IEEE Control Systems Letters paper presented at
2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Physics-Informed Extreme Learning Machine Lyapunov Functions

Ruikun Zhou, Maxwell Fitzsimmons, Yiming Meng, and Jun Liu, *Senior Member, IEEE*

***Abstract*— We demonstrate that a convex optimization formulation of physics-informed neural networks for solving partial differential equations can address a variety of computationally challenging tasks in nonlinear system analysis and control. This includes computing Lyapunov functions, region-of-attraction estimates, and optimal controllers. Through numerical examples, we illustrate that the formulation is effective in solving both low- and high-dimensional analysis and control problems. We compare it with alternative approaches, including semidefinite programming and nonconvex neural network optimization, to demonstrate its potential advantages.**

***Index Terms*— Neural networks, Lyapunov function, nonlinear systems, stability analysis, optimal control**

I. Introduction

Several longstanding challenges in nonlinear systems and control are tied to the computational difficulty of solving partial differential equations (PDEs). For example, the celebrated Hamilton-Jacobi-Bellman (HJB) equation, which characterizes the optimal value function for a controlled system, is notoriously difficult to solve in high dimensions. A closely related topic is the construction of Lyapunov functions. Despite being a century-old idea, a general approach to effectively computing Lyapunov functions remains elusive. To address this challenge, computational methods have been investigated, notably including sum-of-squares (SOS) [19] and radial basis function (RBF) [7] approaches.

More recently, due to the increased popularity of neural networks, a number of researchers have investigated the use of neural networks for computing Lyapunov functions [1], [4], [5], [8], [11], [16], [21] and solving HJB equations [9]. Most relevant to the current work, in [15], [17], the authors have demonstrated that physics-informed learning [20] with Zubov's equation [22] can outperform traditional sums-of-squares techniques [19] for computing Lyapunov functions in terms of verifiable region-of-attraction (ROA) estimates. Intuitively, such improvements come from the universal approximation properties of neural networks and the characterization of the problem in terms of PDEs. Often, training neural networks involves embracing non-convex optimization. However, a convex optimization formulation can significantly speed up training and leave no optimality gap because a global minimum can be easily achieved.

An extreme learning machine (ELM) [10] contrasts with traditional neural networks by randomly assigning weights and biases to hidden nodes and adjusting only the weights of the output layer. This approach transforms the optimization into a linear least squares problem that can be efficiently solved, exhibiting fast learning speed and strong generalization performance. In this paper, we explore the application of ELM in obtaining physics-informed neural networks to solve PDEs for computing Lyapunov functions, region-of-attraction estimates, and optimal control.

The contributions of this paper are threefold. First, we theoretically analyze ELM for providing guarantees in practical stability analysis and region-of-attraction estimates. Second, we discuss using ELM to estimate the domain of attraction for nonlinear systems via solving Zubov's PDE. Lastly, to the best of the authors' knowledge, this is the first work to thoroughly demonstrate the computational advantages of ELM in solving neural Lyapunov certificates and optimal control problems compared to existing approaches.

II. Preliminaries and problem formulation

Consider an autonomous system of form

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

and a control-affine system of the form

$$\dot{x} = f(x) + g(x)u, \tag{2}$$

$f : \mathbb{R}^d \to \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}^{n\times k}$. The state lies in $\mathbb{R}^d$ and the control input u lies in $\mathbb{R}^k$. For simplicity, we assume f and g are both continuously differentiable. We also assume $f(0) = 0$, the linearization of the system (1) is asymptotically stable, i.e., $A = Df(0)$ is Hurwitz, and the linearization of the control system (2) is stabilizable, i.e., (A, B) is stabilizable, where $B = g(0)$.

We are primarily concerned of the following computational tasks for systems (1) and (2) in this paper.

Computing Lyapunov functions: Given that $x = 0$ is an asymptotically stable equilibrium point for (1), how can we compute Lyapunov functions that not only characterize asymptotic stability of the origin but also certify as large a region of attraction estimate as possible?

Computing domain of attraction: Related to the point above, how can we estimate the regions of attraction that can be verified to be contained in the true domain of attraction

$$\mathcal{D} := \left\{ x_0 \in \mathbb{R}^d : \lim_{t\to\infty} \|\phi(t, x_0)\| = 0 \right\},$$

where $\phi(t, x_0)$ is the solution to (1) starting from $x(0) = x_0$?

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Canada Research Chairs program. This research was enabled in part by computing resources provided by the Digital Research Alliance of Canada (alliance.ca) and the Math Faculty Computing Facility at the University of Waterloo.

