Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 58-11 (2024) 201–206

LyZNet with Control: Physics-Informed Neural Network Control of Nonlinear Systems with Formal Guarantees ⋆

Jun Liu * **Yiming Meng** ** **Ruikun Zhou** *

* *Department of Applied Mathematics, University of Waterloo (e-mail: j.liu@uwaterloo.ca; ruikun.zhou@uwaterloo.ca).*

** *Coordinated Science Laboratory, University of Illinois Urbana-Champaign (e-mail: ymmeng@illinois.edu, yiming.meng@uwaterloo.ca)*

Abstract: Optimal control for high-dimensional nonlinear systems remains a fundamental challenge. One bottleneck is that classical approaches for solving the Hamilton-Jacobi-Bellman (HJB) equation suffer from the curse of dimensionality. Recently, physics-informed neural networks have demonstrated potential in overcoming the curse of dimensionality in solving certain classes of PDEs, including special cases of HJB equations. However, one perceived limitation of neural networks is their lack of formal guarantees in the solutions they provide. To address this issue, we have built LyZNet, a Python tool that combines physics-informed learning with formal verification. The previous version of the tool demonstrated the capability for stability analysis and region of attraction estimates. In this paper, we present the tool for solving optimal control problems. We expand the functionalities of the tool to support the formulation and solving of optimal control problems for control-affine systems via physics-informed neural network policy iteration (PINN-PI). We outline the methodology that enables the learning and verification of PINN for optimal stabilization tasks. We demonstrate with a classical control example that the learned optimal controller indeed has significantly improved performance and verifiable regions of attraction.

Keywords: Neural networks, formal verification, nonlinear control, optimal control, policy iteration

1. INTRODUCTION

Optimal control has diverse applications across various fields, but effectively controlling nonlinear systems remains a fundamental challenge. A perceived bottleneck is that computational approaches for solving high-dimensional Hamilton-Jacobi-Bellman (HJB) equations, which characterize the optimal controller, often suffer from the curse of dimensionality (Bellman, 1957). While for certain special cases of HJB equations, one can overcome this curse (McEneaney, 2007; Darbon and Osher, 2016), general high-dimensional HJBs remain beyond the reach of traditional numerical methods.

Recently, physics-informed neural networks (Lagaris et al., 1998; Raissi et al., 2019) have inspired many researchers to explore neural networks for solving various PDEs. Notably, several researchers have reported that certain neural network architectures can overcome the curse of dimensionality when solving certain HJB equations Darbon et al. (2020); Hu et al. (2023); Han et al. (2018) and for finding Lyapunov functions Grüne (2021); Kang et al. (2023). On the other hand, the use of complex neural networks in safety-critical control applications is always met with questions of transparency, reliability, and formal correctness. In fact, how to certify AI-enabled CPS remains an important issue. Motivated by such promises and challenges, the authors recently developed the first tool, LyZNet, that bridges physics-informed neural network learning with formal verification. The tool (Liu et al., 2024) has demonstrated that physics-informed learning with Zubov's equation (Zubov, 1964) can outperform traditional sums-of-squares techniques (Papachristodoulou and Prajna, 2005) for computing Lyapunov functions in terms of verifiable region-of-attraction estimates. This is also in contrast to current approaches to neural Lyapunov functions, where either no formal verification is provided (Gaby et al., 2022; Grüne, 2021; Kang et al., 2023) or maximization of regions of attraction is not considered (Zhou et al., 2022; Abate et al., 2021; Chang et al., 2019).

In this paper, we extend the LyZNet tool to tackle nonlinear optimal control problems via physics-informed learning. To do so, we develop functionalities of LyZNet for formulating and solving optimal control problems in a control-affine form using policy iteration and use LyZNet to formally verify that the learned controller, while significantly improving performance, still provides stability guarantees. We present the methodology of physics-informed training, physics-informed policy iteration, and formal verification of stability using SMT solvers. We demonstrate the effectiveness of the proposed tool and methodology with an inverted pendulum example.

⋆ The research is supported in part by the Natural Sciences and Engineering Research Council of Canada and the Canada Research Chairs program.

10.1016/j.ifacol.2024.07.448

2. PROBLEM FORMULATION

We consider a nonlinear control system of the form

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

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times k}$. The control input u lies in $\mathbb{R}^k$. We are interested in designing a feedback controller $u = \kappa(x)$ such that the closed-loop system

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

satisfies a given specification. In this paper, we are primarily concerned with a stabilization task with an associated cost:

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

where $Q(x)$ and $R(x)$ are assumed to be positive definite. Control problems of this type are prevalent in control and robotics applications (Khalil, 2015; Spong et al., 2020).

