In: 2025 33rd Mediterranean Conference on Control and Automation (MED), 10-13 June 2025, Tangier, Morocco
DOI:10.1109/med64031.2025.11073312

# Comparative Analysis of Sum-of-Squares Optimization and Neural Network Lyapunov functions for Region of Attraction Estimation

Sorachat Chuenwongaroon[1], Argyrios Zolotas[2] and Dmitry Ignatyev[3]

***Abstract*— Region of Attraction (ROA) estimation with accuracy is crucial for effective control design in nonlinear dynamical systems. Sum-of-squares (SOS) optimization refines Lyapunov function representations and expands ROA estimation for polynomial dynamical systems. However, traditional SOS methods tend to be overly conservative. In contrast, deep learning has emerged as a powerful tool in robotics, enabling data-driven ROA estimation. While deep learning offers strong empirical performance, its lack of stability guarantees remains challenging in safety-critical applications. This paper compares two ROA estimation methods: Sum-of-squares (SOS) optimization and Neural Network Lyapunov Functions (NLFs). We examine their effectiveness in approximating Lyapunov functions for stability assessment, highlighting their strengths, limitations, and practical relevance. Using the Van der Pol oscillator as a benchmark, we conduct detailed simulations to evaluate each method's performance. We provide a comprehensive comparison, considering computational efficiency, accuracy, and scalability, offering insights into their applicability in real-world aerospace systems.**

## I. Introduction

Safety-critical applications like aerospace, robotics, and power systems require stability analysis. Establishing a system's region of attraction (ROA) around a stable equilibrium can ensure the system returns to equilibrium even after disturbances. The Lyapunov function is a crucial core theory invented more than a hundred years ago [1] that certifies stability and estimates the region of attraction (ROA). There have been many attempts to develop methods to create Lyapunov functions. First, starting with Lyapunov's direct linear system method works well. However, when it comes to nonlinear systems, a more advanced method is required to create Lyapunov functions.

Sum-of-squares is the method that gets much attention and can approximate Lyapunov functions as polynomials. This method for constructing Lyapunov functions is a relatively modern development in control theory and optimization. It emerged in the late 1990s and early 2000s [2] [3] primarily due to advances in computational tools and the application of semidefinite programming (SDP) to solve polynomial optimization problems. The Sum-of-Squares (SOS) optimization approach [4]–[6] leverages polynomial Lyapunov function candidates and semidefinite programming (SDP) to automate stability proofs. Despite its elegance and strong theoretical guarantees, SOS can become computationally intractable for higher-dimensional systems or when high-degree polynomials are required. Moreover, the resulting ROA estimates might be overly conservative due to polynomial function restrictions. Extending the region of attraction can be achieved by integrating a tailored shape function within the estimated stability region. However, systematically selecting such functions beyond conventional choices like norm-based [7] or quadratic formulations [8] remains an open challenge.

The AI revolution has propelled machine learning-driven innovations in recent years. New machine learning-based approaches for constructing Lyapunov functions use Neural Network Lyapunov functions (NLFs) [9]–[15]. This technique uses data-driven training to approximate more general classes of Lyapunov functions. Such approaches may enlarge the estimated ROA and reduce modeling biases. NLFs have emerged to overcome some limitations of polynomial-based methods. One can capture more complex or non-polynomial behaviors by training a network on sampled states to enforce approximate Lyapunov conditions. However, a critical challenge is post-training verification. Often, the verification step via solvers (e.g., Satisfiability Modulo Theories (SMT), Mixed-Integer Programming (MIP)) exceeds the training time, casting doubt on the method's applicability in real-time systems without further optimization or specialized tools. Neural networks approximate Lyapunov functions with trainable parameters. The training phase enforces Lyapunov conditions on oversampled states, often via penalty terms in the loss function. After training, a verification phase uses solvers (SMT, MIP, or interval-based) to ensure these conditions hold for all in the domain. This extra step can be computationally significant, which is a key drawback compared to SOS. For a recent development, LyZNet [15] is a lightweight Python tool that exemplifies recent advancements in efficiently learning and verifying Lyapunov functions and regions of attraction (ROA). It streamlines the verification process by utilizing physics-informed neural networks to solve the associated partial differential equations (PDEs) and enhances computational efficiency. Creating Lyapunov functions that deal with nonlinearities and uncertainties is a big leap in this area. But using neural networks introduces new challenges. For starters, checking if the learned Lyapunov function satisfies all stability requirements can take much time, sometimes even longer than the initial training. Next, guaranteeing safety for safety-critical applications remains a

