, Antonis Papachristodoulou, and Pablo A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 1, pp. 741–746, 2002. doi: 10.1109/CDC.2002.1184594.

Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of the 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 466–476. PMLR, 2018. URL https://proceedings.mlr.press/v87/richards18a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Eduardo D. Sontag. A "universal" construction of artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989. doi: 10.1016/0167-6911(89)90028-5.

A. Vannelli and M. Vidyasagar. Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985. doi: 10.1016/0005-1098(85)90099-8.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural lyapunov control for discrete-time systems. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0131. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/08bf1773e94763b6cc366ee7c6582f27-Abstract-Conference.html.

Lujie Yang, Hongkai Dai, Zhouxing Shi, Cho-Jui Hsieh, Russ Tedrake, and Huan Zhang. Lyapunov-stable neural control for state and output feedback: A novel formulation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 56033–56046. PMLR, 2024. URL https://proceedings.mlr.press/v235/yang24f.html.

Jingfeng Zhong, Zhengxiang Liu, Zhijie Wang, and Shuai Li. Learning first integrals via backward-generated data and guided reinforcement learning. *arXiv preprint arXiv:2605.21160*, 2026. URL https://arxiv.org/abs/2605.21160.

Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-2111. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/bba3160c42f4c1d23175b5abf04347f5-Abstract-Conference.html.

Haohan Zou, Jie Feng, Hao Zhao, and Yuanyuan Shi. Analytical lyapunov function discovery: An RL-based generative approach. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80776–80804. PMLR, 2025. URL https://proceedings.mlr.press/v267/zou25a.html.

A VERIFICATION PROTOCOL

A.1 VERIFICATION SCOPE AND ACCEPTANCE CRITERION

As introduced in Sec. 3.1, verification is performed on the centered candidate

$$\bar{V}(x) = V(x) - V(0) \tag{A.1}$$

and its Lie derivative

$$\mathcal{L}_f V(x) = \nabla V(x)^\top f(x) \tag{A.2}$$

over the bounded admissible region

$$\Omega_R = [-R, R]^n \cap \mathrm{Dom}(f, V). \tag{A.3}$$

Centering removes any additive constant in the predicted expression, so the original candidate is not required to satisfy $V(0) = 0$. The domain term is active whenever the dynamics or the candidate contain operators with restricted natural domains.

The verification procedure combines symbolic prechecks with numerical verification. The symbolic stage parses the dynamics and candidate under the declared state dimension, checks supported operators and legal variables, constructs $\bar{V}$ and $\mathcal{L}_f V$, and extracts natural-domain constraints when required. It serves primarily as a structural and consistency check rather than a complete symbolic proof of positivity or decrease. These Lyapunov conditions are subsequently assessed by the numerical verification stage described below. The same SHGO acceptance criterion is used whenever final candidates are evaluated during data construction, solver evaluation, and testing.

A.2 SHGO NUMERICAL VERIFICATION

Following the symbolic precheck, the SHGO Verifier uses simplicial homology global optimization (SHGO) to search for violations of the Lyapunov conditions.

$$\widehat{m}_V \approx \min_{x \in \Omega_R \setminus B_\epsilon(0)} \bar{V}(x), \qquad \widehat{m}_{\dot{V}} \approx \min_{x \in \Omega_R} [-\mathcal{L}_f V(x)]. \tag{A.4}$$

The strict-positivity search excludes a small neighborhood of the equilibrium with $\epsilon = 10^{-6}$. A candidate is accepted when all requested checks succeed and

$$\widehat{m}_V \geq -\tau, \qquad \widehat{m}_{\dot{V}} \geq -\tau. \tag{A.5}$$

Accordingly, acceptance indicates that no violation beyond the specified numerical tolerance is found over the bounded admissible region,rather than an unrestricted analytical guarantee over the entire state space.

Table A.1: Reference verification configuration used for reported evaluation.

Component	Setting	Decision semantics
Verification stages	symbolic, SHGO	Symbolic precheck followed by numerical verification
Region	$R = 5$	Box intersected with enabled natural-domain constraints
Origin-gradient tolerance	10^{-14}	Numerical origin check
SHGO sampling	$n = 256$	Simplicial global search
Acceptance tolerance	$\tau = 10^{-10}$	Numerical decision tolerance
Requested margins	$m_+ = m_- = 0$	No additional positivity/decrease margin

At the system level, beam-search evaluation counts a system as solved if at least one of the five decoded candidates passes the SHGO Verifier. Candidates recovered through test-time adaptation are evaluated by the same SHGO Verifier.

A.3 Parsing and Natural Domains

The symbolic parser requires complete arity-consistent expressions, supported operators, legal variables, and a state dimension consistent with the vector field. For NonPoly expressions, the admissible domain additionally accounts for restrictions induced by division, logarithms, square roots, inverse trigonometric functions, tangent singularities, and non-integer or negative powers.

Undefined or non-finite evaluations are never interpreted as valid margins. Parse errors, unsupported expressions, dimension mismatches, origin-gradient failures, domain failures, and numerical errors therefore produce no passing candidate. This fail-closed procedure is applied consistently to all symbolic methods evaluated using the SHGO Verifier.

B Independent Verification Audits

To examine whether the bounded numerical criterion used throughout the paper remains consistent with stronger but substantially more computationally demanding verification procedures, we independently audit system–Lyapunov pairs from our Poly and NonPoly test benchmarks, together with the final SHGO-accepted model outputs. The model-output audit covers all final symbolic candidates that contribute to the reported system-level success rates, combining candidates obtained through direct Solver inference and verification-guided test-time adaptation. We use dReal for both Poly and NonPoly candidates and additionally apply global sum-of-squares (SOS) verification to Poly constructed pairs and model outputs.

B.1 dReal Audit

dReal is a nonlinear SMT solver for real-valued constraints based on δ-complete decision procedures. We use precision $\delta = 10^{-8}$ and a 300-second timeout for each query. An audited pair or candidate is recorded as *Verified* when both Lyapunov-condition violation queries return UNSAT. Outcomes that cannot be conclusively resolved because of δ-sat ambiguity, timeout, preprocessing failure, or backend execution issues are grouped as *Unresolved / incomplete*. A *Confirmed counterexample* is reported only when high-precision reevaluation confirms an actual violation of the original Lyapunov conditions.