Ruikun Zhou, Maxwell Fitzsimmons, and Jun Liu are with the Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Emails: ruikun.zhou@uwaterloo.ca, mfitzsimmons@uwaterloo.ca, j.liu@uwaterloo.ca

Yiming Meng is with the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, USA (e-mail: ymmeng@illinois.edu, yiming.meng@uwaterloo.ca)

Computing optimal value and control: Define a cost

$$J(x_0, u) = \int_0^\infty Q(\phi(t, x_0, u)) + u^T(t) R(\phi(t, x_0, u)) u(t) dt, \tag{3}$$

where $\phi(t, x_0, u)$ is the solution to (2) under control signal u, $Q(x)$ and $R(x)$ are assumed to be positive definite functions of x. For simplicity, one can consider $Q(x) = x^T Q x$ and $R \in \mathbb{R}^{k\times k}$ for some positive definite matrices Q and R. Let $\mathcal{U}$ denote the set of admissible controls that can asymptotically stabilize the system with a finite cost. The objective is to find u^* that achieves optimal value $V^*(x) = J(x, u^*) = \inf_{u\in\mathcal{U}} J(x, u)$. The function V^* is the optimal value function. In practice, we hope to compute accurate approximations to u^* and V^* and obtain a close-to-optimal controller that is also stabilizing.

III. Physics-informed ELM Lyapunov functions for stability analysis and control

A. Neural network architecture

We consider an extreme learning machine (ELM) that is essentially a single-hidden-layer feedforward neural network:

$$V(x; \beta) = \beta^T \sigma(Wx + b), \tag{4}$$

where $W \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, $\beta \in \mathbb{R}^m$, and σ is an activation function applied elementwise. While any activation function[1] is theoretically viable, we shall exclusively use the hyperbolic tangent function $\tanh(\cdot)$ in all our examples.

For the purpose of solving partial differential equations (PDEs) using ELM, we need to compute derivatives of V with respect to x. It can be easily verified that the gradient can be computed as

$$DV(x; \beta) = \beta^T \text{diag}(\sigma'(Wx + b))W, \tag{5}$$

where diag$(\cdot)$ transforms the m-vector $\sigma'(Wx + b)$ into an $m \times m$ diagonal matrix, with $\sigma'(\cdot)$ denoting the derivative of σ, applied elementwise to $Wx + b$.

B. Physics-informed neural network and convex optimization

A physics-informed neural network (PINN) [13], [20] is essentially a neural network for solving PDEs. It minimizes errors deemed important for forcing the network to approximate the true solution of the PDE.

Consider a generic first-order PDE of the form:

$$F(x, V, DV) = 0, \quad x \in \Omega, \tag{6}$$

with the boundary condition $V = h$ on $\partial\Omega$. PINNs seek a neural network solution $V(x; \beta)$ to (6) by minimizing a loss function that encompasses the PDE and any supplementary information pertinent to the problem. The loss function is

$$\text{Loss}(\beta) = \mathcal{L}_{\text{residual}}(\beta) + \mathcal{L}_{\text{boundary}}(\beta) + \mathcal{L}_{\text{data}}(\beta), \tag{7}$$

where $\mathcal{L}_{\text{residual}}$ targets the PDE's residual error, $\mathcal{L}_{\text{boundary}}$ addresses the boundary condition error, and $\mathcal{L}_{\text{data}}$ pertains to the error in data or any auxiliary information.

[1]To achieve universal approximation guarantees, a non-polynomial activation is often required. However, for practical purposes, polynomial activation functions can still be used, e.g., to obtain a polynomial network.

In this paper, we focus on utilizing the architecture (4) to formulate a convex optimization problem for optimizing $V(x; \beta)$. This is feasible if the loss function Loss(β) is convex with respect to β. Specifically, if the PDE given in (6) is linear in V and DV, and we employ the mean square error for $\mathcal{L}_{\text{residual}}$, along with $\mathcal{L}_{\text{boundary}}$ and $\mathcal{L}_{\text{data}}$, then we are led to a linear least squares problem for optimizing β.

C. ELM neural Lyapunov functions via Lyapunov's PDE

Let $\Omega \subseteq \mathbb{R}^d$ be any open set containing the origin and contained in the domain of attraction $\mathcal{D}$. A Lyapunov function for (1) can characterized by the following PDE:

$$\dot{V}(x) := DV(x) \cdot f(x) = -\omega(x), \tag{8}$$

where ω is a positive definite function with respect to the origin, i.e., $\omega(0) = 0$ and $\omega(x) > 0$ for all $x \in \Omega \setminus \{0\}$. We consider a single boundary condition $V(0) = 0$. We refer interested readers to [15] for technical results on well-posedness and solution properties for this PDE, both in the viscosity and classical solution senses.