[1]Sorachat Chuenwongaroon is with the School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, United Kingdom. s.chuenwongaroon@cranfield.ac.uk

[2]Argyrios Zolotas is a professor in the School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, United Kingdom. a.zolotas@cranfield.ac.uk

[2]Dmitry Ignatyev is a senior lecturer in the School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, United Kingdom. d.ignatyev@cranfield.ac.uk

Published by IEEE. This is the Author Accepted Manuscript issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.1109/med64031.2025.11073312. Please refer to any applicable publisher terms of use.

significant obstacle without a robust certification process.

This paper presents a comparison between SOS optimization and Neural Network Lyapunov functions. Specifically, we address the following questions:

1) Can Neural Network Lyapunov functions outperform classical SOS regarding ROA estimation?
2) What are the computational overheads associated with each method, including the cost of verification for Neural Network approaches?
3) Can Neural Network-based methods be safely deployed in real-world, safety-critical settings, or is SOS currently more reliable despite its conservatism?

The rest of this paper is organized as follows. Section II covers background and preliminary concepts, including an overview of ROA and classical Lyapunov stability. Section III & IV proposes an analytical framework detailing Sum-of-Squares (SOS) optimization and Neural Network Lyapunov approaches, respectively. Section V provides a comparative analysis illustrated by the Van der Pol oscillator. We conclude in Section VI with key takeaways and directions for future work.

## II. Background and Preliminaries

### A. Region of Attraction (ROA)

Region of Attraction (ROA) represents the initial conditions from which the system trajectories converge to a stable equilibrium [16]. For a nonlinear system:

$$\dot{x} = f(x), \quad x(0) = x_0, \tag{1}$$

ROA is mathematically defined as

$$R_A = \left\{ x \in D : \phi(t;x) \text{ and } \lim_{t\to\infty} \|\phi(t;x)\| = 0 \right\}, \tag{2}$$

where $D \subset \mathbb{R}^n$ is a domain containing the equilibrium point, $f : D \to \mathbb{R}^n$ is locally Lipschitz, and $\phi(t,x)$ is the solution to the system.

### B. Lyapunov Local Stability

Lyapunov functions provide an indispensable tool for stability analysis and ROA estimates (sometimes called their Lie derivative):

$$V(x) > 0, \forall x \neq 0 \text{ and } V(0) = 0, \tag{3}$$

$$\dot{V}(x) := \frac{dV}{dx} \cdot f(x) < 0, \forall x \neq 0, \tag{4}$$

where any such function and any positive constant $c > 0$, the sublevel set $x \in R^n : V(x) \leq c$ is a region of attraction in 2, provided that the set is contained in $D$.

Local stability analysis using quadratic Lyapunov functions is fairly standard [17]. Suppose that $x = 0$ is an exponentially stable equilibrium point for (1); i.e, $f(0) = 0$ and the Jacobian Matrix $A = Df(0) = \frac{df}{dx}|_{x=0}$ is a Hurwitz matrix i.e., all eigenvalues of $A$ have negative real parts which can rewritten as

$$\dot{x} = Ax + g(x), \tag{5}$$

where $g(x) = f(x) - Ax$ satisfies $\lim_{x\to 0} \frac{\|g(x)\|}{\|x\|} = 0$. Lyapunov equations are:

$$V(x) = x^T P x, \quad P > 0, \tag{6}$$

$$PA + A^T P = -Q \tag{7}$$

where $P$ is a unique solution to the Lyapunov equation, and $Q \in R^{n\times n}$ is any symmetric positive definite real matrix.

$$\begin{aligned}
\dot{V}(x) &= x^T(PA + A^T P)x + 2x^T P g(x), \\
&= -x^T Q x + 2x^T P g(x), \\
&\leq -\lambda \|x\|^2 + 2x^T P g(x), \\
&= -\varepsilon \|x\|^2 + (2x^T P g(x) - r\|x\|^2),
\end{aligned}$$

where we set $r = \lambda - \varepsilon > 0$ for some sufficiently small $\varepsilon > 0$ and $\lambda$ is the minimum eigenvalue of $Q$. The goal of this local stability analysis is to determine the largest number $c_1$ such that:

$$x \in P \Rightarrow 2x^T P g(z) \leq r\|x\|^2, \tag{8}$$

where $P$ is a sublevel set of quadratic Lyapunov functions.