3. LYZNET WITH CONTROL

The primary goal of the tool LyZNet (Liu et al., 2024) is to provide a Python framework for computing neural Lyapunov functions and supporting formal verification using satisfiability modulus theories (SMT) solvers (Gao et al., 2013). It is demonstrated in Liu et al. (2024, 2023) that neural Lyapunov functions obtained using physics-informed neural networks (PINNs) (Lagaris et al., 1998; Raissi et al., 2019) can outperform traditional sums-of-squares Lyapunov functions (Papachristodoulou and Prajna, 2005) computed using semidefinite programming in terms of region-of-attraction estimates.

In this paper, we presents a significant extension of LyZNet to cope with control synthesis problems for formal guarantees. The main idea is to leverage the universal approximation of neural networks and use it for the computation of the optimal value function and optimal controller for the system (1) with a quadratic-in-control cost (3). Our goal is to leverage PINNs for solving PDEs that characterize the solution to this optimal control.

3.1 Tool description

The tool comprises various Python modules integrated with the PyTorch machine learning framework (Paszke et al., 2019) and the dReal numerical satisfiability modulo theories (SMT) solver (Gao et al., 2013). Additionally, it utilizes the NumPy library (Harris et al., 2020) for numerical operations, the SciPy library (Virtanen et al., 2020) for scientific computing, and the SymPy library (Meurer et al., 2017) for symbolic calculations.

For a detailed description of the modules for computing and verifying neural Lyapunov functions, see Liu et al. (2024). Additional information on physics-informed computation by solving Zubov's equation, which improved region-of-attraction estimates over SOS Lyapunov functions, is available in Liu et al. (2023). Below, we outline the new features specifically introduced in the tool LyZNet for solving the optimal control problem (1) and (3):

(a) We augment the `dynamical_systems.py` module with a derived class `ControlAffineSystem` from the class `DynamicalSystem`. It provides methods for automatic computation of linearization, checking controllability and stabilizability, solutions of the linear quadratic regulator (LQR) and algebraic Riccati equation (ARE), as well as interfaces for numerical and symbolic function valuations related to the vector field f and control input matrix g.

(b) The module `neural_pi.py` is designed for solving the optimal control problem (1) and (3) using policy iteration (PI). The computation engine of policy iteration is provided by `neural_learner.py`, which implements PINN learning (Lagaris et al., 1998; Raissi et al., 2019) of Lyapunov functions. The tool allows customization of the neural network structure as well as the selection of different loss function modes. The specific mode of loss we implemented is `Lyapunov_GHJB`, which stands for a Lyapunov PDE termed the generalized Hamilton–Jacobi–Bellman (HJB) equation in the classical literature on policy iteration (Beard, 1995). Note that the PINN approach to PDE solving has been shown to scale better than classical approaches, such as the Galerkin method used in Beard (1995), which is known to suffer from the curse of dimensionality.

3.2 Tool usage

We use the following code snippet to demonstrate how to formulate a control problem and solve it using PINN-PI with LyZNet.

```
import sympy as sp
import lyznet

lyznet.utils.set_random_seed()

x1, x2 = sp.symbols('x1 x2')
symbolic_vars = (x1, x2)

f = sp.Matrix([x2, 19.6*sp.sin(x1) - 4.0*x2
    ])
g = sp.Matrix([0, 40.0])

domain = [[-1, 1]] * 2
sys_name = "pendulum"

R = 2.0*sp.eye(1)

initial_u = sp.Matrix([-(1/40)*x1 -
    (19.6/40)*sp.sin(x1)])

system = lyznet.ControlAffineSystem(f, g,
    domain, sys_name, R=R)

lyznet.neural_pi(system, initial_u=
    initial_u, num_of_iters=10, lr=0.001,
    layer=2, width=20, num_colloc_pts
    =300000, max_epoch=5)
```

The above code snippet sets up a control affine system for the inverted pendulum by specifying the uncontrolled vector field f, control input matrix g, and domain. The expressions of f and g are specified with the help of Sympy. The specification of Q and R are both optional, with default Q and R being identity matrices of appropriate dimensions. The choice of an initial controller is also optional. When the initial controller is not specified, it defaults to the linear controller obtained by solving an

LQR problem for the linearization of (3). By calling the `neural_pi` function, we solve the optimal control problem for (1) with cost (3) using policy iteration. The hyperparameters one can set include neural network size, number of iterations, learning rate, and number of epochs per iteration for optimizing the neural network, and number of collocation points (see Section 4.1 for details).