Constructed-pair audit. We first audit system–Lyapunov pairs sampled from the Poly and NonPoly test benchmarks, covering both conventional backward construction and solver-aware generation subsets. This audit examines whether the constructed test pairs accepted by the SHGO Verifier remain consistent with the stronger dReal procedure.

Table B.1: Independent dReal audit of system–Lyapunov pairs from the Poly and NonPoly test benchmarks. Outcomes that cannot be conclusively resolved are grouped as unresolved or incomplete.

Data source	N	Verified	Unresolved / incomplete	Confirmed counterexamples
Poly				
Conventional backward construction	200	51	149	0
Solver-aware generation	200	39	161	0
NonPoly				
Conventional backward construction	200	21	179	0
Solver-aware generation	200	38	162	0

No confirmed counterexample is found in any of the four audited subsets. All remaining cases are unresolved or incomplete rather than confirmed violations of the original Lyapunov conditions. The larger number of incomplete NonPoly cases reflects the greater symbolic translation and backend-processing burden introduced by non-polynomial operators. Although the stronger audit cannot

certify every test pair within the prescribed computational budget, it identifies no confirmed validity failure in either Poly or NonPoly.

Model-output audit. We further audit all SHGO-accepted model outputs that contribute directly to the reported system-level success rates. For each family, the audited set combines candidates accepted during direct Solver inference with those recovered through verification-guided test-time adaptation. The purpose is to assess whether the final accepted predictions remain consistent with the stronger verifier. In particular, this audit includes candidates whose generation is influenced by verifier-derived test-time feedback and therefore directly examines whether such optimization produces outputs that remain consistent with an independent stronger verifier.

Table B.2: Independent dReal audit of all SHGO-accepted SEVA-Lyap model outputs contributing to the reported system-level success rates. Each family combines candidates obtained through direct inference and verification-guided test-time adaptation.

Family	N	Verified	Unresolved / incomplete	Confirmed counterexamples
Poly	368	102	266	0
NonPoly	298	138	160	0

Across all 666 final model outputs contributing to the reported SEVA-Lyap success rates, dReal finds no confirmed counterexample. Specifically, dReal verifies 102 of 368 Poly outputs and 138 of 298 NonPoly outputs, while the remaining cases are unresolved or incomplete under the stronger audit rather than confirmed violations. Because this evaluation covers the complete set of accepted final predictions, including candidates recovered through test-time adaptation, it provides direct evidence that no detectable validity failure is introduced among the reported successful outputs by verifier-guided test-time optimization.

B.2 Global SOS Audit

We additionally perform a global SOS audit on Poly test pairs and final Poly model outputs. The direct SOS formulation used here applies to polynomial inputs and is therefore restricted to Poly. Rather than searching for a new Lyapunov function, the audit directly tests the candidate V by forming

$$p_1(x) = V(x) - V(0), \qquad p_2(x) = -\nabla V(x)^\top f(x), \tag{B.1}$$

and checking whether both admit SOS representations

$$p_i(x) = \sum_j q_{ij}(x)^2. \tag{B.2}$$

For constructed test pairs, $V = V_{\mathrm{GT}}$, whereas for model outputs, V is the final SHGO-accepted Solver prediction. Using the standard Gram-matrix formulation,

$$p_i(x) = z_i(x)^\top Q_i z_i(x), \qquad Q_i \succeq 0, \tag{B.3}$$

the SOS conditions are converted into coefficient-matching constraints and semidefinite feasibility problems.

We use CVXOPT as the SDP backend, with a 300-second limit for each polynomial and a 660-second hard limit for each pair. A candidate is counted as *SOS pass* when both polynomial tests are reported feasible. Timeouts and backend errors are grouped as *Unresolved / incomplete*, while non-feasible outcomes are reported separately. A non-feasible SOS problem indicates that the corresponding SOS certificate is not established under this formulation and is not, by itself, interpreted as a confirmed violation of the Lyapunov conditions.

Table B.3: Independent global-SOS audit of Poly test pairs and all SHGO-accepted Poly model outputs.

Candidate source	N	SOS pass	Unresolved / incomplete	Non-feasible
Conventional backward construction	200	35	165	0
Solver-aware generation	200	13	187	0
SEVA-Lyap model outputs	368	34	334	0

None of the three audited sources reports an SOS non-feasible result. For the final Poly model outputs, 34 of the 368 accepted candidates receive explicit global SOS certificates, while the remaining 334 cases are unresolved or incomplete under the corresponding computational procedure. The relatively large number of unresolved or incomplete cases reflects the substantially higher computational cost of global SOS verification for the high-degree polynomial expressions considered here.

Taken together, the independent audits provide complementary consistency evidence for the bounded numerical verification used in the main experiments. On the Poly and NonPoly test pairs, dReal finds no confirmed counterexample, and the global SOS audit reports no non-feasible result on Poly. More importantly, the model-output audit extends this evidence to all final symbolic predictions that determine the reported SEVA-Lyap success rates, including candidates recovered through verification-guided test-time adaptation. dReal finds no confirmed counterexample among these outputs, and the global SOS audit likewise reports no non-feasible result across all 368 successful Poly predictions.

At the same time, the unresolved and incomplete cases illustrate the substantially higher computational cost and more limited coverage of these stronger verification procedures. Accordingly, these audits do not replace the bounded SHGO criterion with an unrestricted analytical guarantee. Rather, they provide independent evidence that neither the constructed test pairs nor the final accepted model outputs exhibit a confirmed validity failure under the stronger verification procedures considered here. This supports the practical reliability of the SHGO-based verification protocol as a scalable candidate-acceptance criterion for large-scale symbolic Lyapunov discovery.

C Symbolic Representation and Structured Assembly

C.1 Symbolic Representation

Symbolic expressions are represented in prefix notation. Let $\rho(\cdot)$ denote the serialization of a symbolic expression or all coordinates of a vector field. The two learned modules therefore model

$$p_\theta\left(\rho(Q,G,H) \mid \rho(V_{\mathrm{proper}})\right), \qquad p_\phi\left(\rho(V) \mid \rho(f)\right). \tag{C.1}$$

The Generator representation uses dedicated delimiters for the Q, G, and H groups, while the Solver serializes the system dimension and vector-field coordinates before predicting the target Lyapunov expression. Prompt and padding positions are masked during teacher-forced optimization.