### C. Case Study: Van der Pol Oscillator

A widely cited nonlinear benchmark is the Van der Pol oscillator,

$$\begin{aligned}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= -x_1 + \mu(1 - x_1^2)x_2,
\end{aligned} \tag{9}$$

where $\mu > 0$ controls the nonlinearity. It features a stable limit cycle for certain parameter ranges and poses challenges for polynomial-based methods due to its nonlinear term $(1 - x_1^2)$. Neural networks can, in principle, approximate more intricate Lyapunov functions for this system, but verifying them involves lengthy computations.

## III. Sum-of-Squares (SOS) Optimization

Sum-of-Squares (SOS) optimization is a convex optimization framework that enables stability certification and Region of Attraction (ROA) estimation for polynomial dynamical systems. It extends Lyapunov stability analysis by representing the Lyapunov function $V(x)$ as an SOS polynomial and optimizing its level sets to estimate a larger ROA [18].

*1) Problem Setup:* Consider a nonlinear polynomial dynamical system in (1).

*2) Candidate Lyapunov Function:* Define a polynomial candidate Lyapunov function:

$$V(x) = \sum_{i=1}^{t} q_i^2(x), \tag{10}$$

where $q_i(x)$ are polynomials in $\mathbb{R}_n$. The set of SOS polynomials in $n$ variables is defined as:

$$\Sigma_n := \{ p \in \mathbb{R}_n \mid p = \sum_{i=1}^{t} f_i^2, \quad f_i \in \mathbb{R}_n \}. \tag{11}$$

If $p(x) \in \Sigma_n$, then $p(x) \geq 0$ for all $x \in \mathbb{R}^n$, ensuring positivity.

*3) Shape Functions and Union Theorem:* A shape function (SF) is a positive definite function used in SOS optimization to improve ROA estimation. It guides the expansion of the Lyapunov function $V(x)$ by increasing the enclosed area. If its center is at the origin, it is called an ordinary SF; otherwise, it is a shifted SF. The optimization problem maximizes the ROA using SFs:

$$\max \beta \quad \text{subject to} \quad \{x : p(x) \leq \beta\} \subseteq \{x : V(x) \leq \gamma\}. \quad (12)$$

Where $p(x)$ is the shape function, $\beta$ is the scaling factor that determines the expansion of $p(x)$, $\gamma$ is the level set of $V(x)$ defining the estimated ROA.

To refine the ROA estimate using multiple SFs, the Union Theorem is applied:

$$-(V-\gamma)+(p_i-\beta_i)s_i(x)(\in \Sigma_n, \quad i=1,\ldots,n. \quad (13)$$

Where $s_i(x)$ are SOS multiplier polynomials ensuring the inclusion constraints hold.

Thus, the union of the regions defined by all shape functions is enclosed by $V(x) \leq \gamma$.

*4) V-S Iteration Algorithm [18]:* The SOS optimization iteratively enlarges the ROA by alternating between 4.1), 4.2), and 4.3) steps:

4.1) $\gamma$-Step: Solve for the SOS multiplier $s_0(x)$ and the optimal bound $\gamma^*$ for a given $V(x)$ and a fixed stability margin polynomial $l_2(x)(x)$:

$$\gamma^* = \max_{\mathbf{s_0} \in \Sigma_n} \gamma \quad \text{s.t.} \; -\left[\frac{\partial V}{\partial x} f + l_2(x)\right] + (V-\gamma)s_0 \in \Sigma_n. \quad (14)$$

where $l_2(x)$ is a fixed positive definite polynomial that ensures strict negativity of $\dot{V}(x)$.

4.2) $\beta$-Step: Solve for the SOS multipliers $s_1, s_2, ..., s_n$ and the optimal shape function bounds $\beta_1^*, \beta_2^*, ..., \beta_n^*$ using the obtained $\gamma^*$:

$$\left[\beta_1^*, \beta_2^*, ..., \beta_n^*\right] = \max_{\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_n} \in \Sigma_\mathbf{n}} \beta_1, \beta_2, ..., \beta_n \quad \text{s.t.} : \\ -(V-\gamma^*)+(p_i-\beta_i)s_i(x) \in \Sigma_n, \quad i=1,...,n. \quad (15)$$

where $s_i(x)$ are SOS polynomials ensuring that each shape function region is contained in the ROA estimate.