4. METHODOLOGY

In this section, we outline the methodology behind LyZNet with control, with a special emphasis on physics-informed neural network policy iteration (PINN-PI).

4.1 Physics-Informed Neural Networks

In a nutshell, a physics-informed neural network (PINN) (Lagaris et al., 1998; Raissi et al., 2019) is a neural network that solves a PDE. As an example, consider a first-order PDE of the form

$$F(x, W, DW) = 0, \quad x \in \Omega, \tag{4}$$

subject to the boundary condition $W = h$ on $\partial\Omega$.

Let $W_N(x;\theta)$ denote a multi-layer feedforward neural network parametrized by θ. Training of W_N is achieved by minimizing a specifically designed loss function. Our goal is to train W_N to solve (4) as accurately as possible for a given choice of architecture for W_N.

The loss function for optimizing θ is given by

$$\begin{aligned}\text{Loss}(\theta) = &\frac{1}{N_c}\sum_{i=1}^{N_c} F(x_i, W_N(x_i;\theta), DW_N(x_i;\theta))^2 \\ &+ \lambda_b \frac{1}{N_b}\sum_{i=1}^{N_b}(W_N(y_i;\theta) - h(y_i))^2, \\ &+ \lambda_d \frac{1}{N_d}\sum_{i=1}^{N_d}(W_N(z_i;\theta) - \hat{W}(z_i))^2,\end{aligned} \tag{5}$$

where $\lambda_b > 0$ and $\lambda_d > 0$ are weight parameters. In (5), the points $\{x_i\}_{i=1}^{N_c} \subset \Omega$ are called (interior) collocation points. We evaluate $W_N(x;\beta)$ and its derivative at these points to obtain the mean-square residual error $\frac{1}{N_c}\sum_{i=1}^{N_c} F(x_i, W_N(x_i;\beta), DW_N(x_i;\beta))^2$ for (4) at these points. The use of mean-square error is not essential; any error that captures the size of residual error can be used. The set $\{y_i\}_{i=1}^{N_b} \subset \partial\Omega$ are boundary points at which the mean-square boundary error $\frac{1}{N_b}\sum_{i=1}^{N_b}(W_N(y_i;\beta) - h(y_i))^2$ can be evaluated. Finally, we may also obtain a set of data points $\{(z_i, \hat{W}(z_i))\}_{i=1}^{N_d}$, where $\{z_i\}_{i=1}^{N_d} \subset \Omega$ and $\{\hat{W}(z_i)\}$ are approximations to the ground truth values of W at $\{z_i\}_{i=1}^{N_d}$. Such values can be obtained either from physical experiments or numerical solutions.

The premise of this approach is that by minimizing the residual and boundary losses, the physics of the problem is taken into account, and the (unique) solution of the PDE can be well approximated by the neural network W_N. Training of θ is typically achieved through gradient descent methods. By default, LyZNet employs mini-batch stochastic gradient descent with the Adam optimizer.

4.2 Physics-Informed Neural Network Policy Iteration

The above description outlines a generic solver for (4) using PINN. This section details its application in policy iteration (PI), an approach we refer to as PINN-PI.

The optimal value function for (1) with cost (3) is characterized by the HJB equation

$$\inf_{u\in\mathbb{R}^k}\left\{Q(x) + u^T R(x) u + DV(x)\cdot(f(x) + g(x)u)\right\} = 0. \tag{6}$$

We can minimize the left-hand side (which is quadratic in u) by choosing

$$u = \kappa(x) = -\frac{1}{2}R^{-1}(x)g^T(x)DV^T(x), \tag{7}$$

where $R^{-1}(x)$ is the matrix inverse of $R(x)$ and $DV(x)$ is the gradient of V written in a row vector form. In fact, the derivation of this is very similar to LQR, except for the nonlinear dependence of Q, R, f, and g on x. The control-affine nature of (1), i.e., the right-hand side of (1) is linear in u, and the quadratic-in-control cost (3) are essential for this derivation (Farsi and Liu, 2022).