Poly and NonPoly use separate symbolic vocabularies. Integer constants are represented by signed base-1000 chunks, while floating-point constants use a signed scientific-notation representation.

C.2 Structured Construction Interface

SEVA-Lyap retains the structured backward construction while making its key symbolic choices learnable through the $Q/G/H$ interface. The Generator produces the symbolic components Q, G, and H, while the validity-preserving transformations and final assembly remain deterministic. Table C.1 summarizes the role of each component and its correspondence to the construction described in Sec. 3.2.

Table C.1: Roles of the $Q/G/H$ interface in structured backward construction.

Component	Functional role	Relation to backward construction
V_{proper}	Positive-definite base structure	Provides the initial Lyapunov backbone
Q	Symbolic terms enriching the Lyapunov function	Centered and squared to form additional Lyapunov terms
G	Weights for gradient-orthogonal directions	Controls motion along directions that leave V unchanged
H	Dissipative factors along state dimensions	Controls the dissipative contribution along each state dimension
Orthogonal basis	Directions orthogonal to ∇V	Constructed deterministically from the Lyapunov gradient
Assembler	Combines the structured components into (f, V_{GT})	Deterministically constructs the target and dynamics

C.3 Deterministic Assembly

For the generated group

$$Q = \{q_k\}_{k=1}^{m_q}, \tag{C.2}$$

the assembler first centers each symbolic term at the equilibrium,

$$q_k(x) \leftarrow q_k(x) - q_k(0), \tag{C.3}$$

so that $q_k(0) = 0$. The additional Lyapunov terms are then constructed as

$$V_{\text{cross}}(x) = \sum_{k=1}^{m_q} q_k^2(x). \tag{C.4}$$

The assembled target is therefore

$$V_{\text{GT}}(x) = V_{\text{proper}}(x) + V_{\text{cross}}(x). \tag{C.5}$$

For the generated group

$$H = \{h_i\}_{i=1}^{n}, \tag{C.6}$$

the assembler similarly centers each dissipative factor,

$$h_i(x) \leftarrow h_i(x) - h_i(0), \tag{C.7}$$

before using $h_i^2(x)$ in the dissipative component. For

$$G = \{g_j\}_{j=1}^{m_g}, \tag{C.8}$$

each $g_j(x)$ weights a direction $e_j(x)$ orthogonal to the Lyapunov gradient, satisfying

$$e_j(x)^\top \nabla V_{\text{GT}}(x) = 0. \tag{C.9}$$

The resulting vector field has the form

$$f(x) = -\sum_{i=1}^{n} h_i^2(x) \partial_i V_{\text{GT}}(x) \mathbf{u}_i + \sum_{j=1}^{m_g} g_j(x) e_j(x), \tag{C.10}$$

which yields

$$\mathcal{L}_f V_{\text{GT}}(x) = -\sum_{i=1}^{n} h_i^2(x) \left(\partial_i V_{\text{GT}}(x)\right)^2 \leq 0. \tag{C.11}$$

Thus, the Generator learns the symbolic components Q, G, and H that determine the construction, while centering, squaring, gradient-orthogonal basis construction, and final system assembly remain deterministic. Generated pairs are subsequently checked by the SHGO Verifier before entering the evolving training buffer.

D TRAINING AND INFERENCE IMPLEMENTATION

D.1 OFFLINE INITIALIZATION

Before solver-aware data evolution, the Generator and Solver are warm-started independently on a filtered corpus of records produced by backward construction. The Generator learns

$$V_{\text{proper}} \to (Q, G, H), \tag{D.1}$$

while the Solver learns

$$f \to V. \tag{D.2}$$

Both modules are optimized with teacher-forced cross-entropy.

The Generator initialization objective is

$$\mathcal{L}_G^{\text{init}} = -\mathbb{E}_{\mathcal{D}_0}\left[\log p_\theta\left(Q, G, H \mid V_{\text{proper}}\right)\right], \tag{D.3}$$

and the Solver is initialized using

$$\mathcal{L}_S^{\text{init}} = -\mathbb{E}_{\mathcal{D}_0}\left[\log p_\phi\left(V \mid f\right)\right]. \tag{D.4}$$

The token-level losses are evaluated only on valid target positions; prompt and padding positions are masked. The corresponding data split and optimization settings are reported in Appendix E.1.

D.2 SOLVER-AWARE DATA EVOLUTION

During solver-aware data evolution, the Generator predicts structured $Q/G/H$ components conditioned on V_{proper}. The deterministic assembler described in Appendix C.3 converts these outputs into explicit system–Lyapunov pairs (f, V_{GT}). Generated pairs are then parsed, assembled, canonicalized, deduplicated, and evaluated using the SHGO Verifier described in Appendix A. Only accepted pairs enter the evolving training buffer.

Generator learning combines the supervised objective and the solver-relative policy-gradient objective defined in Sec. 4.2. Supervised updates preserve the Generator's ability to produce valid structured constructions, while the solver-relative signal redistributes generation probability toward systems that remain difficult for the current Solver.

For each accepted pair (f, V_{GT}), the current Solver first produces

$$\widehat{V}_t = S_{\phi_t}(f), \tag{D.5}$$

and the solver-relative difficulty is evaluated from the Lyapunov violation risks of $\widehat{V}_t$ and V_{GT}.

The shared training buffer $\mathcal{B}_t$ combines newly accepted records with historical replay from earlier evolution rounds and offline replay from the warmup corpus, and is used for both Generator and Solver updates. As the Solver improves, previously difficult systems produce weaker feedback, while unresolved regions continue to guide the Generator toward new parts of the capability boundary. The evolution continues until the cumulative number of accepted pairs reaches the target size N_{target} or the maximum number of evolution rounds T is reached. Algorithm 1 summarizes the complete training loop.