4.3) $V$-Step: Solve for a new Lyapunov function $V(x)$ that satisfies:

$$V - l_1 \in \Sigma_n, \quad (16)$$

$$-\left[\frac{\partial V}{\partial x} f + l_2(x)\right] + (V-\gamma^*)s_0 \in \Sigma_n, \quad (17)$$

$$-(V-\gamma^*)+(p_i-\beta_i^*)s_i(x) \in \Sigma_n, \quad i=1,...,n. \quad (18)$$

where $l_1(x)$ is a positive definite polynomial ensuring $V(x)$ is strictly positive everywhere except at $x=0$.

4.4) Scaling $V$: Normalize $V(x)$ after each iteration:

$$V(x) \leftarrow \frac{V(x)}{\gamma^*}. \quad (19)$$

This step ensures numerical stability and prevents $V(x)$ from growing unbounded.

4.5) Repeat steps 1-4 until the final feasible Lyapunov function $V^*(x)$ is obtained, defining the estimated ROA:

$$R_A = \Omega^* := \{x \in \mathbb{R}^n : V^*(x) \leq 1\}. \quad (20)$$

## IV. Neural Network Lyapunov functions (NLFs)

This section presents two algorithms for Neural Network methods: one derived from [13] and the other from [15].

### A. Neural Network Lyapunov Functions with Stability Guarantees (NLFG)

This approach employs two neural networks: one to approximate the unknown system dynamics and another to learn a Lyapunov function.

*1) Learning the Unknown Dynamics:* Parameters are given that

- $\|(x)-(y)\| \leq \delta \quad \forall x$ and $y$
- $\left\|\frac{\partial V}{\partial x}\right\| < M$ Note: $V$ is the output of a neural network
- $\alpha$: maximum of the 2-norm loss of all known data samples

Then, the generalization error of approximated dynamics $\phi$ is bounded by

$$\begin{aligned} \|f(x)-\phi(x)\| &\leq \|f(x)-f(y)\| + \|f(y)-\phi(y)\| \\ &\quad + \|f(x)-f(y)\| + \|\phi(y)-\phi(x)\| \\ &\leq K_f\delta + \alpha + K_\phi\delta \\ &< \frac{\beta}{M} \end{aligned} \quad (21)$$

where $K_f, K_\phi$ are the Lipschitz constants of $f$ and $\phi$ respectively.

The unknown system dynamics are approximated by a neural network $\phi(x)$ with a mean square loss:

$$L_{\text{dyn}}(\phi) = \sum_i ||f(x_i)-\phi(x_i)||^2. \quad (22)$$

*2) Lyapunov Function Learning:* After learning the unknown dynamics using $\phi(x)$, the second neural network is designed to learn a neural Lyapunov function $V(x)$. The training utilizes data generated from the learned system dynamics, which introduces approximation errors. To account for these errors, the standard Lyapunov conditions in 3 and 4 are reformulated as:

$$V(0)=0, \quad \forall x \in D \setminus \{0\}, V(x)>0, \quad \nabla_\phi V(x) < -\beta.$$

Here, $\beta$ is a positive real number that can be determined through an error-bound analysis.

*3) Error-Bound Analysis:* Given an unsampled state $(x)$ and its nearest sampled state $(y)$, let $\delta > 0$ be the bound such that:

$$\|(x)-(y)\| \leq \delta. \quad (23)$$

Define $M > 0$ as an upper bound on the gradient norm of $V(x)$:

$$\left\|\frac{\partial V}{\partial x}\right\| < M, \quad \forall x \in D. \quad (24)$$

Let $\alpha$ be the maximum 2-norm loss across all known samples in the training or test dataset. The bound on the generalization error can then be expressed as:

$$\|f(x)-\phi(x)\| \leq K_f\delta+\alpha+K_\phi\delta < \frac{\beta}{M}, \tag{25}$$

where $K_f$ and $K_\phi$ are Lipschitz constants for $f$ and $\phi$, respectively.