Clearly, the PDE given by (8) is linear in DV. Hence, we can formulate a convex optimization problem to solve it using an ELM neural network. For clarity, we explicitly present a mean square loss function as follows:

$$\begin{aligned}&\text{Loss}(\beta)\\ =&\frac{1}{N}\sum_{s=1}^{N} \|DV(x_s; \beta) \cdot f(x_s) + \omega(x_s)\|^2 + \lambda_0 \|\beta^T \sigma(b)\|^2,\end{aligned} \tag{9}$$

where $\{x_s\}_{s=1}^N \subseteq \Omega$ is a set of collocation points and $\lambda_0 > 0$ is a weight parameter for the condition $V(0) = 0$. The loss function loss(β) is clearly a quadratic function of β. Solving this to determine β provides an approximate solution to (8) in the form of (4). This is summarized in Algorithm 1. We next state a technical result that says if we solve (8) well in an approximate sense, then the function obtained as an approximate solution to (8) is a Lyapunov function for (1) in a practical sense, to be made precise below.

Algorithm 1: ELM Neural Lyapunov Function via Lyapunov's PDE

Require: f, Ω, N, m

1: Generate random W and b
2: Generate random collocation points $\{x_s\}_{s=1}^N \subseteq \Omega$
3: Finding β that minimizes (9) to form V from (4)

Proposition 1: Suppose that there exist two sequences $\{\varepsilon_R^n\} \downarrow 0$, $\{\varepsilon_0^n\} \downarrow 0$ (approaching 0 from the above as $n \to \infty$), and a sequence of continuously differentiable functions $\{V_n\}$, where each $V_n : \mathbb{R}^d \to \mathbb{R}$ such that

$$|DV_n(x) \cdot f(x) + \omega(x)| \le \varepsilon_R^n, \quad |V_n(0)| \le \varepsilon_0^n,$$

for all $x \in \Omega$. Furthermore, suppose that the sequence V_n is equicontinuous[2] at $x = 0$. Then, there exists a sequence

[2]This holds, in particular, when all V_n share a uniform Lipschitz constant or modulus of continuity around zero.

$\mu_n \to 0$ such that, for all n sufficiently large, V_n is a practical Lyapunov function in the following sense:

$$DV_n(x) \cdot f(x) < 0, \quad V_n(x) > 0, \tag{10}$$

for all $x \in \Omega$ such that $\|x\| \geq \mu_n$.

Proof: According to [12, Theorem 4.18], V_n satisfying the stated condition (10) leads to solutions being ultimately bounded, with the ultimate bound arbitrarily small, as $\mu_n \to 0$. This can be interpreted as a practical notion of stability.

Since ω is positive definite, there exists a class $\mathcal{K}$ function that $\omega(x) \geq \eta(\|x\|)$ for all $x \in \Omega$. Pick $r_n > 0$ such that $\eta(r_n) > \varepsilon_R^n$. Then

$$DV_n(x) \cdot f(x) \leq \varepsilon_R^n - \omega(x) \leq \varepsilon_R^n - \eta(r_n) < 0 \tag{11}$$

for all $x \in \Omega$ such that $\|x\| > r_n$. Note that we can choose $r_n \downarrow 0$ as $\varepsilon_R^n \downarrow 0$. Let $m_{r_n} = \min_{\|x\|=r_n} V_n(x)$. By the equicontinuity of the V_n's at 0, the fact that $r_n \downarrow 0$ and $|V_n(0)| \leq \varepsilon_0^n \downarrow 0$, we have $m_{r_n} \to 0$.

Pick any $\mu_0 > 0$ such that the closed Euclidean ball of radius μ_0, $\bar{B}_{\mu_0}$, has $\bar{B}_{\mu_0} \subseteq \Omega \subseteq \mathcal{D}$. Let $M_f = \max_{x \in \mathcal{R}(\bar{B}_{\mu_0})} \|f(x)\|$, where $\mathcal{R}(\bar{B}_{\mu_0}) := \phi([0,\infty) \times \bar{B}_{\mu_0})$ stands for the forward reachable set of (1) from $\bar{B}_{\mu_0}$, which is known to be compact [14]. Pick $\mu_n \in (r_n, \mu_0]$ such that

$$-\frac{\mu_n - r_n}{M_f}(\varepsilon_R^n - \eta(r_n)) + m_{r_n} > 0. \tag{12}$$

Clearly, for n sufficiently large, this is always possible, and we can choose $\mu_n \to 0$. We claim that $V_n(x) > 0$ for all $x \in \Omega$ such that $\|x\| > \mu_n$. To prove this, first note that it takes at least $T \geq \frac{\mu_n - r_n}{M_f}$ units of time to reach from x to the sphere $\|x\| = r_n$. Let T be the first hit time of $\phi(t,x)$ reaching the sphere $\{y \in \mathbb{R}^d : \|y\| = r_n\}$. Then, we have $\|\phi(T,x)\| = r_n$ and $T \geq \frac{\mu_n - r_n}{M_f}$. By (11) and (12),

$$\begin{aligned} V_n(x) &= V_n(\phi(T,x)) - \int_0^T DV_n(\phi(s,x)) \cdot f(\phi(s,x)) \\ &\geq m_{r_n} - T(\varepsilon_R^n - \eta(r_n)) \\ &\geq m_{r_n} - \frac{\mu_n - r_n}{M_f}(\varepsilon_R^n - \eta(r_n)) > 0, \end{aligned}$$

for all $x \in \Omega$ such that $\|x\| > \mu_n$. ∎

Remark 1: The proposition above describes: (i) a situation where a sequence of $\{V_n\}$ can be found to solve the Lyapunov PDE (8) with increasing accuracy, e.g., by increasing the number of hidden units m and the number of collocation points N; (ii) what stability guarantees can be obtained by one of these functions as a Lyapunov function.