D.3 VERIFIER-DERIVED REWARD FOR TEST-TIME ADAPTATION

For an unresolved test system, each sampled candidate τ_j receives a scalar verifier-derived reward r_j. The implemented reward combines discrete verifier status with continuous numerical shaping:

$$r_j = b_j + p(s_j) + s_j^V + s_j^{\dot{V}} + 5a_j^{\text{shape}} - 10^{-3}T_j, \tag{D.6}$$

where T_j is the candidate length in tokens.

The base reward is

$$b_j = \begin{cases} -8, & \text{input-system parsing failure,} \\ -12, & \text{candidate parsing failure,} \\ 20, & \text{internal validity check succeeds,} \\ -4, & \text{otherwise.} \end{cases} \tag{D.7}$$

Algorithm 1 Solver-Aware Data Evolution

Require: Warm-started Generator G_θ, Solver S_ϕ, offline corpus $\mathcal{D}_0$, assembler $\mathcal{A}$, verifier $\mathcal{V}$, target size N_{target}, maximum rounds T

Initialize shared training buffer $\mathcal{B}_0$
$t \leftarrow 0$
while $N_{\text{acc}} < N_{\text{target}}$ **and** $t < T$ **do**
  Sample V_{proper}
  $(Q, G, H) \sim p_\theta(\cdot \mid V_{\text{proper}})$
  $(f, V_{\text{GT}}) \leftarrow \mathcal{A}(V_{\text{proper}}, Q, G, H)$
  Keep verified construction records in $\mathcal{C}_t$
  if $\mathcal{C}_t \neq \emptyset$ **then**
    $N_{\text{acc}} \leftarrow N_{\text{acc}} + |\mathcal{C}_t|$
    Update shared buffer $\mathcal{B}_t$ with accepted records from $\mathcal{C}_t$, historical replay, and offline replay
    Update Generator G_θ with $\mathcal{L}_G^{\text{sup}}$ on $\mathcal{B}_t$
    Predict $\widehat{V} = S_\phi(f)$ for $(f, V_{\text{GT}}) \in \mathcal{C}_t$
    Compute solver-relative rewards r^{diff} from $\widehat{V}$ and V_{GT}
    Update Generator G_θ with $\mathcal{L}_G^{\text{PG}}$
    Update Solver S_ϕ with $\mathcal{L}_S$ on $\mathcal{B}_t$
  end if
  $t \leftarrow t + 1$
end while
$(\theta^\star, \phi^\star) \leftarrow (\theta, \phi)$
return $G_{\theta^\star}, S_{\phi^\star}$

The status-dependent term is

$$p(s_j) = \begin{cases} -2, & s_j = \texttt{grad_nonzero}, \\ -1, & s_j = \texttt{domain_fail}, \\ 0, & \text{otherwise.} \end{cases} \tag{D.8}$$

For continuous local feedback, let

$$\bar{V}_j(x) = V_j(x) - V_j(0), \qquad \dot{V}_j(x) = \mathcal{L}_f V_j(x). \tag{D.9}$$

On a locally sampled state set $\mathcal{X}_j^{\text{loc}}$, the positivity and decrease components are

$$s_j^V = \tanh\left(\min_{x \in \mathcal{X}_j^{\text{loc}}} \bar{V}_j(x)\right), \qquad s_j^{\dot{V}} = \tanh\left(\min_{x \in \mathcal{X}_j^{\text{loc}}} [-\dot{V}_j(x)]\right). \tag{D.10}$$

A complementary sampled-state shaping term is computed from $x_1, \ldots, x_m$:

$$P_j = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}_{\{\|x_i\|_2 \geq 10^{-12}\}} [0.05 - \bar{V}_j(x_i)]_+, \tag{D.11}$$

$$D_j = \frac{1}{m} \sum_{i=1}^{m} [\dot{V}_j(x_i) + 0.2]_+, \tag{D.12}$$

and

$$a_j = \frac{1}{1 + 2.5P_j + D_j}. \tag{D.13}$$

When this component is available,

$$a_j^{\text{shape}} = 2a_j - 1, \tag{D.14}$$

and otherwise its contribution is set to zero.

Algorithm 2 Verification-Guided Test-Time Adaptation

Require: Reusable Solver $S_{\phi^\star}$, test system f, verifier $\mathcal{V}$, group size K, quantile q, adaptation budget B

Decode candidates with $S_{\phi^\star}$
if a candidate passes $\mathcal{V}$ **then**
  return verified candidate
end if
Initialize local Solver $S_\phi \leftarrow S_{\phi^\star}$
while budget B is not exhausted **do**
  $\phi_{\text{old}} \leftarrow \phi$
  Sample $\{\tau_j\}_{j=1}^K \sim \pi_{\phi_{\text{old}}}(\cdot \mid f)$
  Compute verifier-derived rewards $\{r_j\}_{j=1}^K$
  if a candidate passes $\mathcal{V}$ **then**
    Store the verified candidate
    break
  end if
  Form the upper-reward set $\mathcal{E}_q$ using quantile q
  if $\mathcal{E}_q = \emptyset$ **then**
    continue
  end if
  Compute risk-seeking advantages $\{A_j^{\text{RS}}\}$
  Update local Solver S_ϕ with $\mathcal{L}_{\text{RS-GRPO}}$
end while
Reset local Solver to $S_{\phi^\star}$
return verified candidate if found; otherwise failure

D.4 Risk-Seeking GRPO Adaptation

The risk-seeking objective is defined in Sec. 4.3. Here we summarize its instance-level implementation. For each unresolved test system, an instance-specific Solver is initialized from the trained Solver $S_{\phi^\star}$. Each adaptation step samples a group of candidates, evaluates their verifier-derived rewards, and retains the high-reward tail through the group-relative advantage A_j^{RS} defined in the main text.

Rewards are clipped before the group quantile is computed. Candidates at or below the resulting threshold receive zero policy-gradient weight, while upper-tail candidates drive the local RS-GRPO update. The update is skipped when the clipped-reward population standard deviation satisfies

$$\sigma_r < 10^{-3}, \tag{D.15}$$

or when the upper-tail set $\mathcal{E}_q$ is empty.