From this, we derive the inequality:

$$\begin{aligned}\nabla_f V(x)-\nabla_\phi V(x) &\leq \left\|\frac{\partial V}{\partial x}\right\|\cdot\|f(x)-\phi(x)\| \\ &< M\cdot\frac{\beta}{M}=\beta. \end{aligned}\tag{26}$$

Thus, we conclude:

$$\nabla_f V(x) < \nabla_\phi V(x)+\beta < -\beta+\beta = 0, \quad \forall x\in D\setminus\{0\}.$$

The Lyapunov function is trained using:

$$\begin{aligned} L_1(\theta) &= \sum_i \max(0,-V_\theta(x_i)),\\ L_2(\theta) &= \sum_i \max(0,\dot{V}_\theta(x_i)),\\ L_3(\theta) &= ||V_\theta(x)||^2,\\ L_4(\theta) &= \left\|\frac{\partial V}{\partial x}\right\|. \end{aligned}$$

Total loss function:

$$L(\theta)=\lambda_1 L_1+\lambda_2 L_2(x)+\lambda_3 L_3+\lambda_4 L_4. \tag{27}$$

*4) Falsification Process:* To verify the Lyapunov function's validity, we encode the Lyapunov conditions as falsification constraints using an SMT solver [19] [13]:

$$\Phi_\varepsilon(x) := \left(\sum_{i=1}^n x_i^2 \geq \varepsilon\right)\wedge(V(x)\leq 0\vee\nabla_\phi V(x)\geq-\beta), \tag{28}$$

Where $\varepsilon$ is a numerical error parameter used to control sensitivity around the origin; if the SMT solver returns 'UNSAT,' it confirms that the Lyapunov conditions hold. Otherwise, a counterexample is generated and added to the training set.

*5) Theoretical Guarantees:* We provide theoretical guarantees on the stability of the unknown system under the learned controller. Using bounds on the Lipschitz constants and neural network error estimates, we show that:

$$\nabla_f V(x) < 0, \quad \forall x\in D\setminus\{0\},$$

which ensures that the origin remains in an asymptotically stable equilibrium.

### *B. Physics-Informed Neural Network Lyapunov Functions (PINNs)*

This algorithm integrates Physics-Informed Neural Network Lyapunov Functions (PINNs) and Zubov's equation to improve the ROA estimation [15] [20].

*1) Assumptions:* The framework verifies four sets

- $P_1$: A local Region of Attraction (ROA) verified by linearization where $V_P(x)\leq c_{1_p}$.
- $P_2$: The set of initial conditions that reach $P_1$ using a quadratic Lyapunov function where $V_P(x)\leq c_{2_p}$.
- $V_1$: A target invariant set contained in $P_2$, verified using a neural Lyapunov function where $V_N(x)\leq c_{1_V}$.
- $V_2$: An invariant set from which solutions reach $V_1$, defining the verified ROA where $V_N(x)\leq c_{2_V}$.

These sets satisfy the following inclusions:

$$P_1\subseteq P_2, \quad V_1\subseteq P_2, \quad \text{and} \quad V_1\subseteq V_2.$$

*2) Local Stability Analysis with Quadratic Lyapunov Functions:* A quadratic Lyapunov function is obtained by solving the Lyapunov equations in Section II-B.

*3) Learning via Zubov's Equation:* It uses Physics-Informed Neural Networks (PINNs) to solve Zubov's Partial Differential Equation (PDE) for the Lyapunov function.

$$\nabla W(x)\cdot f(x) = -\Psi(x)(1-W(x)) \tag{29}$$

where $W(x)$ is a positive definite function to be solved, and

$$\Psi(x) = -\alpha(1+W(x))\omega(x) \tag{30}$$

It is also a positive definite function that can be chosen, with $\alpha>0$ being a parameter. The loss function consists of:

$$L_{\text{residual}} = \frac{1}{N}\sum_{i=1}^N\left(\nabla_x W_N(x_i;\theta)f(x_i)+\Psi(x_i)(1-W_N(x_i;\theta))\right)^2$$

where $x_i$ are collocation points sampled from the system state and $\theta$ represents the trainable parameters of the neural network.

$$L_{\text{boundary}} = \sum_{i\in\beta}\left(W_N(x_{\beta_i})-W_{\text{target}}(x_{\beta_i})\right)^2$$

where $x_\beta$ are boundary points of the domain, and $W_{\text{target}}(x)$ defines the expected values: $W_N(0)=0$ (at equilibrium) and $W_N(x)=1$ (outside the domain).

$$L_{\text{data}} = \frac{1}{N_d}\sum_{i=1}^{N_d}\left(W_N(y_i;\theta)-\hat{W}(y_i)\right)^2$$

$y_i$ are points obtained from simulated system trajectories, $\hat{W}(y_i)$ is the computed Lyapunov function value using numerical integration, and $N_d$ is the number of simulation training points.