With (7), the HJB equation reduces to

$$\begin{aligned}0 = &Q(x) + DV(x)\cdot f(x) \\ &- \frac{1}{4}DV(x)g(x)R^{-1}(x)g^T(x)DV^T(x).\end{aligned} \tag{8}$$

In many applications in robotics and control, we are primarily interested in asymptotic stabilization while minimizing the cost (3). It turns out that, under suitable conditions, optimality with respect to (3) and asymptotic stabilization are characterized by $u = \kappa(x)$ satisfying (8) and V satisfying (7), which is well-known in the optimal control literature (see, e.g., Farsi and Liu (2022)).

Unfortunately, despite the simplification, the HJB equation (8) remains nonlinear in V and finding its solution remains extremely challenging, especially for high-dimensional systems. Policy iteration (PI) provides a mitigation to this problem, by iteratively solving a simpler, linear PDE instead.

PI has its root in optimal control of Markov decision processes (MDPs) (Bellman, 1957; Howard, 1960) (for more recent monographs, see Bertsekas (2012, 2019). The version we introduce here for (1) and (3) dates back to Leake and Liu (1967); Saridis and Lee (1979).

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 the following:

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

$$\begin{aligned}&DV_i(x)\cdot(f(x) + g(x)\kappa_i(x)) \\ &\quad = -Q(x) - \kappa_i(x)^T R(x)\kappa_i(x)\end{aligned} \tag{9}$$

subject to $V_i(0) = 0$.

(2) (**policy improvement**) Update the controller using

$$\kappa_{i+1}(x) = -\frac{1}{2}R^{-1}(x)g^T(x)DV_i^T(x). \tag{10}$$

Clearly, the challenge lies in solving the PDE (9). Note that

$$f_c(x) := f(x) + g(x)\kappa_i(x)$$

represents the vector field of the closed-loop system (2) under the controller $u = \kappa_i(x)$. Let

$$\omega(x) := Q(x) + \kappa_i(x)^T R(x)\kappa_i(x)$$

denote the so-called stage cost, i.e., the integrand of (3). We refer the readers to Liu et al. (2023) regarding characertization of Lyapunov function through a PDE of form

$$DV \cdot f_c + \omega = 0.$$

Note that the PDE (9) is now linear in V, which is potentially easier to solve. Moreover, the fact that an initial stabilizing policy is used allows one to establish nice monotonic properties of value functions and show that stabilization of controllers is preserved through all iterations. Convergence of PI in the classical sense has been established well in the literature (Saridis and Lee, 1979; Vaisbord, 1963; Milshtein, 1964) (see also (Farsi and Liu, 2022, Chapter 3) for a recent exposition). The PhD thesis (Beard, 1995) investigated Galerkin approximations for solving (9), which is termed generalized HJB there. Galerkin approximations are known to suffer from the curse of dimensionality, as is evident by the only low-dimensional examples solved in Beard (1995). More recently, data-driven approaches have been proposed based on the PI algorithm for completely unknown dynamics (Jiang and Jiang, 2014, 2017).

PINN-PI in LyZNet implements a loss function based on (5), where F is given by

$$F(x, V, DV) := DV \cdot f_c = -\omega,$$

where f_c is the closed-loop dynamics, and ω is the state cost described above. Furthermore, we customize the loss by incorporating additional knowledge of the analytic properties of V. This is a key enabler for the convergence of the algorithm. Due to space limitations and the nature of this paper, the technical aspects of this algorithm, along with the analysis of the PI algorithm and its convergence under the notion of viscosity solutions (Crandall and Lions, 1983), will be presented in Meng et al. (2024).

4.3 Formal Verification

While neural network approximation embraces non-convex optimization and provides a powerful tool for approximating functions with remarkable flexibility, solving non-convex optimization is inherently difficult, and we are often only given "sufficiently good approximations". To ensure that such approximations are sufficient for our needs, in this setting, providing stability guarantees, we need to take additional measures.

In LyZNet, we approach this through formal verification. It supports formal verification of the resulting controller to ensure that while performance is improved, stability is guaranteed. We explain the methodology for formal verification as follows.

Local asymptotic stability: Assume that resulting controller κ renders the equilibrium point $x = 0$ exponentially stable. We also assume that the vector field of the closed-loop system $f_c(x) = f(x) + g(x)\kappa(x)$ is continuously differentiable. This is easily achieved for neural network approximation of V with smooth activation and the controller given by (7) and f and g that are already continuously differentiable. We assume the uncontrolled vector field f has an equilibrium at the origin, i.e., $f(0) = 0$, and we also need $g(0)\kappa(0) = 0$ to ensure that the origin remains an equilibrium for the closed-loop system.