During adaptation, only the target embedding, Transformer decoder, and output projection are updated; the source embedding and encoder remain fixed. Sampling, verifier evaluation, and RS-GRPO optimization continue until a candidate passes the SHGO Verifier or the per-system adaptation budget is exhausted. Each test system is processed independently: the local Solver is initialized from $S_{\phi^\star}$ and its instance-specific updates are discarded before subsequent test systems. Algorithm 2 summarizes this procedure.

E Hyperparameters and Reproducibility

E.1 Model Architecture

The Generator and Solver use separate Transformer architectures adapted to their respective symbolic generation tasks. Their principal architectural settings are summarized in Table E.1.

Table E.1: Generator and Solver architectures used throughout the experiments.

Hyperparameter	Generator	Solver
Architecture	causal Transformer	encoder–decoder Transformer
Hidden / FFN width	384 / 1,536	640 / 2,560
Layers	6	6 encoder + 6 decoder
Attention heads	6	10
Dropout	0.1	0.0
Positional encoding	sinusoidal	sinusoidal

E.2 Solver-Aware Data-Evolution Configuration

Table E.2 reports the principal training settings for solver-aware data evolution. The main comparison uses the Solver checkpoint obtained at 500K accepted system–Lyapunov pairs.

Table E.2: Principal training configuration for solver-aware data evolution.

Setting	Poly	NonPoly
Generation batch size	128	128
Generator optimizer / base LR	AdamW / 5×10^{-6}	AdamW / 5×10^{-6}
Solver / Generator batch size	32 / 32	32 / 32
Solver offline / history replay	0.20 / 0.10	0.20 / 0.10

E.3 Test-Time Adaptation Configuration

Table E.3 summarizes the principal configuration for verification-guided RS-GRPO adaptation.

Table E.3: Principal configuration for verification-guided RS-GRPO test-time adaptation.

Component	Value	Notes
Group size K	8	Candidates per adaptation group
Upper-tail quantile q	0.75	Top 25% reward region
Reward clip c	5	Before quantile estimation
Advantage tolerance ϵ_A	10^{-6}	Upper-tail threshold
Temperature / top-k	0.9 / 128	Local Solver decoding
Optimizer / learning rate	AdamW / 3×10^{-7}	Local policy optimization
Ratio clip ϵ_{clip}	0.10	RS-GRPO clipping
Reference KL β	0.01	Token-level forward KL
Shaping points / radius	128 / 5	State-based reward shaping
Adaptation budget	600 s	Per unresolved system

E.4 Execution Environment

Experiments were run using NVIDIA A100-SXM4-40GB GPUs and Intel Xeon Platinum 8358 CPUs. The software environment uses Ubuntu 22.04.5 LTS, Python 3.10.12, PyTorch 2.5.1 with CUDA 12.1 and cuDNN 9.1.0, together with NumPy 1.26.4, SymPy 1.13.1, and SciPy 1.10.1. CPU-side symbolic processing and numerical verification are parallelized across worker processes.

E.5 Computational Resources

We additionally profile the computational resources used by solver-aware data evolution and verification-guided test-time adaptation. Table E.4 reports the training cost at the 500K accepted-pair scale and the amortized adaptation cost per unresolved test system.

Table E.4: Computational resources for solver-aware data evolution and test-time adaptation. Data-evolution times at the $500\mathrm{K}$ milestone are linearly scaled from monitored runs containing approximately one million accepted pairs. Test-time adaptation costs are amortized over 120 Poly and 123 NonPoly unresolved systems. GPU time is obtained by integrating sampled GPU utilization over wall time.

Phase	Family	Wall time	GPU time	Peak memory (GiB)
Solver-aware data evolution	Poly	8.50 h	2.64 h	10.99
Solver-aware data evolution	NonPoly	104.93 h	2.25 h	15.02
Test-time adaptation	Poly	26.7 s/system	8.4 s/system	10.22
Test-time adaptation	NonPoly	380.5 s/system	5.3 s/system	5.56

The data-evolution statistics correspond to the $500\mathrm{K}$ training scale used in the main comparison, whereas the test-time adaptation statistics report amortized computational cost per unresolved system.

E.6 Multi-Seed Paired Stability Analysis

We further evaluate SEVA-Lyap across three random seeds to assess the stability of its performance gains. For each seed, we compute the percentage-point improvement of SEVA-Lyap (Direct) and SEVA-Lyap (Full) relative to the corresponding Symbolic Transformer result, and then aggregate these paired improvements across seeds.

Table E.5: Paired improvement in system-level verification success over the Symbolic Transformer across three random seeds. Entries report mean $\pm$ sample standard deviation in percentage points (pp).

Method	Poly (pp)	NonPoly (pp)
SEVA-Lyap (Direct)	$+4.08 \pm 1.59$	$+3.83 \pm 2.02$
SEVA-Lyap (Full)	$\mathbf{+17.38 \pm 1.41}$	$\mathbf{+11.75 \pm 2.17}$

Across the three seeds, SEVA-Lyap (Direct) improves over the Symbolic Transformer by an average of 4.08 pp on Poly and 3.83 pp on NonPoly. With verification-guided test-time adaptation, SEVA-Lyap (Full) increases the average gains to 17.38 pp and 11.75 pp, respectively. These results show that the improvements from both solver-aware data evolution and the complete SEVA-Lyap pipeline are consistently preserved across different random seeds.

F Into-the-Wild Benchmark Construction

To complement the aggregate Poly and NonPoly results reported in the main paper, we detail the construction and filtering of the Into-the-Wild (ITW) benchmark used to evaluate transfer beyond the data construction mechanisms seen during training.

ITW is constructed directly from randomly generated autonomous dynamics, without first sampling or constructing a reference Lyapunov function. It therefore differs fundamentally from the data produced by backward construction and used for supervised learning. Polynomial systems are generated by randomly combining monomials up to degree five, while the NonPoly generator additionally composes polynomial terms with non-polynomial operators such as trigonometric and exponential functions. The resulting systems span dimensions 2–5 for Poly and 2–3 for NonPoly.