The total loss function minimizes the PDE residuals:

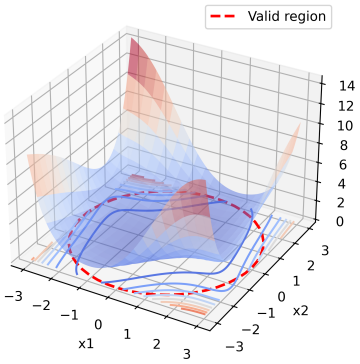
$$L(\theta) = L_{\text{residual}}+L_{\text{boundary}}+L_{\text{data}} \tag{32}$$

An SMT solver (dReal) is used to verify the Lyapunov conditions:

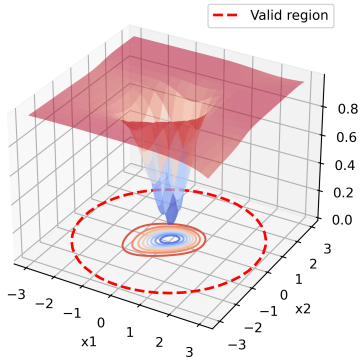
$$(c_1\leq W_N\leq c_2)\wedge(x\in X)\Rightarrow\dot{W}_N(x)\leq-\varepsilon \tag{33}$$

$$(W_N\leq c_1)\wedge(x\in X)\Rightarrow V_P(x)\leq c, \tag{34}$$

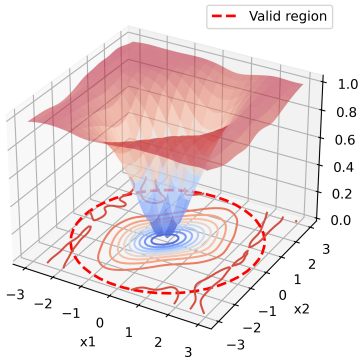
where $\varepsilon>0$, $c_2>c_1>0$. If counterexamples are found, the Region of Attraction (ROA) size is reduced iteratively, ensuring formal verification of the learned ROA.




(a) Polynomial Lyapunov function (SOS) with a $6^{th}$ degree polynomial.




(b) The Neural Network Lyapunov function with stability guarantees (NLFG) features a single hidden layer with 6 neurons and utilizes the Tanh activation function.




(c) The Physics-Informed Neural Network Lyapunov function (PINN) employs a two-layer neural network with 30 neurons per layer and utilizes the Tanh activation function

Fig. 1: Experimental Results for the Van der Pol system ($\mu = 1$).

TABLE I: Comparison of Sum-of-Squares (SOS) Optimization and Neural Network Lyapunov functions (NLFs)

| Criteria | SOS Optimization | NLFs |
|---|---|---|
| Cost for training | Medium (SDP solving) | High |
| Conservatism | Conservative | No conservative |
| Scalability | Limited | Scalable |
| Guarantees | Direct certification | Requires verification |
| Practicality | Mature tools available | Parameter tuning |

## V. Comparative Analysis

In this section, we present the experimental results in the form of Lyapunov functions for both the SOS and NLF methods, as illustrated in Figure 1, and perform a comparative analysis based on the Region of Attraction (ROA), shown in Figure 2. We achieve results as in Table II. All simulations were conducted on a system with a 13th Gen Intel® Core™ i7-13700H processor (2.40 GHz) and an NVIDIA RTX A2000 GPU.

### A. Comparative Criteria

We define five key metrics to compare Sum-of-Squares (SOS) and Neural Network Lyapunov Function (NLF) methods:

1) Computational Complexity: Compares SOS (SDP solving) with NLF (data generation, training, and verification).
2) ROA Conservatism: Evaluates the certified ROA size while ensuring stability.
3) Scalability: Assesses performance as system complexity increases.
4) Safety Guarantees: Compares SOS's direct certification with NLF's separate verification.
5) Implementation Practicality: Examines tool availability and parameter tuning ease.

