Analytical Lyapunov Function Discovery: An RL-based Generative Approach

Haohan Zou [*1] **Jie Feng** [*1] **Hao Zhao** [2] **Yuanyuan Shi** [1]

Abstract

Despite advances in learning-based methods, finding valid Lyapunov functions for nonlinear dynamical systems remains challenging. Current neural network approaches face two main issues: challenges in scalable verification and limited interpretability. To address these, we propose an end-to-end framework using transformers to construct analytical Lyapunov functions (local), which simplifies formal verification, enhances interpretability, and provides valuable insights for control engineers. Our framework consists of a transformer-based trainer that generates candidate Lyapunov functions and a falsifier that verifies candidate expressions and refines the model via risk-seeking policy gradient. Unlike Alfarano et al. (2024), which utilizes pre-training and seeks global Lyapunov functions for low-dimensional systems, our model is trained from scratch via reinforcement learning (RL) and succeeds in finding local Lyapunov functions for *high-dimensional* and *non-polynomial* systems. Given the symbolic nature of the Lyapunov function candidates, we employ efficient optimization methods for falsification during training and formal verification tools for the final verification. We demonstrate the efficiency of our approach on a range of nonlinear dynamical systems with up to ten dimensions and show that it can discover Lyapunov functions not previously identified in the control literature. Full implementation is available on Github.

1. Introduction

A Lyapunov function is an energy-like function used to certify stability of dynamical systems. A sufficient condition for stability is that the Lyapunov function decreases along system trajectories. Lyapunov functions also play a central role in controller design, providing formal guarantees of closed-loop stability and robustness Khalil (2002). However, designing a Lyapunov function for nonlinear systems has long been considered more of an 'art' than a science, even for stable dynamics, due to its inherent complexities. Motivated by this challenge, we have witnessed great interest in the development of computational algorithms for Lyapunov function construction. McGough et al. (2010) employed an evolutionary algorithm for the symbolic computation of Lyapunov functions, but the exponential growth search space impedes its scalability. Alternatively, sum-of-squares (SOS) methods reformulate the task as a semidefinite program (SDP) that certifies stability with polynomial candidates (Papachristodoulou & Prajna, 2005a;b; Ahmadi & Majumdar, 2016; Dai & Permenter, 2023). However, handling local constraints or non-polynomial dynamics requires auxiliary variables and extra (in)equality constraints (Papachristodoulou & Prajna, 2005a;b), which leads to scalability issues of the SOS methods for real-world problems. Moreover, the theoretical result Ahmadi et al. (2011) on asymptotic Lyapunov stability shows that even some very simple globally asymptotically stable dynamics may not agree with a polynomial Lyapunov function of any degree.

Recent advances in deep learning have enabled data-driven neural Lyapunov function with formal verification (Chang et al., 2019; Zhou et al., 2022; Wu et al., 2023; Dawson et al., 2023b; Edwards et al., 2024; Wang et al., 2024; Yang et al., 2024). However, these methods face two key challenges: 1) lack of interpretability and 2) high verification costs (Dawson et al., 2023b). Neural networks' black-box nature limits insights into the system's dynamical behavior. Additionally, over-parameterization and nonlinear activations complicate formal verification, leading to scalability issues. Tools like SMT Chang et al. (2019), MIP Wu et al. (2023), and α, β-CROWN Yang et al. (2024) require small, specialized networks to ensure feasible verification times.

Compared with neural Lyapunov functions, analytical Lyapunov functions offer two distinct advantages. First, their symbolic nature offers interpretability, and provides insights for designing stability-guaranteed control policy (Sontag, 1989; Feng et al., 2023a;b; 2024b; Cui et al., 2023b). Second, analytical functions enable efficient verification due to their simplicity and symbolic structure, which seamlessly

[*]Equal contribution [1]Department of Electrical and Computer Engineering, UC San Diego, La Jolla, United States [2]Swiss Federal Institute of Technology in Lausanne. Correspondence to: Haohan Zou <hazou@ucsd.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

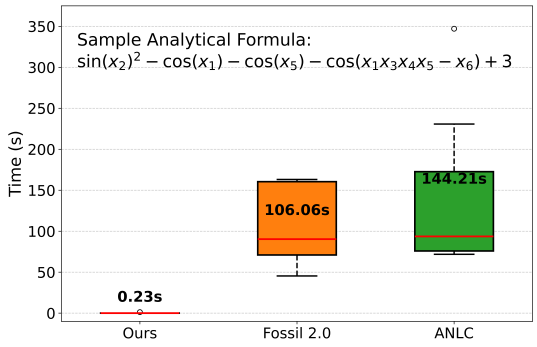

Figure 1. We present the statistics of runtime for single SMT verification (precision e^{-6}, tolerance error e^{-1}) on the 6-D system (Appendix F.4). Each method is validated once per epoch. The red line represents the median. The mean verification time is averaged over one complete training, and the value is displayed in bold. Neural Network in FOSSIL 2.0 Edwards et al. (2024) has a 10-D hidden layer, and Augmented Neural Lyapunov Control (ANLC Grande et al. (2023)) has two 10D hidden layers. Our formulas have a complexity (number of tokens) of 20 or fewer. For ours, the best candidate in each epoch is verified in this motivation example.

integrate with formal verification tools like SMT solvers. This reduces parameter complexity and eliminates the need to verify complex neural network elements like nonlinear activations. Figure 1 highlights the efficiency of verifying analytical expressions compared to neural networks.

In this work, we aim to address the following question:

Can neural networks effectively discover valid analytical Lyapunov functions directly from complex system dynamics?

To tackle this challenge, we introduce an end-to-end framework designed to find analytical Lyapunov functions for nonlinear dynamical systems given in analytical form. Building on the transformer's ability to model complex dependencies Vaswani et al. (2017) and the success of deep symbolic regression methods Holt et al. (2023), our framework deploys a symbolic transformer Kamienny et al. (2022) for Lyapunov function discovery. The transformer's encoder captures system dynamics represented as token sequences derived from the ordinary differential equations (ODEs), while the decoder generates candidate Lyapunov functions by modeling symbolic token distributions. Given the lack of high-quality (local) Lyapunov function datasets, particularly for high-dimensional systems, we propose a reinforcement learning (RL)-based approach to search for Lyapunov functions on a per-system basis, instead of pre-training like Alfarano et al. (2024). We verify Lyapunov conditions by localized sampling in the neighborhoods of minimizers of the candidate expressions, which are most likely to have violations. The identified counterexamples are then incorporated into the training set for further optimization. Our main contributions are summarized as follows:

- We introduce the first RL-based framework for directly discovering analytical Lyapunov functions for nonlinear dynamical systems, bypassing the need for supervised learning with large-scale datasets.
- We propose a novel and efficient policy optimization pipeline integrating global-optimization-based Lyapunov verification, reward design for candidate Lyapunov evaluation, and risk-seeking policy gradient to optimize the symbolic transformer, trainable on machines with limited computation resources.
- We demonstrate the efficiency of our method on various systems, including non-polynomial dynamics like the pendulum, quadrotor, and power system frequency control. Notably, our approach scales to a 10-D system and discovers a valid local Lyapunov function for power system frequency control with lossy transmission lines, that is previously unknown in the literature.

2. Related Work

2.1. Learning-based Lyapunov Function Construction

The field of learning-based Lyapunov function construction is advancing rapidly. Chang et al. (2019) formulated Lyapunov condition violations as the objective, jointly learning a neural Lyapunov function and a linear controller to guarantee stability for a given system, with stability verified via SMT solvers. Zhou et al. (2022) extended this to unknown dynamics with a neural controller. Dai et al. (2021) and Wu et al. (2023) focused on discrete-time systems, using MIP solvers for stability verification, requiring piecewise linear approximations. Yang et al. (2024) applied α, β-CROWN for scalable neural network verification, extending state feedback to output feedback control. However, scalability remains a challenge: SMT solvers handle up to 30 neurons, MIP solvers scale to 200 neurons Dawson et al. (2023a), and α, β-CROWN Yang et al. (2024) is limited to a three-layer architecture (16 neurons per layer).

In contrast to neural Lyapunov functions, Feng et al. (2024c) and Alfarano et al. (2024) derived analytical Lyapunov functions. Feng et al. (2024c) combined a neural network with the symbolic regression package *PySR* (Cranmer, 2023), which approximates the network to produce analytical Lyapunov functions, but the lack of interaction between system dynamics and symbolic regression component limits its potential. Alfarano et al. (2024) pre-trained a transformer on backward- and forward-generated global Lyapunov function datasets, relying on beam search for candidate generation. However, their method cannot adaptively refine the candidate Lyapunov functions if the beam search fails on specific dynamics, and it requires a dataset that is expensive to generate (e.g., thousands of CPU hours for a 5-D dynamics dataset) to achieve adequate generalization during inference.

Furthermore, its emphasis on global stability limits its applicability to real-world, nonpolynomial control systems, which typically only admit local stability. Consequently, an RL-based approach that directly searches Lyapunov functions for a given system is indispensable.

2.2. Symbolic Regression with Generative Model

Symbolic regression is a supervised learning task, searching for an analytical function $f : \mathbb{R}^n \to \mathbb{R}$ that fits $y_i \in \mathbb{R}$ from input $x_i \in \mathbb{R}^n$ (Petersen et al., 2020). With suitable extensions, it can scale to multi-input–multi-output mappings.

RL-based Symbolic Regression. RL-based symbolic regression algorithms Petersen et al. (2020); Costa et al. (2020); Landajuela et al. (2021) employed generative models, typically RNNs, to generate distributions of symbolic tokens representing mathematical operations and variables, from which analytical expressions are sampled. Rewards, evaluating the quality of the sampled expressions, are measured by fitness metrics like RMSE. Due to the non-differentiable step of converting token sequences into symbolic equations, policy gradients are used to optimize the output distributions. Mundhenk et al. (2021) extended this approach by integrating Genetic Programming (GP) to refine generated expressions, improving the overall performance.

Pre-trained Symbolic Regression methods. These methods are inspired by the success of transformers in Natural Language Processing (NLP) tasks. These algorithms contain two steps: 1) pre-train an encoder-decoder network to model $p(f|\mathcal{D})$ on curated datasets by cross-entropy loss, and 2) sample from this distribution during inference via beam search or Monte Carlo Tree Search (MCTS). Methods like Biggio et al. (2021); Kamienny et al. (2022); Bendinelli et al. (2023) rely on beam search without gradient refinement, often yielding suboptimal results for out-of-distribution data. In contrast, Holt et al. (2023) integrates RL-based policy gradient optimization with end-to-end RMSE loss for both pre-training and inference, allowing gradient refinement for unseen datasets during inference. Further, Shojaee et al. (2023); Kamienny et al. (2023) enhance the decoding process (expression generation) by incorporating MCTS with feedback, such as fitting accuracy and equation complexity.

3. Preliminary

Our framework searches for analytical Lyapunov functions for autonomous nonlinear dynamical systems at an equilibrium point. Without loss of generality, we assume the origin to be the equilibrium point.

Definition 3.1 (Dynamical systems)**.** An n-dimensional autonomous nonlinear dynamical system is formulated as

$$\frac{dx}{dt} = f(x)\,, x(0) = x_0, \tag{1}$$

where $f : \mathcal{D} \to \mathbb{R}^n$ is a Lipschitz-continuous vector field, and $\mathcal{D} \subseteq \mathbb{R}^n$ is a set containing the origin that defines the state space. Each $x(t) \in \mathcal{D}$ is a state vector.

Definition 3.2 (Asymptotic stability)**.** A system of (1) is stable at the origin if $\forall\, \epsilon > 0$, there exists $\delta = \delta(\epsilon) > 0$ such that $\|x(0)\| < \delta \implies \|x(t)\| < \epsilon, \forall t \geq 0$. The origin is asymptotically stable if it is stable and δ can be chosen such that $\|x(0)\| < \delta \implies \lim_{t\to\infty} x(t) = 0$ (Khalil, 2002).

Definition 3.3 (Lie derivative)**.** The Lie derivative of a continuously differentiable scalar function $V : \mathcal{D} \to \mathbb{R}$ along the trajectory of (1) is given by

$$L_f V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i}\frac{dx_i}{dt} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x). \tag{2}$$

Proposition 3.4 (Lyapunov functions for asymptotic stability)**.** *Let $x = 0$ be an equilibrium point for (1) and $\mathcal{D} \subseteq \mathbb{R}^n$ be a domain containing the $x = 0$. Let $V : \mathcal{D} \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(0) = 0 \text{ and } V(x) > 0 \text{ in } D\backslash\{0\}, \tag{3a}$$
$$L_f V(x) < 0 \text{ in } D\backslash\{0\}, \tag{3b}$$

then the origin is asymptotically stable.

Definition 3.5 (Lyapunov risk)**.** Consider a candidate Lyapunov function $\tilde{V}$ for system f from Definition 3.1. For a dataset $\mathcal{X} = \{x_1, \cdots, x_N\}$ where $x_i \in \mathcal{D}$, the Lyapunov risk of $\tilde{V}$ Chang et al. (2019) over $\mathcal{D}$ is defined by

$$\mathcal{L}(\tilde{V}) = \frac{1}{N}\sum_{i=1}^{N}\Big(\max(0, L_f\tilde{V}(x_i)) + \max(0, -\tilde{V}(x_i))\Big). \tag{4}$$

4. Proposed Framework

This section introduces our RL-based generative approach, which aims to find an analytical Lyapunov function for a given dynamical system that certifies asymptotic stability following conditions in Proposition 3.4. Successfully identifying such a function guarantees system stability.

The framework, visualized in Figure 2 , consists of three components: 1) a symbolic transformer, parameterized as $\phi = \{\zeta, \theta\}$, for candidate analytical Lyapunov functions generation, where ζ and θ denote the parameters of encoder and decoder, 2) a numerical verifier employing the SHGO Endres et al. (2018) global optimization algorithm for Lyapunov conditions' checking (Proposition 3.4) and counterexamples' feedback, and 3) a risk-seeking policy gradient algorithm optimizing the transformer's parameters based on candidate Lyapunov functions' rewards. To tackle the challenges posed by the exponentially growing search space of complex, high-dimensional systems, our framework integrates Genetic Programming as expert guidance to improve expression quality and training efficiency. We denote $\mathcal{X} \subseteq \mathcal{D}$ as the training set for reward calculation.