D. ELM neural Lyapunov functions via Zubov's PDE

The result in the previous section provided a systematic way to construct a Lyapunov function on an a priori specified set *within* the domain of attraction. In this section, we use ELM to find Lyapunov functions that can provide region-of-attraction estimate close to the true domain of attraction.

The domain of attraction the equilibrium point $x = 0$ of (1) is captured by Zubov's PDE of the form

$$DV \cdot f = -\omega\psi(V)(1 - V), \tag{13}$$

where ω is defined above and ψ is a function satisfying some technical assumptions [22] (see also [3], [11], [15]). Two special cases of $\psi(s)$ are given by (i) $\psi(s) = \alpha$ or (ii) $\psi(s) = \alpha(1+s)$ for some constant $\alpha > 0$. In order to obtain a linear PDE, we choose $\psi(s) = \alpha$ for some constant $\alpha > 0$, which gives a linear Zubov's PDE:

$$DV(x) \cdot f(x) = -\alpha\omega(x)(1 - V(x)). \tag{14}$$

We refer interested readers to [15] for the well-posedness of the PDE and properties of its solutions in both the viscosity and classical senses. Let V be the solution to (14). By Zubov's theorem, the domain of attraction $\mathcal{D}$ is given by the sub-level set $\mathcal{D} = \{x \in \mathbb{R}^d : V(x) < 1\}$. Hence, obtaining an accurate neural solution of (14) implies an accurate approximation of the domain of attraction [15], [17].

We next state an ELM algorithm for solving (14). For the sake of completeness, we write down an explicit loss similar to the loss (9), but for Zubov's PDE (14), as follows:

$$\begin{aligned} &\text{Loss}(\beta) \\ =&\frac{1}{N}\sum_{s=1}^{N} \|DV(x_s;\beta) \cdot f(x_s) - \alpha\omega(x)V(x_s;\beta) + \alpha\omega(x_s)\|^2 \\ &+ \lambda_b \frac{1}{N_b}\sum_{s=1}^{N_b} \|V(y_s;\beta) - 1\|^2 + \lambda_0 \|\beta^T \sigma(b)\|^2, \end{aligned} \tag{15}$$

where both $DV(\cdot;\beta)$ and $V(\cdot;\beta)$, detailed expressions of which can be found in (5) and (4), are linear in β. We added a new term $\lambda_b \frac{1}{N_b}\sum_{s=1}^{N_b} \|V(y_s;\beta) - 1\|^2$, $\lambda_b > 0$, to describe the boundary condition. In this formulation, we assume $\mathcal{D} \subseteq \Omega$, i.e., the domain of attraction is entirely contained within Ω. Hence, at the boundary points $\{y_s\}_{s=1}^{N_b} \subseteq \partial\Omega$, we should have $V(y_s;\beta) = 1$. Thus, optimizing β remains a linear least squares problem. We state the process of solving (14) with an ELM neural network in Algorithm 2.

Algorithm 2: ELM Neural Lyapunov Function via Zubov's PDE

Require: f, Ω, N, N_b, m

1: Generate random W and b
2: Generate random collocation points $\{x_s\}_{s=1}^{N} \subseteq \Omega$
3: Generate random boundary points $\{y_s\}_{s=1}^{N_b} \subseteq \partial\Omega$
4: Finding β that minimizes (15) to form V from (4)

Remark 2: A data loss of the form $\frac{\lambda_d}{N_d}\sum_{s=1}^{N_d} \|V(z_s;\beta) - \hat{V}(z_s)\|^2$ can potentially be added to the loss function, which does not affect the linearity of the loss with respect to β. A loss like this can be beneficial when the domain of attraction is not entirely contained in Ω. In that case, specifying values of V to be equal to 1 on the boundary of Ω does not conform to the solution of Zubov's PDE. We note that in [11], a purely data-driven approach is taken to compute neural network Lyapunov functions, without taking into account any PDE loss or formal verification, whereas in [15], [16], both data and PDE losses are combined to achieve better approximation in certain numerical examples, as verified by

SMT solvers. In this paper, we show that the ELM encoding without any data loss can already achieve good results, as shall be demonstrated with numerical examples.