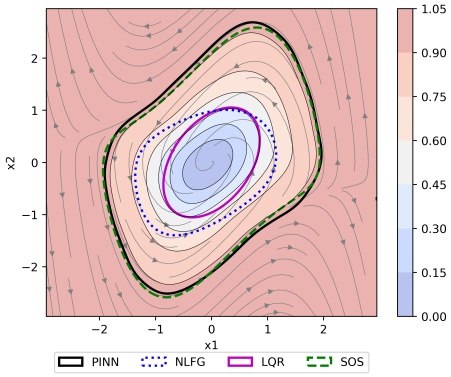



Fig. 2: Comparison of the Regions of Attraction (ROA) for Quadratic (LQR), Sum-of-Squares (SOS), Neural Lyapunov Function with stability guarantees (NLFG), and Physics-informed Neural Lyapunov Function (PINN) methods

TABLE II: Parameters and verification results for a Van der Pol Oscillator ($\mu = 1$ & $c_{max} = 0.94025$)

| Method | Comp. Time | ROA Corverage |
|---|---|---|
| Quadratic | 32 secs | 19.42% |
| SOS | 518 secs | 94.16% |
| NLFG | 2274 secs | 36.76% |
| PINN | 780 secs | 93.58% |

### B. Discussion

Applying these criteria from Section V-A to the Van der Pol oscillator case study (Section II-C), we can summarize the comparative analysis in Table I.

*1) Neural Network vs. SOS in ROA Estimation:* As discussed in Section 1, our findings indicate that Neural Network Lyapunov Functions (NLFs) do not yet outperform SOS in estimating the ROA. While Neural Network offers greater flexibility in function approximation, their ROA estimate from NLFG is generally smaller than those obtained through SOS optimization with multiple shape functions, as

shown in Figure 2. Moreover, SOS provides robust formal certification, whereas the NLFs require additional verification procedures. From Figure 2 and Table II, it is evident that PINN and SOS estimate the ROA more accurately, closely matching the actual ROA. In contrast, NLFG estimates a smaller ROA but provides stability guarantees by determining Lipschitz constants, as discussed in Section IV-A.5. SOS and NLFs (NLFG and PINN) also estimate a larger ROA than quadratic Lyapunov functions (LQR). Thus, these methods effectively optimize ROA estimation, leveraging advanced optimization techniques to improve performance.

*2) Computational Overheads:* Regarding computational efficiency (Section 2), NLF methods require significantly higher computational resources due to data generation, training, and verification. In particular, the verification stage necessitates counterexample searches using SMT (Satisfiability Modulo Theories) or MIP (Mixed-Integer Programming) solvers, as employed in this study. In contrast, SOS relies on convex optimization, which, while computationally intensive, does not require separate verification.

*3) Safety Considerations:* As highlighted in Section 3, at this point, NLFs are not considered safe for deployment in real-world, safety-critical applications. While the NLFs have advantages, such as handling high dimensionality, nonlinearity, and uncertainty, which SOS cannot address as effectively, there is a significant drawback. A primary concern is the lack of safety guarantees and solid certification for NLFs. Verification processes for Neural Networks still need further development since finding counterexamples may require extensive test data, and undiscovered counterexamples within the NLFs remain possible.

## VI. Conclusion

In conclusion, this paper compares the strengths and weaknesses of sum-of-squares (SOS) methods and Neural Network Lyapunov functions (NLFs) for ensuring stability in complex nonlinear systems. SOS methods provide stability guarantees through polynomial constraints [18], but they often restrict the Region of Attraction (ROA) estimates. In contrast, NLFs offer greater flexibility in high-dimensional spaces and can adapt to complex systems; however, they face significant challenges in verifying stability.

There are still substantial challenges to address, particularly the need for robust verification methods to support the validation of learned Neural Lyapunov functions. Additionally, Physics-Informed Neural Networks (PINNs) [20] depend on accurate mathematical models, which can limit their applicability in uncertain or poorly understood systems.

Future research should prioritize the development of strong verification methods for NLFs, as ensuring their reliability in real-world applications is critical. Furthermore, combining the strengths of Sum-of-Squares (SOS) Optimization and Neural Network approaches warrants further investigation. Finally, ensuring safety should also be prioritized, using barrier functions [20].