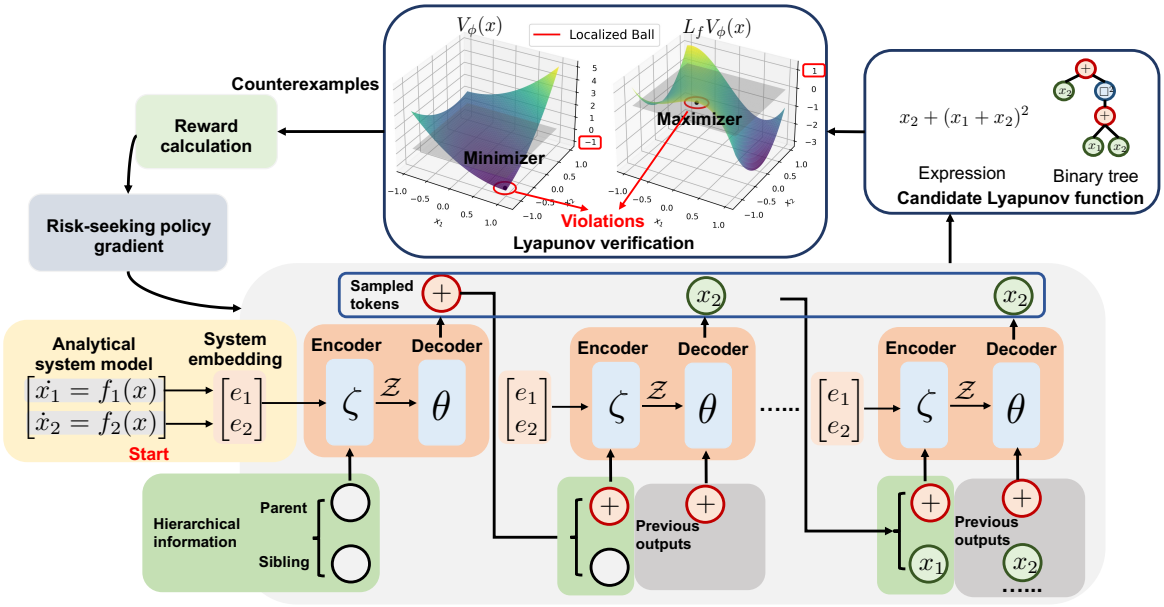

Figure 2. **Framework overview**: The transformer takes embeddings of the dynamical system model as input and generates candidate Lyapunov functions in an autoregressive manner. Hierarchical information is deployed to enhance the model input. For example, when generating the last token x_2, its parent is $+$, and its sibling is x_1. The output is the pre-order traversal of the expression's binary tree. Candidates are verified using a global-optimization-based verification process, with counterexamples added to the training set for reward calculation. The transformer is updated via the risk-seeking policy gradient. The program terminates once a valid expression is found.

4.1. Candidate Lyapunov Function Generation from Symbolic Transformer

Expression Representation. Inspired by the deep symbolic regression frameworks, we use a symbolic transformer model as the backbone. The transformer takes a dynamical system $f(x)$ as input and generates candidate analytical Lyapunov functions $\tilde{V}_\phi$ such that: $\tilde{V}_\phi(x) > 0$ and $L_f\tilde{V}_\phi < 0, \forall\, x \in \mathcal{D}\backslash\{0\}$. Following the expression representation rules in Lample & Charton (2020), our framework represents symbolic transformer models' input and output as sequences of symbolic tokens. Each mathematical expression can be converted into a symbolic expression tree, a binary tree where internal nodes are symbolic operators and terminal nodes (leaves in the tree) are variables or constants. Symbolic operators can be either unary (i.e., one child), such as $\sin, \cos$, or binary (i.e., two children), such as $+, \times$. Furthermore, each symbolic expression tree can be represented as a sequence of node values, either symbolic tokens or numerical coefficients, by its pre-order traversal (i.e., first visiting the parent, then traversing the left child and right child). In this way, each expression obtains a pre-order traversal representation, which can uniquely reconstruct the original expression Petersen et al. (2020). We denote $\tilde{V}_{\phi_i}$ as the i^{th} node value in the pre-order traversal of $\tilde{V}_\phi$'s expression tree, and $\mathcal{L}_s$ as the symbolic library, e.g. $\{+, \times, \log, \sin, x_j\}$, where $\tilde{V}_{\phi_i}$ is sampled from.

Dynamics Tokenization. By Definition 3.1, the symbolic transformer models' input $f(x)$ is composed by n ordinary differential equations $\frac{dx_i}{dt} = f_i(x)$ for $i = 1, \cdots, n$. Each analytical expression $f_i(x)$ can be represented as a sequence of symbolic tokens and numerical coefficients by the pre-order traversal of its expression tree. Concatenated the sequences of pre-order traversal for all $f_i(x)$, with `SOS` (start token) and `EOS` (end token) as separators, we obtain the tokenized dynamics, which is fed into the encoder of symbolic transformer and encoded as a latent vector $\mathcal{F} \in \mathbb{R}^p$. The numerical coefficients are tokenized in two schemes: an integer is represented as a sequence of digits in base $b = 10$ (e.g. 123 is tokenized as $[1, 2, 3]$), and a real number is represented in scientific notation rounded to 4 significant digits (e.g. 3.14 is tokenized as $[3, 1, 4, 0, 10^0]$). A detailed example is illustrated in Figure 3, Appendix A, where we present the symbolic representations of the simple pendulum dynamics in sequences of pre-order traversal.

Candidate Expression Generation. The decoder generates candidate expressions $\tilde{V}_\phi$ in an auto-regressive manner. That is, each token $\tilde{V}_{\phi_i}$ in the pre-order traversal of $\tilde{V}_\phi$ is sampled from the symbolic library $\mathcal{L}_s$ according to conditional distribution $p(\tilde{V}_{\phi_i}|\tilde{V}_{\phi_{1:(i-1)}}, \phi, f(x))$, where $\tilde{V}_{\phi_{1:(i-1)}}$ is the first $(i-1)$ selected symbolic tokens in the pre-order traversal of $\tilde{V}_\phi$. This conditional dependence can be achieved by the decoder, which emits a probability distribution ψ over the symbolic library $\mathcal{L}_s$, conditioned on the previously selected

tokens and input dynamics. Since analytical expression in its pre-order traversal is inherently hierarchical, we also deploy the hierarchical tree state representation (Petersen et al., 2020; Holt et al., 2023). This method enhances the decoder input by concatenating the representations of the parent and sibling nodes with previously selected outputs and the dynamics. Once the token sampling process for $\tilde{V}_\phi$ is complete, we evaluate the function value at origin $\tilde{V}_\phi(0)$ and subtract it from $\tilde{V}_\phi$, ensuring the Lyapunov condition $\tilde{V}_\phi(0) = 0$. Through this process, we can sample a batch of Q candidates $\tilde{\mathcal{V}}_\phi = \{\tilde{V}_\phi^i \sim p(\tilde{V}_\phi|\phi, f(x))\}_{i=1}^{Q}$, which will be verified according to the Lyapunov conditions.

4.2. Verification and Falsification Feedback

Leveraging the analytical nature of candidate Lyapunov functions, efficient methods for symbolic expressions, such as root finding Feng et al. (2024c), can be applied to Lyapunov condition verification and counterexample generation. In this work, we propose a global-optimization-based numerical verification algorithm, using Simplicial Homology Global Optimization (SHGO) (Endres et al., 2018), that effectively checks the Lyapunov conditions around minimizers and feedback counterexamples into the training set $\mathcal{X}$ for reward calculation. SHGO is a constrained global optimization algorithm with theoretical guarantees for convergence. To make the paper self-contained, we present the guarantees in Proposition 4.1. This algorithm identifies the global minimizer over the state space from a set of local minimums, each of which is obtained from a convex subdomain in the feasible search space. Taking advantage of the theoretical results, we employ the SHGO algorithm for counterexample detection in our verification process.

Proposition 4.1 (Convergence Gurantees of SHGO Endres et al. (2018))**.** *For a given continuous objective function f that is adequately sampled by a sampling set of size N_s. If the size of the minimizer pool $\mathcal{M}$ extracted from the directed simplex (a convex polyhedron) $\mathcal{H}$ is $|\mathcal{M}|$. Then any further increase of the sampling size N_s will not increase $|\mathcal{M}|$.*

This result shows that if the initial points are adequately sampled, that is, the union of identified locally sub-convex domains initiated from starting points covers the feasible search space, then the minimizer pool $\mathcal{M}$, which contains all local minimum extracted from the directed simplexes, will contain the global minimizer of the feasible search space. Notably, the required sampling size N_s can be unbounded.

During verification, for a candidate $\tilde{V}_\phi$, SHGO is first applied to identify minimizers x_1^* and x_2^* of $\tilde{V}_\phi$ and its negated Lie derivative $-L_f\tilde{V}_\phi$ in the state space $\mathcal{D}$. These minimizers highlight the regions where Lyapunov conditions are most likely to fail. Next, data points x from neighborhoods around these minimizers, $\mathcal{B}_r(x_1^*)$ and $\mathcal{B}_r(x_2^*)$ where r is a small value relative to $\|\mathcal{D}\|_2$, are sampled to check Lyapunov conditions, i.e. $\tilde{V}_\phi(x) > 0$ and $-L_f\tilde{V}_\phi(x) > 0$ for $x \in \mathcal{D}\backslash\{0\}$. This localized sampling scheme effectively identifies violations within $\mathcal{D}$. Additional random sampling covering approximately 30% of the total data and condition checking across the state space are performed to complement this localized sampling to capture additional potential violations and provide a global assessment. Identified counterexamples $\mathcal{X}_{ce}$ are added to the training set $\mathcal{X}$ for reward calculations. Once a candidate Lyapunov function passes this verification process and does not encounter any violation in $\mathcal{X}$, it indicates a numerically valid solution is found, pending the final formal verification. Appendix B details the verification implementation.

4.3. Risk-Seeking Policy Gradient

The empirical Lyapunov risk $\mathcal{L}(\tilde{V}_\phi)$ in Equation (4) quantifies the violation degree of Lyapunov conditions over a given dataset. Following Petersen et al. (2020); Bastiani et al. (2024), we apply the continuous mapping $g(x) = \frac{1}{1+x}$ and define proposed Lyapunov risk reward as:

$$R(\tilde{V}_\phi) = g\left(\mathcal{L}(\tilde{V}_\phi)\right) = \frac{1}{1 + \mathcal{L}(\tilde{V}_\phi)}, \tag{5}$$

where $\mathcal{L}(\tilde{V}_\phi)$ is measured over training set $\mathcal{X}$. The continuous mapping $g(x)$ bounds the reward value to $[0, 1]$. For candidate expressions that do not incorporate all state variables or are analytically incomplete, we assign their rewards to be 0 to ensure they are effectively penalized.

Given the violation measure for each sampled $\tilde{V}_\phi$ is non-differentiable with respect to the transformer parameters ϕ, we employ the risk-seeking policy gradient to update ϕ end-to-end. The objective of standard policy gradient Williams (1992) is defined to maximize $J_{\text{std}}(\phi) = \mathbb{E}_{\tilde{V}_\phi \sim p(\tilde{V}_\phi|\phi, f(x))}[R(\tilde{V}_\phi)]$, the expectation of the reward function $R(\cdot)$ for candidates' quality evaluation based on the current parameter ϕ. This objective is desirable for problems aiming to optimize the average performance of the policy network. In our task, final performance depends on finding *at least* one valid Lyapunov function that meets the conditions in Proposition 3.4, rather than optimizing for average performance. Consequently, standard policy gradient methods are inadequate due to the misalignment.

In our framework, we adopt risk-seeking policy gradient Petersen et al. (2020) that only focuses on maximizing best-case performance. Let $R_\alpha(\phi)$ as the $1-\alpha$ quantile of the distribution of rewards of sampled candidates under the current policy ϕ. The learning objective of risk-seeking policy gradient, parameterized by α, is formulated as:

$$J_{\text{risk}}(\phi, \alpha) = \mathbb{E}_{\tilde{V}_\phi \sim p(\tilde{V}_\phi|\phi, f(x))}\left[-R(\tilde{V}_\phi) \mid R(\tilde{V}_\phi) \geq R_\alpha(\phi)\right]. \tag{6}$$

Algorithm 1 Training Framework for Analytical Lyapunov Function Discovery via Reinforcement Learning

Input: Dynamics $f(x)$, state space $\mathcal{D}$, quantile α, symbolic library $\mathcal{L}$, batch size Q, max complexity k, radius r.
Output: Valid Lyapunov function $\mathcal{V}^*$ for system $f(x)$.

Initialize the conditional generator with parameters ϕ,
Randomly sample training datapoints $\mathcal{X} = \{x_1, \cdots, x_N\}$ where $x_i \in \mathcal{D}$,
while no valid candidates found **do**
  $\tilde{\mathcal{V}}_\phi \leftarrow \{\tilde{V}_\phi^i \sim p(\tilde{V}_\phi|\phi, f(x))\}_{i=1}^Q$,
  $\tilde{\mathcal{V}}_{gp} \leftarrow$ Genetic Programming$(\tilde{\mathcal{V}}_\phi)$,
  $\tilde{\mathcal{V}} \leftarrow \tilde{\mathcal{V}}_\phi \cup \tilde{\mathcal{V}}_{gp}$,
  $\mathcal{V}^*, \mathcal{X}_{ce} \leftarrow$ verification$(\tilde{\mathcal{V}}, r, \mathcal{D})$, {Verify candidates $\tilde{\mathcal{V}}$, return the valid Lyapunov function $\mathcal{V}^*$ (if any) and counterexamples $\mathcal{X}_{ce}$. Details in App. B. }
  if $\mathcal{V}^*$ is not empty **then**
    Return $\mathcal{V}^*$.
  end if
  $\mathcal{R} \leftarrow \{R(\tilde{V}^i) \; \forall \; \tilde{V}^i \in \tilde{\mathcal{V}}\}$,
  $R_\alpha(\phi) \leftarrow (1-\alpha)$-quantile of $\mathcal{R}$,
  $\phi \leftarrow \phi - \nabla_\phi J_{\text{risk}}(\phi, \alpha)$, {risk-seeking policy gradient update. Equation (6) }
  $\phi \leftarrow \phi - \nabla_\phi \mathcal{L}(\tilde{\mathcal{V}}_{gp})$, {expert guidance loss based on the genetic programming refined Lyapunov functions $\tilde{\mathcal{V}}_{gp}$ to update policy. Equation (7).}
  Concatenate counterexamples $\mathcal{X}_{ce}$ to dataset $\mathcal{X}$.
end while

This objective aims to optimize only the rewards of high-quality candidates from the top $1-\alpha$ quantile.

4.4. Automated Expert Guidance

While the risk-seeking policy gradient algorithm effectively optimizes the model, training efficiency can be enhanced by off-the-shelf tools that further explore the function space based on the transformers. Inspired by Mundhenk et al. (2021), we incorporate a Genetic Programming (GP) component (DEAP Fortin et al. (2012)) into the training paradigm.