Each generated vector field is first shifted so that the origin is an equilibrium,

$$f(x) \leftarrow f(x) - f(0). \tag{F.1}$$

We then apply a lightweight local-instability filter based on the Jacobian

$$J(0) = \left.\frac{\partial f}{\partial x}\right|_{x=0}. \tag{F.2}$$

Systems are rejected when $J(0)$ has an eigenvalue whose real part is larger than 10^{-10}. Records with invalid symbolic expressions, non-finite Jacobians, or failed eigendecompositions are also removed.

This filtering step removes systems that are clearly locally unstable around the origin, but does not certify stability of the retained systems. In particular, no reference Lyapunov function is available and no Lyapunov function search or SHGO-based verification is performed during ITW construction. Each retained record therefore contains only the dynamics f and the associated generation metadata. From the retained pools, we sample without replacement 2,000 Poly and 2,000 NonPoly systems to form the fixed ITW benchmark shared across all compared methods.

G Scaling, Difficulty Evolution, and Cross-Family Transfer

G.1 Scaling and Difficulty Evolution

To further illustrate how model performance changes with training scale, we evaluate direct-inference success at progressively larger accepted-pair checkpoints using the same balanced evaluation protocol.

Table G.1: Aggregate direct-inference success (%) at increasing accepted-pair milestones. Each value combines the two equally sized held-out portions used in the main benchmark.

Accepted pairs	Poly (%)	NonPoly (%)
100K	48.25	48.75
200K	52.00	48.25
500K	76.75	65.50
1M	83.75	71.00

Larger accepted-pair checkpoints provide substantial gains over the early training stage, showing that solver-aware data evolution continues to strengthen the reusable Solver as the training corpus expands.

To examine whether the generated systems themselves become more difficult, the accepted training stream is divided into ten consecutive 50K intervals and evaluated using the fixed Symbolic Transformer baseline. Table G.2 reports fixed-probe success together with expression-complexity statistics for the same sampled systems.

Table G.2: Interval-level difficulty analysis. Each interval contains 200 sampled systems. Probe success is the system-level verification success of the fixed Symbolic Transformer baseline, with 95% Wilson intervals. Dimension, prefix length, and tree-node counts report sample means on the same systems.

Family	Generation interval	Probe success (%)	95% CI	Dim.	Prefix len.	Tree nodes
Poly	0–50K	82.0	[76.1, 86.7]	3.82	82.2	53.6
Poly	50–100K	83.0	[77.2, 87.6]	3.88	132.8	90.6
Poly	100–150K	86.5	[81.1, 90.6]	3.68	109.6	74.4
Poly	150–200K	47.0	[40.2, 53.9]	3.50	72.1	47.1
Poly	200–250K	58.5	[51.6, 65.1]	3.62	89.3	59.3
Poly	250–300K	60.5	[53.6, 67.0]	3.56	88.2	59.0
Poly	300–350K	57.5	[50.6, 64.1]	3.44	83.9	56.0
Poly	350–400K	52.5	[45.6, 59.3]	3.57	89.5	59.9
Poly	400–450K	57.0	[50.1, 63.7]	3.60	87.4	58.1
Poly	450–500K	55.0	[48.1, 61.7]	3.44	87.9	58.9
NonPoly	0–50K	83.0	[77.2, 87.6]	2.56	76.0	47.9
NonPoly	50–100K	80.5	[74.5, 85.4]	2.54	90.0	57.4
NonPoly	100–150K	75.5	[69.1, 80.9]	2.56	95.8	61.4
NonPoly	150–200K	60.0	[53.1, 66.5]	2.54	96.4	59.8
NonPoly	200–250K	64.5	[57.7, 70.8]	2.50	80.0	49.2
NonPoly	250–300K	67.0	[60.2, 73.1]	2.50	75.2	48.2
NonPoly	300–350K	62.5	[55.6, 68.9]	2.48	76.8	50.4
NonPoly	350–400K	57.0	[50.1, 63.7]	2.53	83.0	53.8
NonPoly	400–450K	56.0	[49.1, 62.7]	2.53	83.5	54.5
NonPoly	450–500K	53.5	[46.6, 60.3]	2.52	88.6	57.9

The fixed-probe success declines substantially over generation time, while system dimension and expression size do not increase in parallel. For example, the final Poly interval has a shorter mean prefix and fewer tree nodes than several early intervals despite being markedly harder for the fixed probe, and the same pattern is observed for NonPoly. This suggests that solver-aware data evolution exploits structural properties linking system dynamics and Lyapunov functions that are not captured by simple measures such as dimension or expression length, rather than increasing difficulty merely by generating larger symbolic expressions.

To quantify this overall trend, we additionally fit system-level logistic models using generation interval as an ordered covariate. The adjusted model includes system dimension, prefix length, and symbolic tree size.

Table G.3: System-level logistic trend analysis for fixed-probe success over the ten generation intervals. Coefficients and odds ratios correspond to one $50\mathrm{K}$-accepted-pair increase in generation interval.

Family	Model	Coefficient	95% CI	OR	p-value
Poly	Unadjusted	-0.152	$[-0.184, -0.120]$	0.859	1.6×10^{-20}
Poly	Complexity-adjusted	-0.150	$[-0.188, -0.112]$	0.861	1.0×10^{-14}
NonPoly	Unadjusted	-0.144	$[-0.177, -0.111]$	0.866	8.3×10^{-18}
NonPoly	Complexity-adjusted	-0.177	$[-0.215, -0.139]$	0.838	9.2×10^{-20}

The negative interval effects remain after complexity adjustment, providing quantitative support for the observed increase in solver-relative difficulty.

G.2 Additional Non-Polynomial Evaluation and Cross-Family Generalization Analysis

To broaden the non-polynomial evaluation beyond the main NonPoly benchmark, we first consider a Trig-only NonPoly family using the same balanced 400-system protocol. The Symbolic Transformer reaches 73.25% aggregate success, while SEVA-Lyap reaches 78.75% with direct inference and 86.50% after test-time adaptation.

We then use Poly, NonPoly, and Trig-only NonPoly as distinct source and target families to probe whether the reusable Solver captures structural regularities that transfer beyond the symbolic operator distribution on which it is trained. Specifically, the experiment evaluates whether a Solver trained on one symbolic family can directly solve systems from a different family without additional training or test-time adaptation.