The next technical result states the theoretical guarantees of solving Zubov's PDE well using approximations.

Proposition 2: Let Ω be bounded. Suppose that $\mathcal{D} \subseteq \Omega$ and there exist sequences of numbers $\{\varepsilon_R^n\} \downarrow 0$, $\{\varepsilon_b^n\} \downarrow 0$, $\{\varepsilon_0^n\} \downarrow 0$, and a sequence of continuously differentiable functions $V_n : \mathbb{R}^d \to \mathbb{R}$ such that

$$|DV_n(x) \cdot f(x) - \alpha\omega(x)V_n(x) + \alpha\omega(x)| \leq \varepsilon_R^n, \forall x \in \Omega,$$
$$|V_n(0)| \leq \varepsilon_0^n, \quad |V_n(x) - 1| \leq \varepsilon_b^n, \forall x \in \partial\Omega,$$

where $\alpha > 0$ is a constant. Furthermore, suppose that $\{V_n\}$ is equicontinuous at $x = 0$. Then there exist sequences $c_n \uparrow 1$ and $\mu_n \to 0$ such that, for all n sufficiently large, V_n uniformly converges to the true solution V to (14) on $\bar{\Omega}$ and satisfies (10) for all $x \in \Omega$ such that $V_n(x) \leq c_n$ and $\|x\| \geq \mu_n$.

Proof: The proof relies on the convergence result established in [15, Proposition 4] for approximate solutions of (14) in a more general setting. By the assumptions on V_n, Proposition 4 in [15] guarantees that V_n converges uniformly to the true solution V to (14) on $\bar{\Omega}$. We know that V is a Lyapunov function for (1), with $V(0) = 0$ and $0 < V(x) \leq 1$ for $x \in \Omega$. Furthermore, we have $\mathcal{D} = \{x \in \mathbb{R}^d : V(x) < 1\}$.

We first apply analysis in the proof of Proposition 1 to the set $\Omega' = \{x \in \mathbb{R}^d : V(x) < \frac{1}{2}\} \subseteq \Omega$. On this set, we have $V_n \to V$ uniformly. Hence, for n sufficiently large, we have

$$DV_n(x) \cdot f(x) \leq -\alpha\omega(x)(1-V(x)) + \varepsilon_R^n \leq -\frac{\alpha}{4}\omega(x) + \varepsilon_R^n.$$

Similar to the proof of Proposition 1, there exists a sequence $\mu_n \to 0$ such that, for all n sufficiently large,

$$DV_n(x) \cdot f(x) < 0, \quad V_n(x) > 0, \tag{16}$$

for all $x \in \Omega'$ such that $\|x\| \geq \mu_n$. Furthermore, the choice of μ_n can be guaranteed to satisfy $\eta(\mu_n) > e_R^n$, where η is a class $\mathcal{K}$ function such that $\frac{\alpha}{4}\omega(x) \geq \eta(\|x\|)$ for all $x \in \bar{\Omega}$.

Now choose $c_n \in (0, 1)$ such that $-\alpha w(x)(1-c_n) + e_R^n < 0$ for all $x \in \bar{\Omega} \backslash \Omega'$. This is always possibly for n sufficiently large by setting $c_n < 1 - \frac{e_R^n}{\min_{x \in \bar{\Omega} \backslash \Omega'} \alpha\omega(x)}$. Furthermore, we can choose $c_n \uparrow 1$ as $e_R^n \downarrow 0$. It follows that, for all $x \in \bar{\Omega} \backslash \Omega'$ such that $V_n(x) \leq c_n$, we have

$$\begin{aligned} DV_n(x) \cdot f(x) &\leq -\alpha\omega(x)(1 - V_n(x)) + \varepsilon_R^n \\ &\leq -\alpha\omega(x)(1 - c_n) + \varepsilon_R^n < 0. \end{aligned} \tag{17}$$

Combining (16) and (17), the proof is complete. ∎

Remark 3: A simple Lyapunov argument indicates the existence of a sequence ν_n, such that $B_{\mu_n} \subseteq \{x \in \Omega : V_n(x) < \nu_n\}$. For large n, $DV_n(x) \cdot f(x) < 0$ whenever $\nu_n \leq V_n(x) \leq \mu_n$. An SMT solver can confirm this, ensuring that solutions starting within $\{x \in \Omega : \nu_n \leq V_n(x) \leq \mu_n\}$ eventually enter $\{x \in \Omega : V_n(x) < \nu_n\}$, a subset of the region of attraction for equation (1), further validated by local Lyapunov analysis (see [17]). As $c_n \to 1$ and V_n uniformly converges to V on $\bar{\Omega}$, the regions of attraction certified by V_n approach the actual domain of attraction $\mathcal{D}$.