The GP algorithm starts with a batch of initial populations (expression trees) and iteratively refines these populations through evolutionary operations: mutation, selection, and crossover, with a pre-defined metric to evaluate the fitness of populations to the task (e.g., MSE for symbolic regression task). Within our framework, we feed the latest batch of generated candidates $\{\tilde{V}_\phi^i \sim p(\tilde{V}_\phi|\phi, f(x))\}_{i=1}^Q$ from the decoder into the GP module as the starting population[1], refine these expressions through evolutionary operations with Lyapunov risk reward as the measure of fitness, and obtain a batch of refined expressions. We select an 'elite set' of the refined expressions $\tilde{\mathcal{V}}_{gp} = \{\tilde{V}_{gp}^i \sim \text{GP}(\tilde{V}_\phi)\}_{i=1}^G$, regard them as target expressions, and optimize the transformer model in a supervised learning manner, maximizing the probability that the generated token matches the reference tokens from $\tilde{V}_{gp}$, with the following expert guidance loss:

$$\mathcal{L}(\tilde{\mathcal{V}}_{gp}) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{k_i}R(\tilde{V}_{gp}^i)\sum_{j=1}^{k_i} -\log\left(p(\tilde{V}_{gp_j}^i|\tilde{V}_{gp_{1:(j-1)}}^i, \phi, f(x))\right), \quad (7)$$

where G is the number of expressions in $\tilde{\mathcal{V}}_{gp}$, and k_i is the complexity (number of symbolic tokens in the pre-order traversal) of $\tilde{V}_{gp}^i$. Each expression $\tilde{V}_{gp}^i$ is weighted by its Lyapunov risk reward in $\mathcal{L}$. Algorithm 1 summarizes the training process, with more details in Appendix D. The GP solutions explore the characteristics of Lyapunov functions that have not been captured by the transformer yet and effectively guide the transformer.

Remark 4.2 (Exploration–exploitation trade-off). In our framework, exploration arises during candidate expression generation and through GP's evolutionary operations, while exploitation is driven by the risk-seeking policy gradient and the expert-guided loss.

5. Experiment

We validate the proposed algorithm across a variety of nonlinear dynamics by finding their local Lyapunov functions at the equilibrium point to verify their stability, where the systems are autonomous (or closed-loop systems with known feedback control laws). We use *dReal* Gao et al. (2013) SMT solver for final verification of found Lyapunov functions, with a numerical tolerance error $\epsilon = e^{-3}$ and precision $\delta = e^{-12}$, over the state space, i.e. $V(x) > \delta$ and $L_f V(x) < -\delta$ over $\mathcal{D} \backslash \mathcal{B}_\epsilon(0)$. The excluded ball $\mathcal{B}_\epsilon(0)$ is to avoid numerical issues, which is a common practice for SMT-based formal verification (Chang et al., 2019). Global stability can be determined through further expert analysis; for instance, if the Lie derivative is a negation of SOS, it is sufficient to establish global stability.

5.1. Experimental Setting

Test Dynamics. We categorize our collected test dynamical systems into two kinds: 1). Polynomial Systems, and 2). Non-polynomial Systems, where three polynomial systems are adopted from Alfarano et al. (2024) (Appendices F.2 & F.3), and others come from real-world examples. Detailed information about systems is summarized in Appendices F and G. The dimension of these test systems ranges up to 10.

Implementation Details of Framework. Detailed explanation for all components in our framework is presented in Appendices A (Transformer), B (Global-optimization-based

[1] Without a good initial population from the transformer, GP algorithms itself face significant challenges in directly finding valid Lyapunov functions for high-dimensional systems due to the exponentially growing search space. Appendix H.3.

Table 1. Performance and time consumption of our method on test dynamics. 'App.' refers to Appendix.

Dynamics	Runtime	Ver.[a]	Found Lyapunov Functions	Stab‡	Succ % ‡
2-D Polynomial Sys (App. F.2)	68s	2 ms	$V = 9x_1^2 + 2x_2^2$	l.a.s.	100
2-D Van Der Pol (App. F.1)	126s	1 ms	$V = x_1^2 + x_2^2$	l.a.s.	100
2-D Simple Pendulum (App. G.1)	288s	1 ms	$V = 2(1 - \cos(x_1)) + x_2^2$	l.a.s.	100
3-D Polynomial Sys (App. F.3)	112s	1 ms	$V = 9x_1^2 + x_2^2 + x_3^2$	l.a.s.	100
3-D Trig Dynamics (App. G.2)	157s	1 ms	$V = 1 - \cos(x_1)^2 + x_2^2 + \sin(x_3)^2$	l.a.s.	100
4-D Lossy Power Sys (App. G.5)	3632s	621s	$V = \omega_1^2 + \omega_2^2 + (\omega_2 - \sin(\delta_1) + \sin(\delta_2))^2$	l.a.s.	100
6-D Polynomial Sys (App. F.4)	1667s	7 ms	$V = \sum_{i=1}^{6} x_i^2$	l.a.s.	100
6-D Quadrotor (App. G.4)	3218s	1 ms	$V = \sum_{i=1}^{6} x_i^2$	g.a.s.	80
6-D Lossless Power Sys (App. G.3)	18094s	2 ms	$V = (\sum_{i=1}^{3} \omega_i^2) - 0.5\left(\sum_{i=1}^{3}\sum_{j=1, i\neq j}^{3} \cos(\delta_i - \delta_j) - 1\right)$	l.a.s.	60
9-D Synthetic Sys (App. G.6)	27047s	6.6 s	$V = \left(\sum_{i=1}^{6} x_i^2\right) + \sin(x_7)^2 + x_8^2 - \cos(x_9) + 1$	l.a.s.	60
10-D Polynomial Sys (App. F.6)	64223s	2 ms	$V = \sum_{i=1}^{10} x_i^2$	l.a.s.	60

a. 'Ver.' presents the time consumption for the final verification of the found Lyapunov functions. All found Lyapunov functions passed SMT solver's verification.

‡. 'Stab' means stability. In this column, 'g.a.s' represents globally asymptotically stable, and 'l.a.s.' represents locally asymptotically stable.

‡. 'Succ %' denotes the successful rate of finding a valid Lyapunov function out of 5 random seeds.

Numerical Verification), C (Risk-seeking Policy Gradient), and D (Genetic Programming). The symbolic library $\mathcal{L}_s$ is defined as $\{+, -, \times, \sin, \cos, x_i\}$ in all tests.

Baseline Algorithms. We compare our proposed framework against four baseline algorithms for continuous nonlinear dynamics. *Neural methods*: 1) Augmented Neural Lyapunov Control (ANLC) Grande et al. (2023), and 2) FOSSIL 2.0 Edwards et al. (2024). *Analytical methods*: 3) the transformer-based global Lyapunov search of Alfarano et al. (2024), and 4) SOS methods via SOSTOOLS (Matlab) (Papachristodoulou et al., 2013). We use the formulation of Papachristodoulou & Prajna (2005a) on polynomial systems, and apply the recasting technique of Papachristodoulou & Prajna (2005b) to convert non-polynomial dynamics into rational form so they can also be handled by SOS.

Both ANLC and FOSSIL 2.0 train a neural Lyapunov function with the empirical Lyapunov risk loss and employ a counter-example guided inductive synthesis (CEGIS) loop for better generalization over the state space $\mathcal{D}$. Alfarano et al. (2024) pre-trains a transformer on global Lyapunov function datasets with beam search for candidate generation. SOS methods formulate Lyapunov functions as the feasible solutions of some semi-definite programming tasks and solve these tasks via convex optimization tools. Details of each baseline can be found in Appendix E.

5.2. Performance Analysis

Table 1 summarizes the runtime, success rate, and discovered Lyapunov functions for a selection of tested nonlinear systems, ranging from 2-D to 10-D, demonstrating the robustness and scalability of our framework. As dimensionality increases, runtime grows with the exponentially expanding search space, reward calculations, SHGO optimization, and genetic programming.

Unlike existing methods that produce neural Lyapunov functions, our framework yields interpretable *analytical* candidates. For example, it correctly identifies the energy function as a valid Lyapunov function for the simple pendulum. Likewise, for the 3-bus power system (Appendix G.3), it discovers the commonly used energy-based storage function for incremental passive systems Weitenberg et al. (2018).

Analytical Lyapunov functions can potentially bypass the need for formal verification. In the 3-D Trig system (Appendix G.2), over the state space $\mathcal{D} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid |x_i| \leq 1.5, \forall\, i \in \{1, 2, 3\}\}$, the positive definiteness of the identified Lyapunov function is evident from its formulation. Moreover, the Lie derivative $L_f V = -2x_2^2 - x_3 \sin(2x_3)$ is directly identifiable as non-positive in $\mathcal{D}$, since $x \sin(x) > 0$ for all $x \in (-\pi, \pi)$. By the invariance principle Khalil (2002), the discovered function certifies the asymptotic stability of the origin in state space $\mathcal{D}$. When direct identification is non-trivial, SMT solvers can efficiently verify

Table 2. Training time and success rate comparison between ours and the neural baselines. The Succ % is the successful rate of finding a valid Lyapunov function within 5 hours out of 5 random seeds. Runtime is the average training time for a successful trial.

Frameworks	2-D Dynamics		3-D Dynamics		6-D Dynamics		8-D Dynamics (App. F.5)	
	Succ %	Runtime	Succ %	Runtime	Succ %	Runtime	Succ %	Runtime
Ours	**100**	165s	**100**	124s	**80**	6290s	**80**	14358s
ANLC	53.3	**1.409s**	46.7	**63.91s**	0	-	0	-
FOSSIL 2.0	80	7.708s	66.7	221s	0	-	0	-

Lyapunov conditions given analytical formulations' simplicity.

5.3. Newly Discovered Lyapunov Function

Despite decades of effort in the control community to identify Lyapunov functions, certain stable dynamics still lack a valid Lyapunov function to directly certify their stability. One example is the lossy frequency dynamics in power systems Chiang (1989); Cui & Zhang (2022), where "lossy" refers to the consideration of energy losses in transmission lines. For simplicity, we focus on a 2-bus (4-D) lossy system with the equilibrium point set at the origin, with detailed descriptions provided in Appendix G.5. Using the proposed method, we successfully discover two local Lyapunov functions valid in the considered region $\mathcal{D} = \{(\delta_1, \delta_2, \omega_1, \omega_2) \in \mathbb{R}^4 \mid |\delta_i| \leq 0.75 \text{ and } |\omega_i| \leq 2 \text{ for } i = 1, 2\}$:

$$V_1(\delta_1, \delta_2, \omega_1, \omega_2) = \sum_{i=1}^{2} \omega_1^2 + \Big(\sin(\delta_2) - \sin(\delta_1) + \omega_2\Big)^2,$$

$$V_2(\delta_1, \delta_2, \omega_1, \omega_2) = \sum_{i=1}^{2} \omega_1^2 + \Big(\sin(\delta_2) - \sin(\delta_1) - \omega_1\Big)^2.$$

Both functions are formally verified by the SMT solver within the defined state space. To the best of our knowledge, these are the first analytical Lyapunov functions used to certify the local stability of a 2-bus lossy power system.

5.4. Comparisons with Baselines

Neural Lyapunov Function Baselines. Table 2 compares our success rate and training runtime with two neural Lyapunov function baselines across various test dynamics. For low-dimensional systems, both baselines achieve notably shorter overall training runtime. However, to achieve efficient verification, these methods rely on relatively small neural networks, which fail to converge on more challenging tasks (e.g., the simple pendulum and 3-D Trig dynamics involving trigonometric terms). Consequently, both baselines exhibit lower overall success rates than our approach.

As dimensionality grows, the complexity of the Lie derivative of a neural Lyapunov function increases significantly following Definition 3.3, creating severe verification bottlenecks for both baselines' counter-example feedback paradigm. Even networks with fewer than 15 neurons per layer may require hours to finish formal verification. In contrast, our method remains robust up to 6-D and 8-D systems, as its simpler analytical formulations allow numerical verification to efficiently identify violation regions. Moreover, for final verification, our candidates can pass the SMT solver in milliseconds, thanks to term cancellations and algebraic restructuring made possible by their analytical form.

Analytical Lyapunov Function Baselines. We evaluated the pre-trained model of Alfarano et al. (2024) on our benchmarks. Trained solely on globally stable systems with fewer than six states, it produced valid Lyapunov functions only for the low-dimensional examples in Appendices F.1, F.2, F.3, & G.1, which have global stability guarantees, and failed on every benchmark that is only locally stable.

Table 3 compares the training and solving times of our method and the SOS approach on polynomial systems from Appendix F. For low-dimensional systems, SOS finds valid Lyapunov functions more efficiently, with shorter runtimes. However, it fails to certify stability for the higher-dimensional systems in Appendices F.4, F.5, and F.6, which are only locally stable. While SOS methods can scale to 10-dimensional or higher systems for global stability verification, local stability requires additional constraints on the Lie derivative, which significantly impact scalability. For example, in the 6D polynomial system from Appendix F.4, verifying local stability with a degree-2 polynomial candidate introduces hundreds of symbolic terms in the Lie derivative constraint, rendering the problem intractable for SOSTOOLS. More details are provided in Appendix E.4.

To use SOS on non-polynomial systems without relaxing the dynamics, we follow the recasting scheme of Papachristodoulou & Prajna (2005b), replacing each non-polynomial term with auxiliary variables linked by extra (in)equalities. We applied this procedure to the simple pendulum and the 3-D trig system (Appendices G.1 & G.2). SOS did recover Lyapunov certificates in both cases, but at a much higher computational cost than our framework for the 3-D trig example. Three main limitations are revealed during implementation: (i) recasting demands substantial

Table 3. Training/solving time of ours and sum-of-squares (SOS) on polynomial systems, averaged over successful trials. The stable regions (local or global) of the considered dynamics are indicated in smaller font.

Test Systems	**2-D Systems** (App. F.1, local; App. F.2, global.)	**3-D Systems** (App. F.3, global.)	**6-D System** (App. F.4, local.)	**8-D System** (App. F.5, local.)	**10-D System** (App. F.6, local.)
Ours	97s	108s	**1667s**	**14358s**	**64223s**
Sum-of-squares	**0.765s**	**1.503s**	-	-	-

domain expertise and hand-crafted constraints; (ii) added variables and relations greatly increase the computational burden; and (iii) the formulation certifies only stability, not asymptotic stability. For the locally stable 3-D trig system, eliminating the trigonometric terms introduces four auxiliary variables and two equality constraints, pushing the solve time beyond one hour, versus 157s for our method. These limitations undermine SOS's practicality for high-dimensional, non-polynomial dynamics.

Table 4. Training/solving time of ours and sum-of-squares (SOS) on two non-polynomial systems, averaged over successful trials.

Frameworks	**App.** G.1	**App.** G.2
Ours	288s	**157s**
Sum-of-squares	**19.58s**	6163s

5.5. Ablation Studies

Risk-Seeking Quantile α. We compare the performance on the 3-D Trig dynamics without GP refinement under $\alpha = 0.1, 0.5, 1$. With $\alpha = 0.1$, the framework achieves steady convergence, lower variance, and the highest success rate of $\mathbf{66.67}\%$, outperforming $\alpha = 0.5$ (33.33%) and the vanilla policy gradient $\alpha = 1$ (0%). These findings confirm the importance of the risk-seeking strategy (Appendix H.1).

Verification Comparison. We compare three verification methods—1) root-finding (Feng et al., 2024c), 2) the *dreal* SMT solver, and 3) random sampling—against our approach on the 6-D polynomial system. SHGO-based counterexample feedback is an adversarial reward, allowing fast refinement and stronger final guarantees but risking training instability. In contrast, random sampling yields smoother rewards and more stable training, yet it identifies violations less efficiently. We thus blend both methods to balance final performance and training stability. Meanwhile, the SMT solver and root-finding produce mostly mild violations, offering limited optimization guidance (Appendix H.2).