At the 500K accepted-pair milestone, each source-family Solver is directly evaluated on a different fixed 400-system target benchmark using beam size five.

Table G.4: Cross-family generalization matrix at the 500K accepted-pair milestone. Training and evaluation families are different in every setting, directly probing transfer across symbolic operator distributions.

Training family	Evaluation family	Top-1 (%)	Beam/system (%)
Poly	Trig-only NonPoly	27.75	39.50
Poly	NonPoly	20.25	34.25
Trig-only NonPoly	Poly	21.50	34.00
Trig-only NonPoly	NonPoly	39.00	48.75
NonPoly	Poly	27.25	32.50
NonPoly	Trig-only NonPoly	60.25	69.00

The six source–target settings show that the learned Solver retains non-trivial solving capability even when the symbolic operator distribution changes substantially. A Poly-trained Solver can recover SHGO-accepted Lyapunov candidates for dynamics containing trigonometric operators, while a Solver trained only on Trig-only NonPoly systems can solve purely polynomial dynamics. Since no additional optimization is performed on the target family, these results indicate that the learned solving capability is not confined to family-specific symbolic patterns, but can transfer through structural regularities shared across different classes of dynamical systems. Representative verified cases in both directions are shown below.

Trig-only NonPoly $\rightarrow$ Poly. A Solver trained on the Trig-only NonPoly family successfully recovers the following SHGO-accepted candidates for polynomial dynamics:

$$\text{(i)} \quad \begin{cases} \dot{x}_0 = -6x_0^7\left(4x_0^2+1\right), \\ \dot{x}_1 = -8x_1^{17} \end{cases} \qquad V(x) = 3x_0^8 + 19x_1^2$$

$$\text{(ii)} \quad \begin{cases} \dot{x}_0 = -2x_0^{11}\left(3x_0^2+2\right), \\ \dot{x}_1 = -6x_1^7\left(6x_1^6+1\right) \end{cases} \qquad V(x) = 11x_0^2 + x_1^6$$

$$\text{(iii)} \quad \begin{cases} \dot{x}_0 = -18x_0^5x_1^2, \\ \dot{x}_1 = -2x_1^{11}\left(18x_1^2+5\right) \end{cases} \qquad V(x) = 3x_0^6 + 3x_1^8$$

These examples span polynomial structures ranging from high-order decoupled dynamics to cross-variable coupling. Despite being trained only on Trig-only NonPoly systems, the Solver recovers

compact polynomial candidates that pass the SHGO Verifier. This indicates that the learned solving capability is not tied solely to the operator vocabulary of the source family, but can transfer through structural regularities shared across different dynamical systems.

Poly → Trig-only NonPoly. Conversely, the Poly-trained Solver discovers SHGO-accepted Lyapunov candidates for dynamics containing trigonometric operators:

$$\text{(i)} \quad \begin{cases} \dot{x}_0 = -2x_0 \sin^2(x_0)\cos^2(x_1), \\ \dot{x}_1 = -108x_1^5 \sin^2(x_1) \end{cases} \qquad V(x) = x_0^2 + 2x_1^6$$

$$\text{(ii)} \quad \begin{cases} \dot{x}_0 = -6x_0 \sin^2(x_1)\cos^2(x_0), \\ \dot{x}_1 = -24x_1^7 \sin^2(x_0)\sin^2(x_1) \end{cases} \qquad V(x) = x_0^2 + 3x_1^8$$

$$\text{(iii)} \quad \begin{cases} \dot{x}_0 = 3x_0^7 \left(\cos(4x_0) - 1\right), \\ \dot{x}_1 = -48x_1^5 \left(-\sin(x_1) + \cos(x_0)\right)^2 \end{cases} \qquad V(x) = 5x_0^6 + 4x_1^4 + x_1^2$$

These cases contain state-dependent trigonometric factors and nonlinear cross-variable interactions, yet the Poly-trained Solver still recovers compact polynomial Lyapunov functions. This provides direct evidence that the reusable Solver can generalize beyond the symbolic operator distribution seen during training. In particular, discovering polynomial Lyapunov functions for non-polynomial dynamics suggests that the Solver can exploit structural regularities of stability that are shared across different symbolic families rather than relying only on family-specific surface patterns. Together with the reverse Trig-only NonPoly-to-Poly transfer, these results support the view that solver-aware data evolution can produce reusable solving capability that transfers across structurally different classes of dynamical systems while preserving analytically simple Lyapunov functions.

H Additional Baselines

This section provides implementation details and experimental configurations for four established Lyapunov-synthesis methods: LyZNet, ANLC, FOSSIL 2.0, and SOS synthesis.

LyZNet. LyZNet provides neural learning and verification tools for Lyapunov analysis of nonlinear dynamical systems. It learns neural representations of Lyapunov functions from sampled system states and employs verification procedures to assess the resulting candidates. In our experiments, we use its polynomial-network configuration and independently evaluate the exported candidates using the SHGO Verifier.

ANLC. ANLC is a neural framework for Lyapunov-based stability and control synthesis using sampled states and falsification feedback. For the autonomous systems considered here, our adapter evaluates the supplied dynamics without introducing a learned control input. The resulting Lyapunov expression is exported for independent evaluation using the SHGO Verifier.

FOSSIL 2.0. FOSSIL 2.0 synthesizes Lyapunov functions through an iterative learner–verifier loop. A neural learner proposes a Lyapunov candidate, while the verifier searches for counterexamples that guide subsequent updates. Our adapter supplies the fixed autonomous dynamics and exports the resulting candidate for independent evaluation using the SHGO Verifier.

SOS synthesis. SOS synthesis searches for a polynomial Lyapunov function by expressing polynomial non-negativity requirements as sum-of-squares constraints. For a fixed candidate degree, these constraints yield a semidefinite feasibility problem. We search increasing even degrees under fixed resource limits and retain explicit candidates for verification. The present formulation

Table I.1: Representative challenging-system results. Each reported expression passes the SHGO Verifier; "–" denotes that no candidate passes the SHGO Verifier within the prescribed inference budget.