E. ELM neural policy iteration for optimal control

In this section, we demonstrate that effectively solving Lyapunov's PDE with ELM, as detailed in Section III-C, can be leveraged for controller design. We examine the control-affine system (2) and the cost function (3).

Policy iteration (PI) imitates the challenge of directly solving the HJB equation by iteratively solving a simpler, linear PDE instead. The PI algorithm starts with an initial controller $u = \kappa_0(x)$, which is assumed to be an asymptotically stabilizing controller. For each $i \geq 0$, it repeats

1) (**policy evaluation**) Compute a value function $V_i(x)$ for the controller κ_i by solving the Lyapunov-type PDE

$$\begin{aligned} &DV_i(x) \cdot (f(x) + g(x)\kappa_i(x)) \\ &\quad = -Q(x) - \kappa_i(x)^T R(x)\kappa_i(x) \end{aligned} \tag{18}$$

   subject to $V_i(0) = 0$.
2) (**policy improvement**) Update the controller using

$$\kappa_{i+1}(x) = -\frac{1}{2}R^{-1}(x)g^T(x)DV_i^T(x). \tag{19}$$

Clearly, the challenge lies in solving (18). Note that, by

$$f_i(x) := f(x) + g(x)\kappa_i(x),$$
$$\omega_i(x) := Q(x) + \kappa_i(x)^T R(x)\kappa_i(x),$$

the PDE (18) involved in the PI algorithm reduces exactly to (8) with f replaced by the closed-loop dynamics f_i under k_i. We can use ELM to solve it by convex optimization at each iteration step i. We summarize this in Algorithm 3.

Algorithm 3: Extreme Learning Machine Policy Iteration (ELM-PI)

Require: f, g, Q, R, k_0, Ω, N, m
1: **repeat**
2: Generate random W and b
3: Generate random $\{x_s\}_{s=1}^N \subseteq \Omega$
4: Finding β that minimizes (9) with $f = f_i$ to form V_i from (4)
5: Update κ_{i+1} according to (19)
6: $i = i + 1$
7: **until** desired accuracy or max iterations reached

Remark 4: Physics-informed neural networks were proposed for policy iteration in [18]. Here, we highlight the efficient use of ELM for PI as a result of efficiently solving Lyapunov's PDE (8) (see comparison in Section IV-C).

IV. Numerical experiments

In this section, we present a set of examples to illustrate the potential advantages of utilizing the ELM approach for computing neural Lyapunov functions and optimal control, including fast learning speed and strong generalization performance. All examples are solved using the tool LyZNet [17], a Python tool that facilitates physics-informed learning of Lyapunov functions [15] and nonlinear optimal control [18], incorporating formal verification through SMT solvers [6]. Computations were mostly performed on a

2020 Macbook Pro with a 2 GHz Quad-Core Intel Core i5 and, for GPU-required comparisons, on an NVIDIA Hopper H100 SXM. Code, model equations, and parameters are available at `https://git.uwaterloo.ca/hybrid-systems-lab/lyznet/`. The test error for numerical examples is the maximum residual error over $2 \times N$ random points, where N is the number of collocation points.

A. Local stability analysis

We use a low-dimensional system (inverted pendulum) to study the impact of increasing the size of the training domain and a high-dimensional system [8] to investigate the impact of increasing the system's dimension on the difficulty of computing an ELM Lyapunov function. In all cases, we compute neural Lyapunov functions by solving Lyapunov's PDE (8) using Algorithm 1 with $\omega(x) = \|x\|^2$.

In Table I, it is evident that ELM can solve Lyapunov's PDE with significant accuracy. We note that the accuracy is measured in terms of residual error for satisfying the characterizing PDE (8), in contrast with existing neural network Lyapunov functions in the literature [5], [6], [8], [21], which use Lyapunov inequalities as loss. This crucial difference is important when it comes to region-of-attraction estimates and optimality. As the domain expands, maintaining the same accuracy level becomes increasingly difficult. Despite this, it consistently attains accuracy comparable to that of multi-layer PINNs trained via gradient descent methods, typically around 1E-3. Table II demonstrates ELM's effectiveness in solving high-dimensional problems, achieving accuracies from 1E-3 to 1E-5 for dimensions between 10 and 30.

Domain	Hidden units (m)	Collocation points (N)	Test error	Training time (s)
$[-1, 1]$	100	3,000	1.32E-5	0.02
$[-1, 1]$	200	3,000	8.62E-9	0.05
$[-2, 2]$	200	3,000	2.78E-4	0.05
$[-2, 2]$	400	3,000	2.81E-6	0.1
$[-2, 2]$	800	6,000	8.52E-8	0.5
$[-4, 4]$	800	6,000	5.03E-5	0.5
$[-4, 4]$	1,600	6,000	2.76E-6	1.7
$[-8, 8]$	3,200	6,000	6.49E-3	8.8
$[-8, 8]$	3,200	12,000	1.46E-3	12

TABLE I: Training results for the inverted pendulum.