Expert Guidance. We evaluate four settings on the 6-D polynomial system to assess the impact of GP refinement and expert guidance: 1) transformer only, 2) GP only, 3) transformer + GP refinement, and 4) transformer + GP refinement + expert guidance (ours). While the transformer only can achieve a 100% success rate, it requires triple the training time compared to 3) and 4). GP only fails to converge due to its limited understanding of system dynamics. In contrast, GP refinement and expert guidance learning efficiently accelerate transformer parameters' update and enable faster Lyapunov function discovery (Appendix H.3).

6. Conclusion

This work introduces an end-to-end framework for discovering analytical Lyapunov functions for nonlinear dynamical systems. A symbolic transformer, trained with a risk-seeking policy gradient and augmented by genetic programming, proposes candidate expressions; the SHGO global optimizer rapidly verifies them and generates counterexamples during training; and an SMT solver certifies the final Lyapunov candidates. The framework scales to 10-dimensional dynamics and has discovered previously unknown local Lyapunov functions for lossy power system dynamics. It can be extended to other certificate-function discoveries, such as control barrier functions for safety certificates.

Several promising future research directions emerge from this work. One is efficiently incorporating physical constants, such as the gravitational constant and object mass, which could significantly improve generalization—though directly introducing them as variables may add unnecessary complexity. Another is extending the framework to discover control Lyapunov functions, which are essential for designing stabilizing feedback laws. Theoretical analysis of the framework, including proofs of completeness and convergence, would also be valuable. Finally, since constructing large-scale datasets for local Lyapunov functions remains challenging, using our approach to refine pre-trained models (Alfarano et al., 2024) offers an exciting opportunity to further advance Lyapunov function discovery.

Acknowledgment

The authors thank Dr. Huan Zhang at the University of Illinois Urbana-Champaign for helpful discussions. J. Feng is supported by the UC–National Laboratory In-Residence Graduate Fellowship L24GF7923. Y. Shi acknowledges support from NSF ECCS-2200692, DOE grant DE-SC0025495,

and the Schmidt Sciences AI2050 Early Career Fellowship.

Impact Statement

This paper advances learning-based Lyapunov function discovery in control theory by proposing a systematic approach to finding Lyapunov functions for any dynamical system. Once a valid Lyapunov function is found, it guarantees system stability in the state space and offers insights for control engineering. Our method can be applied in energy systems, robotics, transportation, and other control systems. While these improvements have the potential to enhance safety and efficiency, we do not foresee specific negative ethical or societal consequences arising directly from our approach.

References

Ahmadi, A. A. and Majumdar, A. Some applications of polynomial optimization in operations research and real-time decision making. *Optimization Letters*, pp. 709–729, 2016.

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

Alfarano, A., Charton, F., and Hayat, A. Global lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=kOMrm4ZJ3m.

Bastiani, Z., Kirby, M., Hochhalter, J., and Zhe, S. Complexity-aware deep symbolic regression with robust risk-seeking policy gradients, 2024. URL https://openreview.net/forum?id=krJ73n4Pma.

Bendinelli, T., Biggio, L., and Kamienny, P.-A. Controllable neural symbolic regression. In *International Conference on Machine Learning*, pp. 2063–2077. PMLR, 2023.

Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A., and Parascandolo, G. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pp. 936–945. PMLR, 2021.

Bouabdallah, S., Murrieri, P., and Siegwart, R. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pp. 4393–4398 Vol.5, 2004. doi: 10.1109/ROBOT.2004.1302409.

Chang, Y.-C., Roohi, N., and Gao, S. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

Chiang, H.-D. Study of the existence of energy functions for power systems with losses. *IEEE Transactions on Circuits and Systems*, 36(11):1423–1429, 1989. doi: 10.1109/31.41298.

Costa, A., Dangovski, R., Dugan, O., Kim, S., Goyal, P., Soljačić, M., and Jacobson, J. Fast neural models for symbolic regression at scale. *arXiv preprint arXiv:2007.10784*, 2020.

Cranmer, M. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl, May 2023. URL http://arxiv.org/abs/2305.01582. arXiv:2305.01582 [astro-ph, physics:physics].

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

Cui, W., Jiang, Y., and Zhang, B. Reinforcement learning for optimal primary frequency control: A lyapunov approach. *IEEE Transactions on Power Systems*, 38(2):1676–1688, 2023a. doi: 10.1109/TPWRS.2022.3176525.

Cui, W., Jiang, Y., Zhang, B., and Shi, Y. Structured neural-PI control with end-to-end stability and output tracking guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Dai, H. and Permenter, F. Convex synthesis and verification of control-lyapunov and barrier functions with input constraints. In *2023 American Control Conference (ACC)*, pp. 4116–4123. IEEE, 2023.

Dai, H., Landry, B., Yang, L., Pavone, M., and Tedrake, R. Lyapunov-stable neural-network control. In *Proceedings of Robotics: Science and Systems*, 2021.

Dawson, C., Gao, S., and Fan, C. 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, 2023a. doi: 10.1109/TRO.2022.3232542.

Dawson, C., Gao, S., and Fan, C. 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, 2023b. doi: 10.1109/TRO.2022.3232542.

Edwards, A., Peruffo, A., and Abate, A. 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*, pp. 1–10, 2024.

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

Feng, J., Cui, W., Cortés, J., and Shi, Y. Bridging transient and steady-state performance in voltage control: A reinforcement learning approach with safe gradient flow. *IEEE Control Systems Letters*, 7:2845–2850, 2023a. doi: 10.1109/LCSYS.2023.3289435.

Feng, J., Shi, Y., Qu, G., Low, S. H., Anandkumar, A., and Wierman, A. Stability constrained reinforcement learning for decentralized real-time voltage control. *IEEE Transactions on Control of Network Systems*, pp. 1–12, 2023b. doi: 10.1109/TCNS.2023.3338240.

Feng, J., Cui, W., Cortés, J., and Shi, Y. Online event-triggered switching for frequency control in power grids with variable inertia. *IEEE Transactions on Power Systems*, pp. 1–13, 2024a. doi: 10.1109/TPWRS.2024.3523262.

Feng, J., Muralidharan, M., Henriquez-Auba, R., Hidalgo-Gonzalez, P., and Shi, Y. Stability-constrained learning for frequency regulation in power grids with variable inertia. *IEEE Control Systems Letters*, 8:994–999, 2024b. doi: 10.1109/LCSYS.2024.3408068.

Feng, J., Zou, H., and Shi, Y. Combining neural networks and symbolic regression for analytical lyapunov function discovery. In *ICML 2024 Workshop: Foundations of Reinforcement Learning and Control – Connections and Perspectives*, 2024c. URL https://openreview.net/forum?id=Knj78wY9T4.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A. G., Parizeau, M., and Gagné, C. Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1):2171–2175, 2012.

Gao, S., Kong, S., and Clarke, E. M. dreal: An smt solver for nonlinear theories over the reals. In Bonacina, M. P. (ed.), *Automated Deduction – CADE-24*, pp. 208–214, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Grande, D., Peruffo, A., Anderlini, E., and Salavasidis, G. Augmented Neural Lyapunov Control. *IEEE Access*, 11:67979–67986, 2023. doi: 10.1109/ACCESS.2023.3291349.

Grüne, L. Computing lyapunov functions using deep neural networks. *Journal of Computational Dynamics*, 8, 01 2019. doi: 10.3934/jcd.2021006.

Holt, S., Qian, Z., and van der Schaar, M. Deep generative symbolic regression. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=o7koEEMA1bR.

Kamienny, P.-A., d'Ascoli, S., Lample, G., and Charton, F. End-to-end symbolic regression with transformers. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=GoOuIrDHG_Y.

Kamienny, P.-A., Lample, G., Lamprier, S., and Virgolin, M. Deep generative symbolic regression with monte-carlo-tree-search. In *International Conference on Machine Learning*, pp. 15655–15668. PMLR, 2023.

Khalil, H. K. *Nonlinear systems*. Prentice Hall, 3nd edition, 2002.

Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, volume 1. MIT press, 1992.

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

Landajuela, M., Petersen, B. K., Kim, S., Santiago, C. P., Glatt, R., Mundhenk, N., Pettit, J. F., and Faissol, D. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pp. 5979–5989. PMLR, 2021.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017. URL http://arxiv.org/abs/1706.06083.

McGough, J. S., Christianson, A. W., and Hoover, R. C. Symbolic computation of lyapunov functions using evolutionary algorithms. In *Proceedings of the 12th IASTED international conference*, volume 15, pp. 508–515, 2010.

Mundhenk, T. N., Landajuela, M., Glatt, R., Santiago, C. P., faissol, D., and Petersen, B. K. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. In *Advances in Neural Information Processing Systems*, 2021.

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

Papachristodoulou, A. and Prajna, S. *Analysis of Non-polynomial Systems Using the Sum of Squares Decomposition*, pp. 23–43. Springer, Aug 2005b. ISBN 978-3-540-23948-2. doi: 10.1007/10997703_2.

Papachristodoulou, A., Anderson, J., Valmorbida, G., Prajna, S., Seiler, P., and Parrilo, P. A. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. http://arxiv.org/abs/1310.4716, 2013. Available from http://www.cds.caltech.edu/sostools.

Petersen, B. K., Larma, M. L., Mundhenk, T. N., Santiago, C. P., Kim, S. K., and Kim, J. T. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2020.

Shojaee, P., Meidani, K., Barati Farimani, A., and Reddy, C. Transformer-based planning for symbolic regression. *Advances in Neural Information Processing Systems*, 36: 45907–45919, 2023.

Sontag, E. D. A 'universal'construction of artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13 (2):117–123, 1989.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, Z., Andersson, S. B., and Tron, R. Lyapunov neural network with region of attraction search, 2024.

Weitenberg, E., De Persis, C., and Monshizadeh, N. Exponential convergence under distributed averaging integral frequency control. *Automatica*, 98:103–113, 2018.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wu, J., Clark, A., Kantaros, Y., and Vorobeychik, Y. Neural lyapunov control for discrete-time systems. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Yang, L., Dai, H., Shi, Z., Hsieh, C.-J., Tedrake, R., and Zhang, H. Lyapunov-stable neural control for state and output feedback: A novel formulation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=3xPMW9JURD.

Zhou, R., Quartz, T., Sterck, H. D., and Liu, J. Neural lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*, 2022.

Contents of Supplementary Materials

Framework Details:

Baseline Descriptions:

Tested Dynamical Systems:

Ablation Studies:

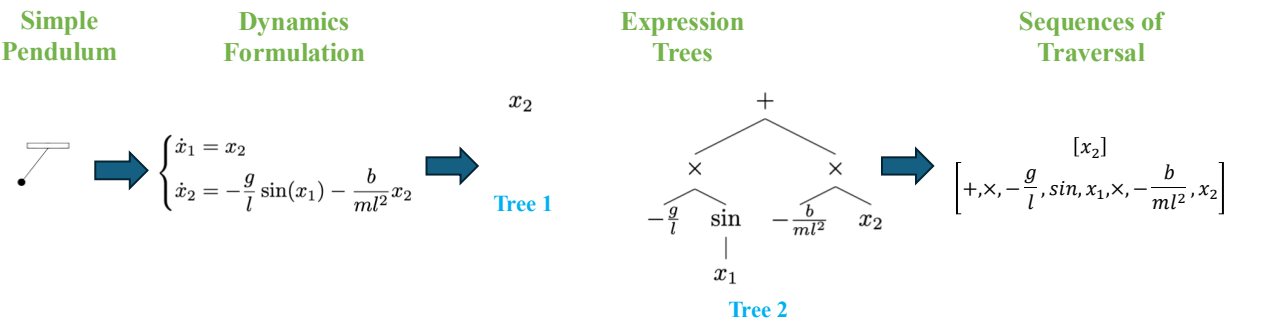

Figure 3. We visualize the dynamics tokenization process of the simple pendulum system. Each analytical formula in the ODE representation of input dynamics is first converted into an expression tree and then represented by the pre-order traversal in symbolic tokens and constant coefficients.

Computation Resources: All experiments in this work were performed on a workstation with an AMD Ryzen Threadripper 2920X 12-Core Processor, an Nvidia RTX 2080Ti GPU 11 GB, and an Nvidia RTX 2080 GPU 8 GB.

A. Symbolic Transformer Model

We outline the details of our symbolic transformer model, a conditional generator for analytical candidate expression generation, comprising two components: 1) an encoder, and 2) a decoder. $\phi = \{\zeta, \theta\}$ denotes the transformer parameters, where ζ and θ denote the parameters of encoder and decoder respectively.

A.1. Encoder Structure

Our framework employs the vanilla transformer encoder from Vaswani et al. (2017) to encode two types of input information: 1) the input system dynamics $f(x)$, encoded into a latent vector $\mathcal{F} \in \mathbb{R}^p$, where $p \in \mathbb{R}$, and 2) the hierarchical tree state representation (Petersen et al., 2020) of the selected tokens, encoded as a latent vector $\mathcal{W} \in \mathbb{R}^k$, where $k \in \mathbb{R}$. The resulting representations are concatenated as $\mathcal{Z}$. Both inputs are expressed as sequences of symbolic tokens and numerical coefficients when fed into the encoder. For damped pendulum example in Figure 3, suppose we have $m = 0.5\,\mathrm{kg}$, $l = 1\,\mathrm{m}$, $g = 9.81$, and $b = 0.1$, the dynamics can be tokenized as: $[\texttt{SOS}, x_2, \texttt{EOS}, \texttt{SOS}, +, \times, -, 9, 8, 1, 0, 10^0, \sin, x_1, \times, -, 2, 0, 0, 0, 10^{-1}, x_2, \texttt{EOS}]$. In this work, we set the embedding dimension to 128, attention head to 2, and applied a 2-layer transformer encoder for system dynamics $f(x)$ encoding and a 3-layer transformer encoder for hierarchical tree state representation encoding.

A.2. Decoder Structure

The decoder of the symbolic transformer model also uses the vanilla transformer decoder, with an additional linear layer to output the token probability ψ over the symbolic library $\mathcal{L}_s$ for token selection. Candidate Lyapunov functions $\tilde{V}_\phi$ are sampled as sequences of symbolic tokens in pre-order traversal. Each symbolic token $\tilde{V}_{\phi_i}$ is sampled autogressively from conditional distribution $p(\tilde{V}_{\phi_i}|\tilde{V}_{\phi_{1:(i-1)}}, \phi, f(x))$. Upon token sampling is complete for $\tilde{V}_\phi$, we subtract $\tilde{V}_\phi(0)$ from the candidate expression to enforce the Lyapunov condition $\tilde{V}_\phi(0) = 0$. In each epoch, a batch of candidate Lyapunov functions $\{\tilde{V}_\phi^i \sim p(\tilde{V}_\phi|\phi, f(x))\}_{i=1}^Q$ is sampled as candidates, which are verified by global-optimization-based numerical verification. In this work, we set the embedding dimension to 128, attention head to 2, and applied a 6-layer transformer decoder for the candidate expression generation. In each epoch, we sample $Q = 500$ expressions as candidates.