System	Dynamics $\dot{x} = f(x)$	Symbolic Transformer	SEVA-Lyap
Zou-F3B	$\dot{x}_0 = -8x_0x_1^2 - 10x_1^4,$ $\dot{x}_1 = 3x_1^2 - 8x_1^3 - 8x_1,$ $\dot{x}_2 = -x_2$	–	$4x_0^2 + 9x_1^2 + 2x_1^4 + x_2^2$
Lorenz-stable	$\dot{x}_0 = -10x_0 + 10x_1,$ $\dot{x}_1 = -x_0x_2 + \frac{1}{2}x_0 - x_1,$ $\dot{x}_2 = x_0x_1 - \frac{8}{3}x_2$	–	$x_0^2 + 4x_1^2 + 3x_2^2$
AEK20-$\theta = \frac{1}{4}$	Let $s = \sin(1/4)$ and $c = \cos(1/4)$: $\dot{x}_0 = -2sx_0^5 + 2cx_0^4x_1 - 4sx_0^3x_1^2$ $\quad - 4cx_0^2x_1^3 + 2sx_0x_1^4 - 2cx_1^5,$ $\dot{x}_1 = 2cx_0^5 + 2sx_0^4x_1 + 4cx_0^3x_1^2$ $\quad - 4sx_0^2x_1^3 - 2cx_0x_1^4 - 2sx_1^5$	–	$V(x) = x_0^6 + x_1^6 - \cos(\pi/4)$
FOSSIL-Linear0	$\dot{x}_0 = -x_0 - x_1,$ $\dot{x}_1 = x_0$	$14x_0^2 + 18x_0x_1 + 14x_1^2$	$4x_0^2 + 4x_1^2$

is restricted to polynomial dynamics and therefore does not directly apply to the general NonPoly operator class.

Configurations. The principal search configurations and computational budgets used for the four baselines are summarized in Table H.1.

Table H.1: Search configurations and computational budgets for the additional baselines.

Method	Search configuration	Budget
LyZNet	Polynomial-network configuration	25 epochs, 1,200 s/system
ANLC	Width 10, 1,000 states, lr 0.03	50 iter., 1,200 s/system
FOSSIL 2.0	Width 10, 1,000 states, lr 0.03	50 CEGIS iter., 1,200 s/system
SOS synthesis	Degrees $2, 4, 6, 8, 10, 12$, CVXOPT	300 s/degree, 2,400 s/system

I Representative Challenging-System Evaluation

To further evaluate symbolic discovery on representative challenging systems, we select four cases from established Lyapunov benchmarks and published examples. Table I.1 summarizes the system dynamics and the symbolic candidates returned by the Symbolic Transformer and SEVA-Lyap that pass the SHGO Verifier. A dash indicates that no candidate passes the SHGO Verifier within the prescribed inference budget.

Zou-F3B. This three-dimensional high-degree polynomial system appears in the benchmark suite of Zou et al., where it is adopted from the benchmark suite used in prior symbolic Lyapunov work (Zou et al., 2025; Alfarano et al., 2024). Its high polynomial degree and coupled nonlinear terms make it a useful test of symbolic discovery on an established benchmark. The Symbolic Transformer does not produce a candidate that passes the SHGO Verifier within beam size five, whereas SEVA-Lyap returns

$$V(x) = 4x_0^2 + 9x_1^2 + 2x_1^4 + x_2^2$$

as its first decoded candidate. This result demonstrates direct transfer to a high-degree benchmark system outside the solver-aware generation process.

Lorenz-stable. We consider the classical Lorenz dynamics (Lorenz, 1963) with $(\sigma, \rho, \beta) = (10, 0.5, 8/3)$, for which the origin lies in a stable parameter regime. The system contains coupled quadratic nonlinearities and differs substantially from the structured constructions used during training. Neither the Symbolic Transformer nor the initial SEVA-Lyap prediction produces a candidate that passes the SHGO Verifier. After five local adaptation steps, SEVA-Lyap recovers

$$V(x) = x_0^2 + 4x_1^2 + 3x_2^2,$$

which passes the SHGO Verifier over the bounded evaluation domain. This provides a concrete example in which test-time adaptation recovers a compact symbolic candidate after direct inference fails.

AEK20-$\theta = \frac{1}{4}$. This example is an exact $\theta = 1/4$ specialization of the homogeneous vector-field family introduced in prior work on algebraic stability proofs (Ahmadi & Khadir, 2020). The published construction provides a compact rational Lyapunov function for the underlying family, making this example a useful test on a structurally different literature system involving exact trigonometric constants.

The Symbolic Transformer does not produce a candidate that passes the SHGO Verifier within its five decoded beams. SEVA-Lyap instead generates

$$V(x) = x_0^6 + x_1^6 - \cos(\pi/4).$$

Under the centered representation used by the verifier,

$$\bar{V}(x) = V(x) - V(0) = x_0^6 + x_1^6,$$

which passes the SHGO Verifier. This example shows that SEVA-Lyap can recover a compact symbolic Lyapunov function on a published system whose analytical structure differs substantially from the structured training constructions.

FOSSIL-Linear0. FOSSIL-Linear0 is a two-dimensional continuous-time linear system from the official FOSSIL benchmark suite (Edwards et al., 2024). Its simple structure also allows the discovered Lyapunov functions to be checked analytically. The Symbolic Transformer returns

$$V_{\mathrm{Tr}}(x) = 14x_0^2 + 18x_0x_1 + 14x_1^2,$$

whose quadratic form is positive definite and whose Lie derivative is

$$\dot{V}_{\mathrm{Tr}} = -10x_0^2 - 18x_0x_1 - 18x_1^2 < 0$$

for every nonzero state. SEVA-Lyap initially fails on this system but subsequently discovers

$$V_{\mathrm{SEVA}}(x) = 4x_0^2 + 4x_1^2, \qquad \dot{V}_{\mathrm{SEVA}} = -8x_0^2 \leq 0.$$

The zero-derivative set contains no nontrivial invariant trajectory, so the recovered function also establishes asymptotic convergence. This example gives a directly checkable instance in which test-time adaptation recovers a Lyapunov candidate after unsuccessful direct inference, with validity additionally established analytically.