B. Region-of-attraction estimates

We use the simple pendulum example to demonstrate the efficiency and advantages of using ELM Lyapunov functions for region-of-attraction estimates by solving Zubov's PDE (14) using Algorithm 2 in Section III-D. We verify the regions of attraction certified by the ELM Lyapunov function using dReal [6] and compare the result with the state-of-the-art approach, sum-of-squares (SOS) semidefinite programming.

We train ELM Lyapunov functions by solving Zubov's equation (14) with $m = 100$ and $m = 200$ hidden units, which takes 20 to 50 milliseconds, respectively. We summarize the verified regions of attraction in Table III, compared with that obtained with a sixth degree SOS Lyapunov function and a Taylor approximation to the vector field. ELM significantly outperforms SOS on this example.

Dim (d)	Hidden units (m)	Collocation points (N)	Test error	Training time (s)
10	1,600	6,000	1.59E-4	2
10	3,200	6,000	3.59E-5	9.3
10	3,200	12,000	3.61E-5	13
10	6,400	12,000	1.13E-5	72
20	1,600	6,000	5.58E-3	2.3
20	3,200	6,000	3.17E-3	10.6
20	3,200	12,000	3.34E-3	16.4
20	6,400	12,000	1.20E-3	127
30	1,600	6,000	1.61E-2	2.8
30	3,200	6,000	1.38E-2	11.2
30	3,200	12,000	6.67E-3	22.5
30	6,400	12,000	4.99E-3	172

TABLE II: Training results for toy high-dimensional systems [8]. We restrict the domain to $[-0.1, 0.1]^d$.

ELM ($m = 100$)	ELM ($m = 200$)	SOS
49.06%	**81.01%**	25.62%

TABLE III: Volume percentage of the verified region of attraction for a simple pendulum.

C. Optimal control

We demonstrate with benchmark nonlinear control problems—including inverted pendulum, cartpole, 2D quadrotor, and 3D quadrotor—that ELM-PI effectively solves nonlinear optimal control problems across low- to high-dimensional systems. We solve all examples with ELM-PI as outlined in Algorithm 3 of Section III-E with 10 iterations. The numbers of hidden units and collocation points are detailed below.

We compare the ELM approach with the traditional successive Galerkin approximation (SGA) method [2] and the physics-informed neural network (PINN) [18] on the inverted pendulum example over the domain $[-1, 1]^2$. For PINN-PI, we train networks with one or two hidden layers of m units for 10,000 epochs to ensure an accurate solution to the linear PDE (18). For ELM-PI and one-layer PINN-PI, we set $N = d \times m$, where $d = 2$ is the system's dimension, and $N = (d+m) \times m$ for two-layer PINN-PI. PINN-PI was evaluated with and without GPUs. Table IV shows that ELM-PI (on CPUs) provides a significant speedup, being at least 200 times faster than PINN-PI on advanced GPUs. It also demonstrates greater data efficiency and improved generalization. Additionally, SGA seems to suffer from divergence with higher-order bases.

Table V summarizes the performance of ELM-PI in solving benchmark nonlinear control problems, where the accuracy of the solution increases as m and/or N increase. It is noted that the traditional SGA approach can only solve the cartpole problem within a very small region [-0.05,0.05] and requires considerably more time. Therefore, a comparison is not provided. Readers are referred to [18] for additional

SGA			ELM-PI			PINN-PI		PINN-PI with GPU		PINN-PI (2-hidden) with GPU	
Order	Test error	Time (s)	m	Test error	Time (s)	Test error	Time (s)	Test error	Time (s)	Test error	Time (s)
2	0.15	4.80	50	**0.03**	**0.02**	0.10	79.17	0.10	113.02	0.04	143.80
4	**3.4E-3**	19.37	100	0.01	**0.05**	0.02	93.54	0.03	113.99	0.10	143.56
6	0.01	66.52	200	**2.18E-5**	**0.16**	0.04	150.98	0.04	113.2	0.13	205.47
8	2.79	212.42	400	**2.3E-6**	**0.52**	0.07	330.48	0.03	114.03	0.24	1582.15

TABLE IV: Comparison of training/computational time for ELM-PI, PINN-PI [18], and SGA [2] on the inverted pendulum example: we run ELM-PI and PINN-PI with $m = 50, 100, 200, 400$ and SGA [2] with polynomial bases of order 2, 4, 6, 8.