B. Global-optimization-based Numerical Verification

For a given dynamics $f(x)$, suppose $\tilde{V}_\phi$ is an invalid analytical candidate Lyapunov function. According to Lyapunov conditions defined in Proposition 3.4, for $x_1^*, x_2^* \in \mathcal{D}$, where x_1^*, x_2^* are the global minimizers of $\tilde{V}_\phi$ and $-L_f\tilde{V}_\phi$ in the state space $\mathcal{D}$, the following two inequalities hold: $\tilde{V}_\phi(x_1^*) \leq 0$ and $L_f\tilde{V}_\phi(x_2^*) \geq 0$. This implies that if $\tilde{V}_\phi$ is invalid, the neighborhoods of x_1^* and x_2^* are highly likely to capture significant violations. Based on this observation, we propose a global-optimization-based numerical verification. This verification identifies minimizers x_1^* and x_2^* by Simplicial Homology

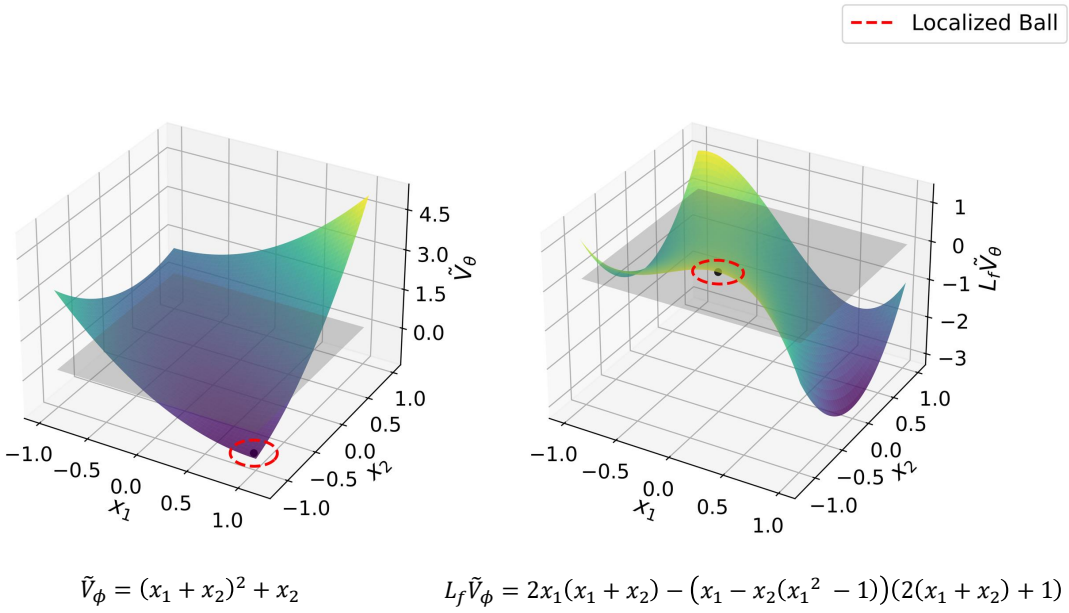

Figure 4. This plot visualizes our proposed verification process on a sampled candidate. $\tilde{V}_\phi = (x_1 + x_2)^2 + x_2$ is a sampled candidate during the training for Van Der Pol Oscillator. Using Simplicial Homology Global Optimization, we first identify the minimizer of the Lyapunov function and the maximizer of the Lie Derivative, the black dots in each graph. Next, data points are sampled in the neighborhoods of the two points, the regions in red circles. For sampled data points that violate the Lyapunov conditions, we feed them into the training set $\mathcal{X}$.

Global Optimization (SHGO), verifies Lyapunov conditions on localized samples in neighborhoods $\mathcal{B}_r(x_1^*)$ and $\mathcal{B}_r(x_2^*)$, and feeds counterexamples back into the training set $\mathcal{X}$. We detail the sampling and condition-checking procedures in Algorithm 2. Figure 4 illustrates this verification process on a sampled candidate, $\tilde{V}_\phi = (x_1 + x_2)^2 + x_2$, for the Van der Pol Oscillator. In the implementation, we initiate with 2048 starting points and iterate 3 times in the SHGO algorithm for the minimizer detection. We tested the number of starting points with values $[1024, 2048, 4096, 8196]$ on various high-dimensional continuous functions, and the setting with 2048 starting points achieves the best efficiency. In datapoint sampling for counter-example identification, for each candidate expression, 800 data points are sampled from each of $\mathcal{B}_r(x_1^*)$ and $\mathcal{B}_r(x_2^*)$, and additional 800 data points are randomly sampled across the state space $\mathcal{D}$.

C. Risk-seeking Policy Gradient

Objective. The standard policy gradient $J_{\text{std}}(\phi) = \mathbb{E}_{\tilde{V}_\phi \sim p(\tilde{V}_\phi|\phi, f(x))}[R(\tilde{V}_\phi)]$ aims to optimize the average performance of a policy given the reward function $R(\cdot)$. However, for the task of Lyapunov function construction, the final performance is measured by identifying a single or a few valid analytical Lyapunov functions that satisfy the Lyapunov conditions. Thus, $J_{\text{std}}(\phi)$ is not an appropriate objective, as there is a mismatch between the objective being optimized and the final performance evaluation metric. To address this misalignment, we adopt risk-seeking policy gradient (Petersen et al., 2020), optimizing the best-case performance via the objective $J_{\text{risk}}(\phi, \alpha)$, as defined in Equation (6). In implementation, we choose $\alpha = 0.1$ in the training for all tested dynamics.

Proposition C.1 (Petersen et al. (2020))**.** *Let $J_{risk}(\phi, \alpha)$ denote the conditional expectation of rewards above the $(1-\alpha)$-quantile $R_\alpha(\phi)$ as in Equation (6). Then the gradient of $J_{risk}(\phi, \alpha)$ is given by:*

$$\nabla_\phi J_{risk}(\phi, \alpha) = \mathbb{E}_{\tilde{V}_\phi \sim p(\tilde{V}_\phi|\phi, f(x))}\left[\left(R_\alpha(\phi) - R(\tilde{V}_\phi)\right) \cdot \nabla_\phi \log p(\tilde{V}_\phi \mid \phi, f(x)) \,\middle|\, R(\tilde{V}_\phi) \geq R_\alpha(\phi)\right]. \tag{8}$$

Algorithm 2 Global-optimization-based Numerical Verification

Input: A set of analytical expressions $\mathcal{V} = \{V^i | \ i = 1, \cdots, Q\}$, radius r, and state space $\mathcal{D}$.
Output: a set of numerically valid candidate $\mathcal{V}^*$, a set of encountered counterexample $\mathcal{X}_{ce}$.

$\mathcal{V}, \mathcal{X}_{ce} \leftarrow \{\}, \{\}$,
for $i = 1$ **to** Q **do**
  $x_1^*, \ x_2^* \leftarrow$ *SHGO*$(V^i, \mathcal{D})$, *SHGO*$(-L_f V^i, \mathcal{D})$, {Identify global minimizers within the state space $\mathcal{D}$}
  $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3 \leftarrow \{x_i | \ x_i \in \mathcal{B}_r(x_1^*)\}, \{x_j | \ x_j \in \mathcal{B}_r(x_2^*)\}, \{x_k | \ x_k \in \mathcal{D}\}$
  Check Lyapunov conditions on $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$,
  if $R(V^i) = 1$ and no counter example found in $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$ **then**
    $\mathcal{V}^* \leftarrow \mathcal{V}^* \cup \{V^i\}$.
  else
    $\mathcal{X}_{ce} \leftarrow \mathcal{X}_{ce} \cup$ identified counterexamples in $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$. {Gather falsification}
  end if
end for
Return $\mathcal{V}^*, \mathcal{X}_{ce}$.

The proposition suggests a Monte Carlo estimate of the gradient of $J_{\text{risk}}(\phi, \alpha)$ from a batch of N samples:

$$\nabla_\phi J_{\text{risk}}(\phi, \alpha) \approx \frac{1}{\alpha N} \sum_{i=1}^{N} \left[\tilde{R}_\alpha(\phi) - R(\tilde{V}_\phi^{(i)}) \right] \cdot \mathbf{1}_{R(\tilde{V}_\phi^{(i)}) \geq \tilde{R}_\alpha(\phi)} \nabla_\phi \log p(\tilde{V}_\phi^{(i)} \mid \phi, f(x)), \tag{9}$$

where $\tilde{R}_\alpha(\phi)$ is the empirical $(1-\alpha)$-quantile of the batch of rewards, and 1_x returns 1 if condition x is true and 0 otherwise. Compared to standard REINFORCE algorithm (Williams, 1992), Equation (9) has two distinct features: (1) it has a specific baseline, $\tilde{R}_\alpha(\phi)$, instead of an arbitrary baseline in standard policy gradients chosen by user; (2) the gradient computation only uses the top α fraction of samples.

Lemma C.2 (Bastiani et al. (2024))**.** *When using the empirical Lyapunov risk in Equation* (4) *as the reward function, the risk-seeking policy gradient is not guaranteed to be unbiased.*

Proof. For a given dataset $\mathcal{X} \subseteq \mathcal{D}$, the empirical Lyapunov risk is a random variable whose value is determined by the random expression sampled from the symbolic transformer model. Let Z denote the negated empirical Lyapunov risk, with its probability density $p(z|\phi)$ depending on transformer parameters ϕ.

Z has range $(-\infty, 0]$. Suppose negated empirical Lyapunov risk is used as the reward function for risk-seeking policy gradient to optimize the average performance of top $(1-\alpha)$-quantile samples. In Equation (9), the risk-seeking policy approximates the gradient of the expectation over the truncated random variable $Z_\alpha = Z \cdot \mathbf{1}_{Z \geq s_\alpha}$ with respect to ϕ, where s_α is the $(1-\alpha)$-quantile of distribution of Z.

$$s_\alpha = \inf\{z : \text{CDF}(z) \geq 1 - \alpha\}.$$

The probability density of Z_α is given by

$$p(z_\alpha) = \frac{1}{\alpha} p(z|\phi) \mathbf{1}_{z \geq s_\alpha}.$$

To effectively penalize the invalid sampled expressions that miss some state variables or are symbolically invalid, their empirical Lyapunov risks are set to be ∞, or a symbolically valid candidate that incorporates all state variables might be less preferable than a symbolically invalid or variable-incomplete expression during the training. Consequently, s_α has a

non-zero probability density of being $-\infty$. If $s_\alpha = -\infty$, then the gradient of the expectation in Equation (8) is:

$$\mathbb{E}[Z_\alpha] = \frac{1}{\alpha}\int_{s_\alpha}^{0} zp(z|\phi)\mathrm{d}z,$$
$$\mathbb{E}[Z_\alpha] = \frac{1}{\alpha}\int_{-\infty}^{0} zp(z|\phi)\mathrm{d}z,$$
$$\nabla_\phi\mathbb{E}[Z_\alpha] = \frac{1}{\alpha}\cdot\nabla_\phi\int_{-\infty}^{0} zp(z|\phi)\mathrm{d}z.$$

Since the integration lower-bound is $-\infty$, the Leibniz rule does not apply, which means interchanging the gradient and the integration changes the result. Therefore, the policy gradient in Equation (8) is no longer guaranteed to be unbiased. □

Reward Design. To optimize the symbolic transformer parameters ϕ such that the decoder generates a valid candidate Lyapunov function $\tilde{V}_\phi$ satisfying Lyapunov conditions, we employ empirical Lyapunov risk as the fitness metric to measure the violation degree of Lyapunov conditions within the state space following Chang et al. (2019). However, as shown in Lemma C.2, directly using the unbounded empirical Lyapunov risk as the reward for risk-seeking policy gradient might introduce bias. To address this issue, we adopt a bounded reward function using the continuous mapping $g(x) = \frac{1}{x}$ (Petersen et al., 2020; Bastiani et al., 2024), defined as:

$$R(\tilde{V}_\phi) = g(\mathcal{L}(\tilde{V}_\phi)) = \frac{1}{1+\mathcal{L}(\tilde{V}_\phi)},$$

where $\mathcal{L}(\tilde{V}_\phi)$ measures the violation degree over the training set $\mathcal{X}$. This design ensures the reward is bounded in $[0, 1]$, avoiding bias in the risk-seeking policy gradient.

Reward Calculation. The reward function in Equation (5) quantifies the degree of violation based on empirical data points. However, as the dimensionality of the input dynamics increases linearly, the training set $\mathcal{X}$ must grow exponentially to maintain precision in the empirical measurement of violation degree, which is impractical for high-dimensional cases. To ensure the reward signal remains a reliable indicator of expression quality in the risk-seeking policy gradient without requiring an excessively large dataset, we incorporate Projected Gradient Descent (PGD) (Madry et al., 2017) into the reward calculation process. Prior to evaluating the reward, PGD is employed on all sampled expressions to efficiently identify a set of "minimizers" in the state space. These minimizers are computed from a randomly sampled set of starting points through PGD for each sampled expression, and the process can be parallelized on a GPU, enabling efficient computation. Though PGD may converge to local minima, it remains effective in identifying a few counterexamples that violate Lyapunov conditions across invalid candidates. These counter-examples are then added to the training set $\mathcal{X}$ for reward calculation of all sampled expressions in the current epoch and are removed immediately after the calculation. This process leverages PGD to capture violation data points in advance, ensuring the quality of the reward signal without the need for an excessively large dataset.

D. Genetic Programming

In the field of symbolic regression, given the large, combinatorial search space, traditional approaches commonly utilize evolutionary algorithms, especially genetic programming (GP) (Koza, 1992), to retrieve analytical expressions that approximate the output values y given input data x. The GP-based symbolic regression operates by evolving the input population of mathematical expressions through evolutionary operations such as selection, crossover, and mutation. A pre-defined fitness metric serves as the objective function to guide the optimization of the population over successive generations. However, for analytical Lyapunov function construction, GP algorithms lack the capability to directly generate Lyapunov functions from the given dynamics and require an initial population that represents potential Lyapunov functions.

As the search space grows exponentially with the expression complexity and the number of states in input dynamics, it is a challenging task even for the symbolic transformer model to search a valid Lyapunov function for complex, high-dimensional systems. Inspired by Mundhenk et al. (2021), we incorporate a GP component into the training framework to complement the symbolic transformer model - the symbolic transformer model outputs a well-behaved initial populations of expressions $\tilde{V}_\phi$, which serve as the starting points for the GP component, and GP component refines $\tilde{V}_\phi$ through evolutionary operations

Algorithm 3 Expert Guidance Loss

Input: 'Elite set' of analytical expressions $\tilde{\mathcal{V}}_{gp}$, input dynamics $f(x)$, and transformer parameters ϕ.
Output: The weighted cross-entropy loss between the transformer model output probability distribution and given refined expressions $\tilde{\mathcal{V}}_{gp}$.