## Acknowledgment

This research was partially supported by the Ministry of Higher Education, Science, Research, and Innovation of Thailand through a PhD scholarship.


## References


[1] A. M. Lyapunov, *The General Problem of the Stability of Motion*. Kharkov: Kharkov Mathematical Society, 1892.

[2] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, USA, 2000, phD Thesis.

[3] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796–817, 2001.

[4] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming, Series B*, vol. 96, pp. 293–320, 5 2003.

[5] W. Tan, "Nonlinear control analysis and synthesis using sum-of-squares programming," Ph.D. dissertation, University of California, Berkeley, 2006.

[6] A. Papachristodoulou' and S. Prajna', "On the construction of lyapunov functions using the sum of squares decomposition'," 2002.

[7] A. Packard, P. Seiler, and G. Balas, "Sos methods for fixed-order $\mathscr{H}_\infty$ control," in *Proceedings of the American Control Conference (ACC)*. Baltimore, MD, USA: IEEE, 2010, pp. 85–90.

[8] H. Khodadadi, A. Karimpour, and N. Pariz, "A new approach to estimate the region of attraction for polynomial nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1294–1299, 2014.

[9] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 255–260, 3 2020. [Online]. Available: http://arxiv.org/abs/2003.08910

[10] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," 2021. [Online]. Available: https://github.com/StanfordASL/neural-network-lyapunov.

[11] H. Dai and F. Permenter, "Convex synthesis and verification of control-lyapunov and barrier functions with input constraints," *arXiv preprint arXiv:2210.00629*, 10 2022. [Online]. Available: http://arxiv.org/abs/2210.00629

[12] R. Tedrake, *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. MIT OpenCourseWare, 2024, pDF version of the course notes available at http://underactuated.csail.mit.edu/.

[13] R. Zhou, T. Quartz, H. D. Sterck, and J. Liu, "Neural lyapunov control of unknown nonlinear systems with stability guarantees," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 12, p. 121003, 2022. [Online]. Available: https://doi.org/10.1115/1.4055444

[14] V. Zinage and E. Bakolas, "Neural koopman lyapunov control," *Neurocomputing*, vol. 527, pp. 174–183, 2023. [Online]. Available: https://doi.org/10.1016/j.neucom.2023.01.029

[15] J. Liu, Y. Meng, M. Fitzsimmons, and R. Zhou, "TOOL LyZNet: A Lightweight Python Tool for Learning and Verifying Neural Lyapunov Functions and Regions of Attraction," in *Proceedings of the 27th International Conference on Hybrid Systems: Computation and Control (HSCC '24)*. Association for Computing Machinery (ACM), 5 2024, pp. 1–8.

[16] H. K. Khalil, *Nonlinear systems*. Prentice Hall, Inc., 2002.

[17] H. Yang, "Optimal control and estimation," 2023, https://hankyang.seas.harvard.edu/OptimalControlEstimation.

[18] B. Biswas, D. Ignatyev, A. Zolotas, and A. Tsourdos, "Region of attraction estimation using union theorem in sum-of-squares optimization," *Preprint*, 2023, arXiv:2305.11655v1. [Online]. Available: https://arxiv.org/abs/2305.11655v1

[19] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *33rd Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2019. [Online]. Available: https://yachienchang.github.io/NeurIPS2019

[20] J. Liu, Y. Meng, M. Fitzsimmons, and R. Zhou, "Physics-informed neural network lyapunov functions: Pde characterization, learning, and verification," *arXiv preprint*, 12 2023. [Online]. Available: http://arxiv.org/abs/2312.09131



https://dspace.org/

# Comparative analysis of sum-of-squares optimization and Neural Network Lyapunov Functions for region of attraction estimation

**Chuenwongaroon, Sorachat; Zolotas, Argyrios; Ignatyev, Dmitry**

**2025-06-10**
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/med64031.2025.11073312
Chuenwongaroon S, Zolotas A, Ignatyev D. (2025) Comparative analysis of sum-of-squares optimization and Neural Network Lyapunov Functions for region of attraction estimation. In: 2025 33rd Mediterranean Conference on Control and Automation (MED), 10-13 June 2025, Tangier, Morocco, pp. 423-428

Downloaded from a DSpace repository. https://dspace.org/