System	Domain	Hidden units (m)	Collocation points (N)	Test error	Training time (s)
cartpole	$[-0.2, 0.2]^4$	400	3,000	3.95E-2	1.6
cartpole	$[-0.2, 0.2]^4$	800	3,000	3.98E-3	3.99
cartpole	$[-0.2, 0.2]^4$	1,600	3,000	1.63E-3	14.95
cartpole	$[-0.2, 0.2]^4$	3,200	6,000	6.43E-4	148.58
cartpole	$[-0.2, 0.2]^4$	6,400	9,000	1.51E-4	237.41
2D quad	$[-0.2, 0.2]^6$	800	6,000	4.00E-2	9.19
2D quad	$[-0.2, 0.2]^6$	1,600	9,000	1.57E-2	35.05
2D quad	$[-0.2, 0.2]^6$	3,200	9,000	1.17E-3	142.28
2D quad	$[-0.2, 0.2]^6$	6,400	9,000	3.59E-4	1010.33
3D quad	$[-0.2, 0.2]^9$	3,200	12,000	5.33E-2	189.06
3D quad	$[-0.2, 0.2]^9$	6,400	12,000	3.64E-2	1008.57

TABLE V: Training results for ELM-PI.

experiments on PINN-PI, which, as demonstrated by the simple pendulum example, take significantly longer to train.

V. Conclusions

In this paper, we propose a physics-informed extreme learning machine framework for computing neural network Lyapunov functions in the stability analysis and control of nonlinear systems by solving the characterizing PDEs. This framework leads to a straightforward convex optimization problem solvable as a linear least squares problem. We theoretically analyze the guarantees of approximation solutions as practical Lyapunov functions that can provide arbitrarily precise ultimate bounds and region-of-attraction estimates, which are close to the true domain of attraction. We also demonstrate the effectiveness and computational efficiency of the proposed framework through a range of examples.

One potential limitation of the framework is the increase in computational difficulty as the computation domain expands. Future work could investigate domain decomposition techniques to alleviate this challenge. Additionally, exploring advanced tools for solving large-scale linear least squares problems could maximize the framework's potential. Another promising area for future research is developing verification techniques specifically designed for one-hidden-layer neural networks. Our studies indicate that existing SMT solvers [6] are not optimized yet for this purpose.

References

[1] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. FOSSIL: a software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks. In *Proc. of HSCC*, pages 1–11, 2021.

[2] Randal W Beard, George N Saridis, and John T Wen. Galerkin approximations of the generalized hamilton-jacobi-bellman equation. *Automatica*, 33(12):2159–2177, 1997.

[3] Fabio Camilli, Lars Grüne, and Fabian Wirth. A generalization of Zubov's method to perturbed systems. *SIAM Journal on Control and Optimization*, 40(2):496–515, 2001.

[4] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov control. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Nathan Gaby, Fumin Zhang, and Xiaojing Ye. Lyapunov-net: A deep neural network architecture for lyapunov function approximation. In *Proc. of CDC*, pages 2091–2096. IEEE, 2022.

[6] Sicun Gao, Soonho Kong, and Edmund M Clarke. dreal: an SMT solver for nonlinear theories over the reals. In *Proc. of CADE*, pages 208–214, 2013.

[7] Peter Giesl. *Construction of Global Lyapunov Functions Using Radial Basis Functions*, volume 1904. Springer, 2007.

[8] Lars Grüne. Computing Lyapunov functions using deep neural networks. *Journal of Computational Dynamics*, 8(2), 2021.

[9] Zheyuan Hu, Khemraj Shukla, George Em Karniadakis, and Kenji Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *arXiv preprint arXiv:2307.12306*, 2023.

[10] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[11] Wei Kang, Kai Sun, and Liang Xu. Data-driven computational methods for the domain of attraction and Zubov's equation. *IEEE Transactions on Automatic Control*, 2023.

[12] Hassan K Khalil. *Nonlinear Systems*. Pearson, 2001.

[13] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[14] Yuandan Lin, Eduardo D Sontag, and Yuan Wang. A smooth converse lyapunov theorem for robust stability. *SIAM Journal on Control and Optimization*, 34(1):124–160, 1996.

[15] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Physics-informed neural network lyapunov functions: PDE characterization, learning, and verification. *arXiv:2312.09131*, 2023.

[16] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Towards learning and verifying maximal neural lyapunov functions. In *Proc. of CDC*, 2023.

[17] Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. LyZNet: A lightweight python tool for learning and verifying neural lyapunov functions and regions of attraction. In *Proc. of HSCC*, 2024.

[18] Yiming Meng, Ruikun Zhou, Amartya Mukherjee, Maxwell Fitzsimmons, Christopher Song, and Jun Liu. Physics-informed neural network policy iteration: Algorithms, convergence, and verification. *arXiv preprint*, 2024.

[19] Antonis Papachristodoulou and Stephen Prajna. A tutorial on sum of squares techniques for systems analysis. In *Proc. of ACC*, 2005.

[20] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[21] Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*, 2022.

[22] V. I. Zubov. *Methods of A. M. Lyapunov and Their Application*. Noordhoff, 1964.