$G \leftarrow |\tilde{\mathcal{V}}_{gp}|$, {Get the size of 'elite set'}
$\mathcal{L} \leftarrow 0$,
for i = 1 **to** G **do**
  $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{k_i} R(\tilde{V}^i_{gp}) \sum_{j=1}^{k_i} - \log \left(p(\tilde{V}^i_{gp_j} | \tilde{V}^i_{gp_{1:(j-1)}}, \phi, f(x)) \right)$, {Calculate the expert guidance loss based on 'elite set' $\tilde{\mathcal{V}}_{gp}$. Equation (7)}
end for
$\mathcal{L}(\tilde{\mathcal{V}}_{gp}) \leftarrow \frac{1}{G}\mathcal{L}$,
Return $\mathcal{L}(\tilde{\mathcal{V}}_{gp})$.

to explore the characteristics of Lyapunov functions that might be overlooked by symbolic transformer. The fitness metric for the GP component is the same as the reward function used in the risk-seeking policy gradient. After each refinement, we select an 'elite set' of the top-performing refined expressions, $\tilde{\mathcal{V}}_{gp}$, based on fitness values. These expressions are treated as ground-truth solutions for the transformer decoder, and transformer parameters ϕ are optimized through the expert guidance loss introduced in Subsection 4.4. In the implementation, the size of 'elite set' $\tilde{\mathcal{V}}_{gp}$ is chosen to be $0.1Q$, where Q is the number of sampled candidate expressions $\tilde{V}_\phi$ in each epoch.

In our framework, we employ three evolutionary operations: mutation, crossover, and selection, within our Genetic Programming component (DEAP (Fortin et al., 2012)). A mutation operator introduces random variations to an expression, such as replacing a subtree of one expression with another randomly generated subtree. A crossover operator exchanges content between two expressions, e.g., by swapping a subtree of one expression with a subtree of another expression, enabling the combination of their features. A selection operator determines which expressions persist into the next population. A common method is tournament selection (Koza, 1992), where a set of l candidate expressions is randomly sampled from the population, and the expression with the highest fitness value is selected. In each iteration of GP evolution, each expression has a probability of undergoing mutation and a probability of undergoing crossover; selection is performed until the new generation's population has the same size as the current generation's population. In empirical experiments, we set the probability of undergoing mutation and crossover to be 0.5, and we adjust the size of the tournament and number of evolutions proportional to the dimension of the input system.

E. Baseline Descriptions

E.1. Augmented Neural Lyapunov Control (ANLC)

The Augmented Neural Lyapunov Control (ANLC) (Grande et al., 2023) combines Artificial Neural Networks (ANNs) with Satisfiability Modulo Theories (SMT) solvers to synthesize stabilizing control laws for the input dynamics $f(x)$ with formal guarantees. The neural network is trained over a dataset of state-space samples to generate candidate control laws and Lyapunov functions, while the SMT solvers are tasked with certifying the Lyapunov conditions of the neural Lyapunov function over a continuous domain and returning a counterexample if the function is invalid. To ease the computationally inefficient verification process in the SMT module, ANLC proposed a discrete falsifier, which discretized the state space for sample selection and evaluation, employed before the SMT call to avoid the frequent calling of the time-consuming SMT falsifier. As the previous learning-based Lyapunov function construction approaches usually initialized the parameters of control policy with pre-computed gains from state-feedback controllers, e.g. Linear-Quadratic Regulators, which requires user time and control expertise to properly perform the initialization process, ANLC instead removes the need of control initialization by its proposed compositional control architecture containing both linear and nonlinear control laws so that the proposed method allows the synthesis of nonlinear (as well as linear) control laws with the sole requirement being the knowledge of the system dynamics. For empirical experiments, we tested the ANLC algorithm for all system dynamics in Appendices F and G. We tested on the Van Der Pol Oscillator and 3D Trig dynamics to get the best hyperparameter setting. In bold, we show the chosen parameters, selected to have the best success discovery rate on Van Der Pol Oscillator and 3D Trig Dynamics.

- lr = $[0.1, \mathbf{0.01}, 0.001]$
- activations = $[(x^2, x^2, x^2), (\tanh, \tanh, x^2), \mathbf{(x^2, x^2, \tanh)}, (\tanh, \tanh, \tanh)]$
- hidden neurons = $[6, \mathbf{12}, 15, 20]$
- data = $[500, \mathbf{1000}, 2000]$
- iteration = $[500, \mathbf{1000}, 2000]$

E.2. FOSSIL 2.0

FOSSIL 2.0 (Edwards et al., 2024) is a software tool for robust formal synthesis of certificates (e.g., Lyapunov and barrier functions) for dynamical systems modelled as ordinary differential and difference equations. FOSSIL 2.0 implements a counterexample-guided inductive synthesis (CEGIS) for the construction of certificates alongside a feedback control law. In the loop of CEGIS, the learner, based upon neural network templates, acts as a candidate to satisfy the conditions over a finite set $\mathcal{D}$ of samples, while the verifier (formal verification tools) works in a symbolic environment that either confirms or falsifies whether the candidate from learner satisfies the conditions over the whole dense domain $\mathcal{X}$. If the verifier falsifies the candidate, one or more counterexamples identified by the verifier are added to the sample set, and the network is retrained. This loop repeats until the verification proves that no counterexamples exist or until a timeout is reached. Similar to the ANLC, in the empirical experiment, we set the hyperparameters based on the Van Der Pol Oscillator and 3-D Trig dynamics and tested for all other dynamics in Appendices F and G.

- lr = $[\mathbf{0.1}, 0.01, 0.001]$
- activations = $[(x^2, x^2), (\tanh, \tanh, x^2), \mathbf{(\tanh, x^2)}]$
- hidden neurons = $[\mathbf{6}, 10, 12]$
- data = $[500, \mathbf{1000}, 2000]$
- iteration = $[25, \mathbf{50}, 100]$

E.3. Global Lyapunov Function Discovery by Pre-trained Transformer

Alfarano et al. (2024) pre-trained a transformer on backward-generated and forward-generated global Lyapunov function datasets. The backward-generated datasets involve sampling arbitrary positive definite functions and deriving corresponding stable dynamics through some specific symbolic designs, while the forward-generated polynomial datasets contain randomly generated dynamics with corresponding Lyapunov functions identified by SOS methods if the system is inherently globally stable. Candidate Lyapunov expressions are sampled using beam search in a token-by-token manner. However, their method cannot adaptively refine the candidate Lyapunov functions if the beam search fails on specific dynamics, and it requires a dataset that is expensive to generate (e.g., thousands of CPU hours for a 5-D dynamics dataset) to achieve adequate generalization during inference. Furthermore, its emphasis on global stability limits its applicability to real-world, nonpolynomial control systems, which typically only admit local stability. Due to the lack of resources of multiple industrial-level GPUs, we contacted the authors of Alfarano et al. (2024) to conduct the evaluation of their pre-trained model on our test systems, which is shown in Section 5.

E.4. Sum-of-Squares (SOS) Methods

SOS methods formulate Lyapunov functions discovery of given dynamics as a semi-definite programming task, where the coefficients of a pre-defined SOS candidate expressions are optimized to satisfy the Lyapunov conditions (hard constraints in the optimization problem) using convex optimization tools. SOS methods are generally applied to polynomial systems for stability analysis. With proper recasting techniques, SOS methods can also be applied to non-polynomial systems.

Definition E.1 (Sum of Squares, Papachristodoulou & Prajna (2005a))**.** For $x \in \mathbb{R}^n$, a multivariate polynomial $p(x)$ is a sum of squares (SOS) if there exist some polynomials $f_i(x), i = 1, \cdots, M$ such that

$$p(x) = \sum_{i=1}^{M} f_i(x)^2.$$

Polynomial systems. By Papachristodoulou & Prajna (2005a), for a given n-dimensional polynomial dynamics $f(x)$ and an integer degree $2d$, to check the globally asymptotical stability of $f(x)$, SOS method aims to find a polynomial $V(x)$ of degree $2d$, such that

1. $V(x) - \sum_{i=1}^{n} \sum_{j=1}^{d} \epsilon_{ij} x_i^{2j}$ is a SOS, where $\sum_{j=1}^{d} \epsilon_{ij} > \gamma, \forall i = 1, ..., n$ with $\gamma > 0$, and $\epsilon_{ij} \geq 0 \; \forall \; i$ and j,

2. $-\frac{\partial V}{\partial x} f(x)$ is a SOS.

For local stability analysis, consider a ball of radius r centered at origin $\mathcal{B}_r(0)$, which can be represented by the semialgebraic set $S = \{x : g(x, r) \geq 0$, where $g(x, r) = r - \sum_{i=1}^{n} x_i^2\}$. We require that the stability condition holds in S. Retaining the same optimization objective and constraints on $V(x)$ as before, a modified constraint on Lie derivative is imposed: $-\frac{\partial V}{\partial x} f(x) - s(x)g(x, r)$ is a SOS for some SOS $s(x)$. If such an $s(x)$ exists, we can establish local stability.

In Section 5, we develop our code based on the `findlyap` function from SOSTOOLS (MATLAB) and issue-16 of SOSTOOLS' official GitHub repo to examine the SOS method on polynomial systems in Appendix F. Table 5 summarizes the experiment results of SOS approach on our polynomial test dynamics.

Table 5. Training\solving time of sum-of-squares (SOS) on test polynomial systems.

Systems	App. F.1	App. F.2	App. F.3 - I	App. F.3 - II	App. F.4	App. F.5	App. F.6
Degree 2d	2	2	2	4	2	2	2
Region	$\mathcal{B}_1(0)$	Global	Global	Global	$\mathcal{B}_1(0)$	$\mathcal{B}_1(0)$	$\mathcal{B}_1(0)$
Runtime	0.697s	0.832s	0.497s	2.509s	-	-	-

Non-Polynomial System. To apply SOS method on non-polynomial system $\dot{z} = f(z), z \in \mathcal{D}_1$, define $x_1 = z$ as the original states and x_2 as the newly introduced variables to recast non-polynomial terms in $f(z)$. Let $x = (x_1, x_2)$. The system dynamics can then be written in rational polynomial forms:

$$\begin{cases} \dot{x}_1 &= f_1(x), \\ \dot{x}_2 &= f_2(x), \end{cases}$$

with constraints $x_2 = F(x_1), G_1(x) = 0, G_2(x) \geq 0$, where F, G_1, G_2 are vectors of functions, to restrict the states of recast dynamics $x = (x_1, x_2)$ equal to the manifold of the original state z.

Define $g(x)$ as the collective denominator of f_1, f_2, and local region of interest as a semialgebraic set:

$$\{(x) \in \mathbb{R}^{n+m} | G_{\mathcal{D}}(x) \geq 0\},$$

where $G_{\mathcal{D}}(x)$ is a vector of polynomials designed to match the original state space. Let $x_{2,0} = F(0)$. Suppose there exist polynomial functions $V(x)$, $\lambda_1(x), \lambda_2(x)$, and SOS polynomials $\sigma_i(x), i = 1, 2, 3, 4$, of appropriate dimensions such that

$$V(0, x_{2,0}) = 0, \tag{10}$$

$$V(x) - \lambda_1^T(x)G_1(x) - \sigma_1^T(x)G_2(x) - \sigma_3^T(x)G_D(x) - \phi(x) \in \text{SOS}, \tag{11}$$

$$-g(x)(\frac{\partial V}{\partial x_1}(x)f_1(x) + \frac{\partial V}{\partial x_2}(x)f_2(x)) - \lambda_2^T(x)G_1(x) - \sigma_2^T(x)G_2(x) - \sigma_4^T(x)G_D(x) \in \text{SOS}, \tag{12}$$

where $\phi(x)$ is some scalar polynomials with $\phi(x_1, F(x_1)) > 0, \forall x_1 \in \mathcal{D}_1 \backslash \{0\}$. If constraints (10), (11), and (12) hold, then $z = 0$ is stable (not necessarily asymptotically stable) (Papachristodoulou & Prajna, 2005b). In Section 5, we develop our code from SOSTOOLS to examine the SOS methods on two non-polynomial systems: simple pendulum and 3-D trig dynamics (Appendices G.1 & G.2).

F. Polynomial Nonlinear Dynamical System

F.1. Van Der Pol Oscillator

Van Der Pol Oscillator is a nonconservative, oscillating system with nonlinear damping (Zhou et al., 2022). The dynamics of the Van Der Pol Oscillator have two state variables, formulated as follows:

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= -x_1 - \mu(1 - x_1^2)\cdot x_2,\end{aligned}$$

where x_1 and x_2 represent the object's position in the Cartesian coordinate, parameter $\mu \in \mathbb{R}^+$ indicates the strength of the damping. Under the state space $\mathcal{D} = \{(x_1, x_2) \in \mathbb{R}^2 \mid |x_i| \leq 1\}$ and setting $\mu = 1$, our proposed method found valid local Lyapunov function $V(x_1, x_2) = x_1^2 + x_2^2$. Other forms of Lyapunov functions for Van Der Pol Oscillator, for example, $V(x_1, x_2) = x_1^2 + x_2(x_1 + x_2)$, are also recovered during the experiments.

F.2. Two-variable-polynomial-system with higher degree

Here we have a polynomial system of two variables with a higher degree, adopted from Alfarano et al. (2024), formulated as:

$$\begin{cases}\dot{x}_1 &= -5x_1^3 - 2x_1\cdot x_2^2,\\ \dot{x}_2 &= -9x_1^4 + 3x_1^3\cdot x_2 - 4x_2^3.\end{cases}$$

Under the state space $\mathcal{D} = \{(x_1, x_2) \in \mathbb{R}^2 \mid |x_i| \leq 1\}$, our proposed method successfully found valid local Lyapunov function $V(x_1, x_2) = 9x_1^2 + x_2^2$.

F.3. Three-variable-polynomial-systems with higher degree

Table 6 describes two polynomial systems of three variables with a higher degree, adopted from Alfarano et al. (2024). Our framework successfully retrieves valid local Lyapunov functions on both examples under the state space $\mathcal{D} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid |x_i| \leq 1\}$.

Table 6. Three-Dimensional Polynomial Example with Higher Degree.

System	Lyapunov function
$\begin{cases}\dot{x}_1 = -3x_1^3 + 3x_1\cdot x_3 - 9x_1\\ \dot{x}_2 = -x_1^3 - 5x_2 + 5x_3^2\\ \dot{x}_3 = -9x_3^3\end{cases}$	$V(x_1, x_2, x_3) = 9x_1^2 + x_2^2 + x_3^2$
$\begin{cases}\dot{x}_1 = -8x_1\cdot x_2^2 - 10x_2^4\\ \dot{x}_2 = -8x_2^3 + 3x_2^2 - 8x_2\\ \dot{x}_3 = -x_3\end{cases}$	$V(x_1, x_2, x_3) = x_1^8\cdot x_2^2\cdot x_3^2 + x_2^2$

F.4. 6-D Polynomial Nonlinear System

This 6-D dynamics consists of three two-dimensional asymptotically stable linear subsystems that are coupled by three nonlinearities with small gains adopted from Grüne (2019). The dynamics are written as:

$$\begin{aligned}
\dot{x}_1 &= -x_1 + 0.5x_2 - 0.1x_5^2, \\
\dot{x}_2 &= -0.5x_1 - x_2, \\
\dot{x}_3 &= -x_3 + 0.5x_4 - 0.1x_1^2, \\
\dot{x}_4 &= -0.5x_3 - x_4, \\
\dot{x}_5 &= -x_5 + 0.5x_6, \\
\dot{x}_6 &= -0.5x_5 - x_6 + 0.1x_2^2.
\end{aligned}$$

Our proposed method is trained over the state space $\mathcal{D} = \{(x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{R}^6 \mid |x_i| \leq 1, \forall i \in \{1, 2, .., 6\}\}$, and is able to find a valid Lyapunov function $V(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$. This dynamics is not globally asymptotically stable since if $x_1, x_2,$ or x_5 has a significantly large value, the perturbations introduced by the small gains will shift the object by a significant amount away from the equilibrium point. By empirical checking, our found Lyapunov function certifies the asymptotical stability of this system over the region $\mathcal{D}' = \{(x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{R}^6 \mid \sum_{i=1}^{6} x_i^2 \leq 500\}$.

F.5. 8-D Polynomial Nonlinear System

This 8-D dynamics consists of four two-dimensional asymptotically stable linear subsystems that are coupled by four nonlinearities with small gains, modified from the above 6D polynomial dynamics. The dynamics are written as:

$$\begin{aligned}
\dot{x}_1 &= -x_1 + 0.5x_2 - 0.1x_5^2, \\
\dot{x}_2 &= -0.5x_1 - x_2, \\
\dot{x}_3 &= -x_3 + 0.5x_4 - 0.1x_1^2, \\
\dot{x}_4 &= -0.5x_3 - x_4, \\
\dot{x}_5 &= -x_5 + 0.5x_6 + 0.1x_7^2, \\
\dot{x}_6 &= -0.5x_5 - x_6, \\
\dot{x}_7 &= -x_7 + 0.5x_8, \\
\dot{x}_8 &= -0.5x_7 - x_8 - 0.1x_4^2.
\end{aligned}$$

Our proposed method is trained over state space $\mathcal{D} = \{(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \in \mathbb{R}^8 \mid |x_i| \leq 1, \forall i \in \{1, 2, .., 8\}\}$, and is able to find a valid Lyapunov function $V(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2$. This Lyapunov function certifies the asymptotical stability of this system over the region $\mathcal{D} = \{(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \in \mathbb{R}^8 \mid \sum_{i=1}^{8} x_i^2 \leq 450\}$. This dynamics is not globally asymptotically stable since if $x_1, x_4, x_5,$ or x_7 has a significantly large value, the perturbations introduced by the small gains will shift the object by a significant amount away from the equilibrium point.

F.6. 10-D Polynomial Nonlinear System

Finally, we extend to the original 10-D polynomial dynamics proposed in Grüne (2019). This 10-D dynamics consists of five two-dimensional asymptotically stable linear subsystems that are coupled by four nonlinearities with small gains. The

dynamics are written as:

$$\begin{aligned}
\dot{x}_1 &= -x_1 + 0.5x_2 - 0.1x_5^2, \\
\dot{x}_2 &= -0.5x_1 - x_2, \\
\dot{x}_3 &= -x_3 + 0.5x_4 - 0.1x_1^2, \\
\dot{x}_4 &= -0.5x_3 - x_4, \\
\dot{x}_5 &= -x_5 + 0.5x_6 + 0.1x_9^2, \\
\dot{x}_6 &= -0.5x_5 - x_6, \\
\dot{x}_7 &= -x_7 + 0.5x_8, \\
\dot{x}_8 &= -0.5x_7 - x_8. \\
\dot{x}_9 &= -x_9 + 0.5x_{10}, \\
\dot{x}_{10} &= -0.5x_9 - x_{10} - 0.1x_4^2.
\end{aligned}$$

Our proposed method is trained over the state space $\mathcal{D} = \{(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) \in \mathbb{R}^{10} \mid |x_i| \leq 1, \forall i \in \{1, 2, .., 10\}\}$, and is able to find a valid Lyapunov function $V(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2 + x_{10}^2$. This Lyapunov function certifies the asymptotic stability of this system over the region $\mathcal{D} = \{(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) \in \mathbb{R}^{10} \mid \sum_{i=1}^{10} x_i^2 \leq 400\}$. This dynamics is not globally asymptotically stable since if $x_1, x_4, x_5,$ or x_9 has a significantly large value, the perturbations introduced by the small gains will shift the object by a significant amount away from the equilibrium point.

G. Non-polynomial Nonlinear Dynamical Systems

G.1. Simple Pendulum

The simple pendulum is a well-known classical nonlinear system that contains two state variables. The dynamics are formulated as follows,

$$\begin{aligned}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= -\frac{g}{l}\sin(x_1) - \frac{b}{m}x_2,
\end{aligned}$$

where x_1 is the angular position from the inverted position, x_2 is the angular velocity, and parameters g, m, l, b are the acceleration of gravity, the mass of the inverted object, the length of the string, and the coefficient of friction, respectively. In experiments, since we don't incorporate the constant generation capability within the training framework, we set $g = 1$, $m = 1\,\text{kg}$, $l = 1\,\text{m}$, and $b = 0.1$. Our proposed method finds the valid Lyapunov function $V = 2 - 2\cos(x_1) + x_2^2$ over the state space: $\mathcal{D} = \{(x_1, x_2) \in \mathbb{R}^2 \mid |x_1| \leq \pi \text{ and } |x_2| \leq 6\}$. This found Lyapunov function has the same analytical structure as the energy function of the inverted pendulum.

G.2. 3-D Trigonometric System

3-D trig dynamics comes from exercise problems in textbook Khalil (2002) whose dynamics are written as follows,

$$\begin{aligned}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= -h(x_1) - x_2 - h(x_3), \\
\dot{x}_3 &= x_2 - x_3,
\end{aligned}$$

where $h(x) = \sin(x) \cdot \cos(x)$. When the state space is $\mathcal{D} = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid |x_i| \leq 1.5, \forall\, i \in \{1, 2, 3\}\}$, the valid local Lyapunov function found by our proposed method is $V(x_1, x_2, x_3) = 1 - \cos(x_1)^2 + x_2^2 + \sin(x_3)^2$, which is consistent to the textbook solution of Lyapunov function for this particular dynamics.

G.3. N-bus Lossless Power System

We test our proposed framework on the N-bus power lossless system (Cui et al., 2023a; Feng et al., 2024a) to examine its ability to handle complex high-dimensional dynamics. Consider θ_i, ω_i as the phase angle and the frequency of bus i,

respectively, the dynamics for each bus are formulated as follows,

$$\dot{\theta}_i = \omega_i,$$
$$m_i\dot{\omega}_i = p_i - d_i\omega_i - u_i(\omega_i) - \sum_{j=1}^{N} B_{ij} \cdot \sin(\theta_i - \theta_j),$$

where m_i is the generator inertia constant, d_i is the combined frequency response coefficient from synchronous generators and frequency sensitive load, and p_i is the net power injection, for each bus $i = 1, \cdots, N$. $B \in \mathbb{R}^{N\times N}$ is the susceptance matrix with $B_{ij} = 0$ for every pair $\{i, j\}$ such that bus i and bus j are not connected, and $u_i(\omega_i)$ is the controller at bus i that adjusts the power injection to stabilize the frequency.

Since the frequency dynamics of the system depends only on the phase angle differences, so we change the coordinates:

$$\delta_i = \theta_i - \frac{1}{N}\sum_{i=1}^{N} \theta_i$$

where δ_i can be understood as the center-of-inertia coordinates of each bus. In our experiment, we test the proposed framework on the 3-bus power system. For simplicity, we set $p_i = 0$, $m_i = 2$, $d_i = 1$, $u_i(\omega_i) = \omega_i$, and $B_{ij} = 1\ \forall\ i \neq j, B_{ii} = 0$. In this case, the equilibrium point for our system is at the origin, i.e. $\delta_i^* = \omega_i^* = 0$, $i = 1, 2, 3$. The state space for our experiment is defined as: $\mathcal{D} = \{(\delta_1, \delta_2, \delta_3, \omega_1, \omega_2, \omega_3) \in \mathbb{R}^6 \mid |\delta_i| \leq 0.75 \text{ and } |\omega_i| \leq 1.2 \text{ for } i = 1, 2, 3\}$. Through our method, we retrieved a valid Lyapunov function $V(\delta_1, \delta_2, \delta_3, \omega_1, \omega_2, \omega_3) = (\sum_{i=1}^{3} \omega_i^2) - 0.5\left(\sum_{i=1}^{3}\sum_{j=1, i\neq j}^{3} \cos(\delta_i - \delta_j) - 1\right)$, which is consistent to the known Lyapunov function presented in (Cui et al., 2023a). The Lie derivative of the identified Lyapunov function can be simplified as $L_f V = -2(\omega_1^2 + \omega_2^2 + \omega_3^2)$. The analytical structure of this found Lyapunov function and invariance principle allows us to easily identify it as a valid Lyapunov function by hand.

G.4. Indoor Micro Quadrotor

For the angular rotations subsystems of the quadrator from Bouabdallah et al. (2004), it has 6 states to describe the angular motion of the quadrotor. The states $x_1, x_3,$ and x_5 describe the roll, pitch, and yaw of the quadrator, and states $x_2, x_4,$ and x_6 represent their time derivatives. With perturbation terms Ω and control inputs $U_1, U_2,$ and U_3, the subsystem can be formulated as follows,

$$\begin{aligned}
\dot{x}_1 &= x_2,\\
\dot{x}_2 &= x_4x_6(\frac{I_y - I_z}{I_x}) - \frac{J_R}{I_x}x_4\Omega + \frac{l}{I_x}U_1,\\
\dot{x}_3 &= x_4,\\
\dot{x}_4 &= x_2x_6(\frac{I_z - I_x}{I_y}) + \frac{J_R}{I_y}x_2\Omega + \frac{l}{I_y}U_2,\\
\dot{x}_5 &= x_6,\\
\dot{x}_6 &= x_2x_4(\frac{I_x - I_y}{I_z}) + \frac{l}{I_z}U_3,
\end{aligned}$$

where $I_x, I_y,$ and I_z represents the body inertia, l denotes the lever, and J_R is the rotor inertia. With the control policy

$$\begin{aligned}
U_1 &= -\frac{I_x}{l}(x_1 - x_1^d) - k_1x_2,\\
U_2 &= -\frac{I_y}{l}(x_3 - x_3^d) - k_2x_4,\\
U_3 &= -I_z(x_5 - x_5^d) - k_3x_6,
\end{aligned}$$

and restricting $I_x = I_y$, the angular rotations subsystems is stabilized to the chosen equilibrium point $X_d = \{x_1^d, 0, x_3^d, 0, x_5^d, 0\}$. In empirical experiments, we set $X_d = \{0, 0, 0, 0, 0, 0\}$, state space $\mathcal{D} = \{(x_1, x_2, x_3, x_4, x_5, x_6) \in$

$\mathbb{R}^6 \mid |x_i| \leq 3, \forall i \in \{1, 2, .., 6\}\}$, $I_x = I_y = 2$, $I_z = 5$, $k_1 = 5$, $k_2 = 20$, $k_3 = 4$, $l = 1$, $J_R = 1$, and $\Omega = \sin(x_2)\cos(x_4)$, our framework successfully found Lyapunov function $V(x_1, x_2, x_3, x_4, x_5, x_6) = \sum_{i=1}^{6} x_i^2$. By examining the Lie derivative $L_f V = -2.5x_2^2 - 10x_4^2 - 0.8x_6^2$ and using the Invariance principle, we conclude that this subsystem is globally asymptotically stable. Under other parameter settings, our framework can also retrieve valid analytical Lyapunov functions for this dynamics.

G.5. N-bus Lossy Power System

Unlike N-bus lossless power systems in Appendix G.3 which has a well-known energy-based storage function served as a valid Lyapunov function for asymptotical stability guarantee, the N-bus lossy power system (Cui & Zhang, 2022) does not have a known analytical Lyapunov function to certify stability, though by passivity the system should be asymptotically stable at origin. Utilizing the proposed framework, we aim to discover a valid analytical Lyapunov function for an N-bus lossy power system to formally certify its asymptotic stability.

The θ_i and δ_i are the angle and frequency deviation of bus i, the dynamics of an N-bus lossy power system is represented by the swing equation, formulated as:

$$\begin{aligned} \dot{\theta}_i &= \omega_i, \\ m_i\dot{\omega}_i &= p_i - d_i\omega_i - u_i(\omega_i) - \sum_{j=1}^{N} B_{ij} \cdot \sin(\theta_i - \theta_j) - \sum_{j=1}^{N} G_{ij} \cdot \cos(\theta_i - \theta_j), \end{aligned}$$

where m_i is the generator inertia constant, d_i is the combined frequency response coefficient from synchronous generators and frequency-sensitive load, and p_i is the net power injection, for each bus $i = 1, \cdots, N$. The susceptance and conductance of the line (i, j) are $B_{ij} = B_{ji}$ and $G_{ij} = G_{ji}$, respectively. The value is 0 if the buses are not connected. In this work, we consider input u_i to be a static feedback controller where only its local frequency measurement ω_i is available. Like the lossless power system, since the frequency dynamics of the system depends only on the phase angle differences, we change the coordinates:

$$\delta_i = \theta_i - \frac{1}{N}\sum_{i=1}^{N} \theta_i$$

where δ_i can be understood as the center-of-inertia coordinates of each bus.

We test the proposed framework on a 2-bus lossy power system. In experiment, we set $p_i = 1$,$m_i = 2$, $d_i = 1$, $u_i(w_i) = w_i$, $B_{ij} = 1, G_{ij} = 1 \; \forall \; i \neq j$, $B_{ii} = 0, G_{ij} = 0$. By this setting, the equilibrium point for this system is at the origin, i.e. $\delta_i^* = \omega_i^* = 0$. The state space for the experiment is defined as: $\mathcal{D} = \{(\delta_1, \delta_2, \omega_1, \omega_2) \in \mathbb{R}^4 \mid |\delta_i| \leq 0.75$ and $|\omega_i| \leq 2$ for $i = 1, 2\}$. The proposed method found two valid Lyapunov function $V(\delta_1, \delta_2, \omega_1, \omega_2) = \omega_1^2 + \omega_2^2 + (\omega_2 - \sin(\delta_1) + \sin(\delta_2))^2$ and $V(\delta_1, \delta_2, \omega_1, \omega_2) = \omega_1^2 + \omega_2^2 + (-\omega_1 - \sin(\delta_1) + \sin(\delta_2))^2$. Both Lyapunov functions pass the formal verification by SMT solver in the state space $\mathcal{D} \backslash \mathcal{B}_\epsilon(0)$, where precision δ is set to be e^{-12} and $\epsilon = e^{-3}$ to avoid tolerable numerical error.

G.6. 9-D Synthetic Dynamics

Consider the synthetic dynamics adapted from Appendices F.4 & G.2 with linear interactions between two subsystems:

$$\begin{aligned} \dot{x}_1 &= -x_1 + 0.5x_2 - 0.1x_5^2, \\ \dot{x}_2 &= -0.5x_1 - x_2 + 0.1x_8, \\ \dot{x}_3 &= -x_3 + 0.5x_4 - 0.1x_1^2, \\ \dot{x}_4 &= -0.5x_3 - x_4, \\ \dot{x}_5 &= -x_5 + 0.5x_6, \\ \dot{x}_6 &= -0.5x_5 - x_6 + 0.1x_2^2, \\ \dot{x}_7 &= x_8, \\ \dot{x}_8 &= -\sin(x_7)\cos(x_7) - x_8 - \sin(x_9)\cos(x_9) - 0.1x_2, \\ \dot{x}_9 &= x_8 - x_9. \end{aligned}$$

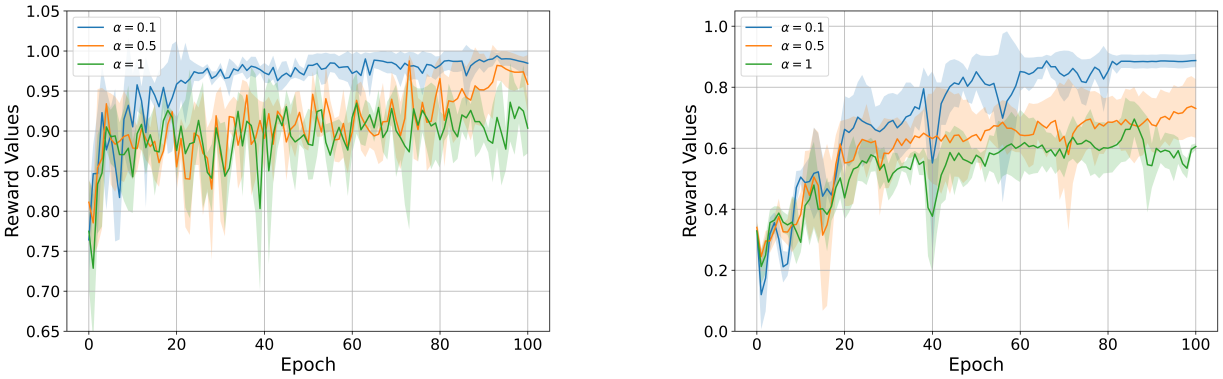

Figure 5. **Reward Trajectory Comparison for Ablation Study I**: Test proposed framework (without GP component) on 3-D Trig dynamics with three different choices of α for 100 epochs. The figure on the left visualizes the highest reward in the batch of samples obtained in each epoch, and the figure on the right visualizes the 90% quantile of reward distributions of the batch of samples obtained in each epoch.

To properly address the trigonometric terms in $\dot{x}_8$, the Lyapunov function for this dynamics can't be a simple form like $\sum_{i=1}^{n} x_i^2$ and should include some trigonometric terms. Setting the state space $\mathcal{D} = \{x \in \mathbb{R}^9 || x_i | \leq 1.5, \forall\, i = 1, \cdots, 9\}$, our method successfully identifies a valid Lyapunov function $V = \sum_{i=1}^{6} x_i^2 + \sin(x_7)^2 + x_8^2 - \cos(x_9) + 1$, which passes formal verification following settings in Section 5.

H. Ablation Studies

H.1. Ablation Study I - Risk-seeking Policy Gradient

To better understand the influence of α in policy gradient on the training performance of the symbolic transformer, we experiment with our framework without GP refinement on 3-D Trig dynamics under different choices of α (the risk-seeking quantile) over 100 epochs. Specifically, we choose $\alpha = 0.1, 0.5, 1$ and compare the transformer model's performance by the reward trajectories and success discovery rate. Figure 5 plots the reward trajectories—both the highest reward and the 90%-quantile—under the three different choices of α (the risk-seeking quantile). For $\alpha = 0.1$, the setting we used on tested dynamics in experiments, the highest reward and the 90% quantile converge steadily to 1 and 0.9, respectively, exhibiting lower variance and faster stabilization. For $\alpha = 0.5$, the median-based threshold increases steadily but shows greater instability. Finally, for $\alpha = 1$ vanilla policy gradient, the best reward reaches only 0.94, while the 90% quantile hovers around 0.6. Additionally, with $\alpha = 0.1$, the framework achieves the highest success recovery rate (66.67%) compared to 33.33% and 0% for $\alpha = 0.5$ and $\alpha = 1$. These findings correspond to the analysis of misaligned objective between the standard policy gradient and Lyapunov function construction in Subsection 4.3, confirm the importance of risk-seeking policy optimization, and highlight the effectiveness of setting $\alpha = 0.1$ in experiments.

H.2. Ablation Study II - Global-optimization-based Numerical Verification Process

This work proposes a global-optimization-based numerical verification for candidate verification and counterexamples' feedback within the training paradigm. To identify the effectiveness of the proposed verification algorithm, this ablation study compares the following verification methods on the proposed framework without the GP component on the 6-D polynomial dynamics (Appendix F.4):

- **Global optimization (SHGO):** Proposed framework with global-optimization-based numerical verification in Subsection 4.2.
- **Root finding:** Proposed framework with root-finding numerical verification introduced in Feng et al. (2024c).
- **SMT:** Proposed framework with formal verification *dreal* (Gao et al., 2013) SMT solver.
- **Ramdom sampling:** Proposed framework with random sampling verification.

Table 7. Number of epoch, training time, success rate, and formal verification time comparison between four different settings in Ablation study II. Runtime is the average training time for successful trials. The Succ. % is the success rate of finding a valid Lyapunov function out of 3 random seeds. Ver. Time is the average formal verification time of identified candidates on successful trials. The experiment terminates either when a valid Lyapunov function is found (passes formal verification), the training epoch exceeds 250, or a timeout (4 hours limit).

Stats.	Global optimization	Root finding	SMT	Random sampling
Epoch	168	-	-	**160**
Runtime	8399s	-	-	**5684s**
Succ. %	**100**	0	0	**100**
Ver. Time	**1.67s**	-	-	15.93s

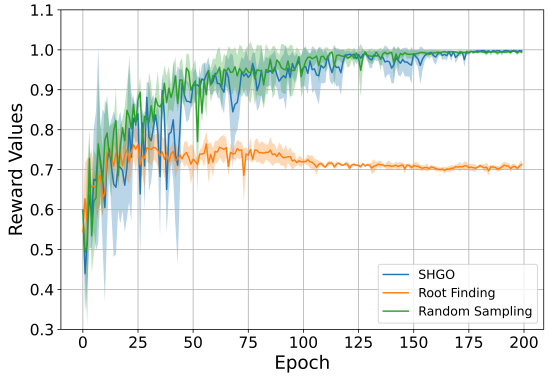

Figure 6. **Best Reward Trajectory for Ablation Study II**: Visualize the best reward trajectory of our framework with SHGO, root finding, and random sampling schemes in Ablation Study II on 6-D polynomial dynamics. The SMT-verification-based trajectory is not included as it timeouts at the beginning training stage.

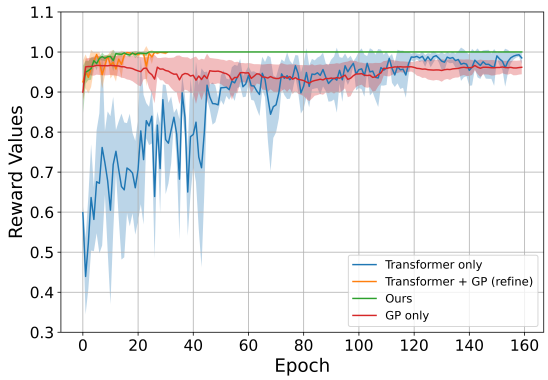

Figure 7. **Best Reward Trajectory for Ablation Study III**: Visualize the best reward trajectory of the proposed framework under four different settings in Ablation Study III on 6-D polynomial dynamics.

Figure 6 compares the highest reward trajectory in each epoch for all four settings, and Table 7 summarizes the training and formal verification statistics. SMT is not included in Figure 6 as the training stuck on the SMT solver for hours in the middle of the training stage. The proposed SHGO-based counterexample feedback makes the training reward an adversarial reward, as it can identify the most critical violations. As a 'challenger', the global optimization method finds the adversarial counterexamples, not just mild violations. Consequently, this leads to a faster refinement of the policy to mitigate the critical violations and results in stronger guarantees on the final Lyapunov candidates. However, it risks training stability and may over-focus on 'hard' violations. In this case, it identifies a valid Lyapunov function $V_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$ which is well-structured and easy to verify by formal verification tools. Note that, unlike in the experiment section, the GP component is not used here, resulting in a less structured local Lyapunov function with a few squared interaction terms $(x_i + x_j)^2$ under one random seed. This increases verification time compared to Table 1, yet remains significantly faster than verifying the final result from the random sampling scheme.

Among the four settings, random sampling achieves the fastest convergence for this polynomial system, as it yields smoother rewards, covers the whole state space uniformly, and promotes training stability. However, because random sampling cannot actively search for violations—and its search space grows exponentially with dimensionality—it becomes less effective for larger or more complex systems (e.g., power systems). Even with a reward near 1, it may fail to find a valid Lyapunov function as the high reward only means it fails to locate counterexamples instead of nonexistence; or if it does, it can converge to a local minimum with poor generalization or verification properties. For instance, the discovered Lyapunov function $V_2(x) = x_5^2 - \cos(x_2)\cos(x_3) - \cos(x_4) - \cos(x_6) - \cos(x_1 + x_4) + 4$ has a complex shape that formal verification tools struggle to handle, and it offers limited insight beyond the training region $\mathcal{D}$. By contrast, $V_1(x)$ remains valid over the

Table 8. Number of epochs, training time, and success rate comparison between four settings of the proposed framework. Runtime is the average training time for successful trials. Succ. % is the successful rate of finding a valid Lyapunov function out of 3 random seeds.

Training Stats.	Transformer only	Transformer + GP	Ours	GP only
Epoch	168	20	**18**	-
Runtime	8399s	2731s	**2467s**	-
Succ. %	**100**	**100**	**100**	0

much larger domain $\mathcal{D}' = \{(x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{R}^6 \mid \sum_{i=1}^{6} x_i^2 \leq 500\}$.

Root-finding usually identifies counterexamples near function roots, producing mostly mild violations. Although an ideal root-finding algorithm might be efficient, off-the-shelf methods become inaccurate for dimensions above five. As a result, they fail to provide precise violation signals or strong optimization guidance to the transformer, causing the best reward to stall at 0.7. Similarly, *dreal* SMT often locates only mild violations rather than critical ones, limiting its impact on optimization. While these counterexamples help the transformer reduce violations, the refined transformer increasingly acts as an adversary to the SMT solver, prolonging verification until timeout. In neural network + formal verification settings, this effect is even exacerbated by the model's overparameterization. Consequently, we rely on global optimization instead of formal verification in the training loop.

H.3. Ablation Study III - Generic Programming and Expert Guidance Loss

In the proposed framework, the Genetic Programming (GP) component refines sampled candidates $\tilde{V}_\phi$ from the symbolic transformer and optimizes the transformer parameters ϕ using the expert guidance loss in Equation (7). To assess the impact of the GP component on the training paradigm, we evaluate the following four settings of the proposed framework on the 6-D polynomial dynamics (Appendix F.4):

- **Transformer only**: A pure symbolic transformer model without GP component.
- **Transformer + GP**: A symbolic transformer model with a GP component for candidate refinement but without expert-guided learning.
- **Transformer + GP + expert-guided learning (ours)**: A symbolic transformer model with a GP component for both candidate refinement and expert-guided learning for transformer optimization.
- **GP only**: GP component only. The initial population of the first epoch comes from a randomly initialized symbolic transformer, and the initial population of other epochs comes from the latest refined expressions.

Figure 7 shows the highest reward in each epoch for the four settings, and Table 8 compares their training statistics. The transformer-only approach achieves a 100% success rate across three random seeds, proving its capability to generate Lyapunov functions on high-dimensional systems. However, without the GP component, it requires significantly more training time and epochs to identify a valid Lyapunov function because it must independently explore all necessary characteristics of a valid solution. Conversely, the GP-only approach fails to find any valid Lyapunov functions when starting from a random population, as it lacks the transformer's deeper understanding of system dynamics and relies on less capable evolutionary operations for exploration.

Figure 8 explicitly compares the reward trajectories with and without expert-guided supervised learning, where ours (with expert-guided learning) achieves the fastest convergence. Comparing the reward trajectory, ours achieves faster and more stable training. Overall, these visualizations and comparisons emphasize the benefits of GP refinement and the expert-guided learning on 'elite set' $\tilde{\mathcal{V}}_{gp}$ to accelerate transformer parameters' update and enable faster Lyapunov function discovery.

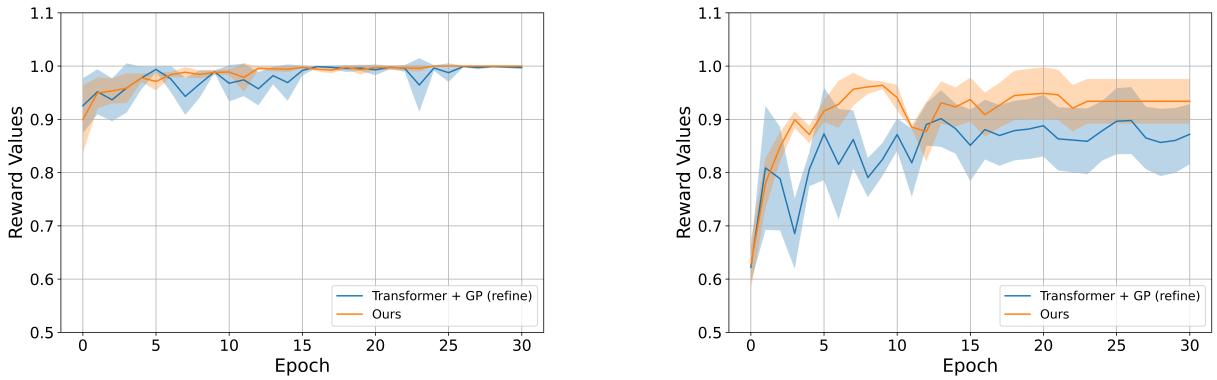

Figure 8. The figure on the left visualizes the overall best reward (best candidate among $\tilde{\mathcal{V}}_\phi \cup \tilde{\mathcal{V}}_{gp}$) trajectory. The figure on the left visualizes the best reward trajectory for transformer candidates (best candidate among $\tilde{\mathcal{V}}_\phi$ only, excludes GP-generated solutions). These two plots compare the performance of transformer + GP refinement and our method in Ablation Study III.