Local asymptotic (exponential) stability is verified by proving that a quadratic Lyapunov function of the form $V_P(x) = x^T P x$ satisfies

$$DV_P(x) \cdot f_c(x) \leq -\varepsilon \|x\|^2$$

in a small neighborhood around the origin, where ε is a small positive number. While this condition normally causes numerical issues for the numerical SMT verifier (Gao et al., 2013) due to the use of conservative interval analysis, we circumvent this issue by essentially looking at a higher-order approximation of $DV_P \cdot f_c$ around the origin. The condition that ensures exponential stability is

$$2 \|P \cdot Dg_c(x)\| \leq r \tag{11}$$

for x in a small neighborhood around the origin. Here $r > 0$ satisfies $r < \lambda_{\min}(Q)$ and P is a solution to the linear Lyapunov equation

$$PA + A^T P = -Q,$$

for some positive definite matrix Q. The function g_c is given by $g_c(x) = f_c(x) - Ax$, where A is the Jacobian matrix of f_c evaluated at the origin. Clearly, $g_c(0) = 0$ and $Dg_c(0) = 0$. Hence, by continuity of Dg_c, condition (11) can always be verified in a sufficiently small neighborhood around the origin. In rare cases, if (11) is verified for all $x \in \mathbb{R}^n$, then we prove global exponential stability of the origin. We refer the readers to Liu et al. (2023, 2024) for a more detailed derivation of local stability verification.

Uniform attraction: Due to the use of a matrix norm in (11), the region certified for stability might be too conservative. One can enlarge the region of attraction by conducting reachability verification using the quadratic Lyapunov function V_P or a learned neural Lyapunov function W_N. LyZNet implements both functionalities. It first computes a larger region of attraction using level sets of the quadratic Lyapunov function V_P. Then, it verifies whether a neural Lyapunov function can further enlarge the region of attraction provided by the quadratic Lyapunov function.

Suppose that we have verified a local region of attraction $\{x \in \mathbb{R}^n : V_P(x) \leq c\}$ for some $c > 0$.

Let $X \subset \mathbb{R}^n$ denote a compact set on which verification takes place. We can verify the following inequalities using SMT solvers:

$$(c_1 \leq W_N(x) \leq c_2) \wedge (x \in X) \implies \dot{W}_N(x) \leq -\varepsilon, \tag{12}$$

$$(W_N(x) \leq c_1) \wedge (x \in X) \implies V_P(x) \leq c, \tag{13}$$

where $\varepsilon > 0$, $c_2 > c_1 > 0$. If these conditions are verified and the set $\mathcal{W}_{c_2} := \{x \in X : W_N(x) \leq c_2\}$ is verified to not intersect with the boundary of X, then we know $\mathcal{W}_{c_2}$ is a verified region of attraction for (1). The conclusion is intuitive, because $W_N(x)$ is strictly decreasing by at least a constant rate before reaching $\mathcal{W}_{c_1}$, which is verified to be contained in the target level set $\{x \in \mathbb{R}^n : V_P(x) \leq c\}$, from where exponential attraction to the origin is guaranteed.

5. AN EXAMPLE

In this section, we use the inverted pendulum example to illustrate the types of results obtainable with LyZNet. Additional examples will be included in the tool repository [1]. Readers are also referred to Meng et al. (2024) for further case studies in high-dimensional settings.

Example 1. Consider an inverted pendulum model

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= 19.6\sin(x_1) - 4x_2 + 40u. \end{aligned} \tag{14}$$

We consider a quadratic cost (3) with $Q = I_2$ and $R = 2$. The initial controller is chosen to be $u = -(1/40)x_1 - (19.6/40)\sin(x_1)$, which essentially cancels the nonlinearity and leads to a stable linear system.

While canceling the nonlinearity (when possible) provides an easy and intuitive stabilizing controller design, the performance of the controller will be by no means optimal according to any prescribed performance metric. We hope to use PINN-PI to optimize the performance of the controller, while preserving asymptotic stability. The code snippet for solving this problem using LyZNet is presented in Section 3.2. With the chosen hyperparameters shown, the training results after 10 iterations are depicted in Figure 1. The left panel illustrates the learned optimal value function, while the right panel displays verified level sets using a quadratic Lyapunov function and the learned value function as the Lyapunov function. In terms of the notation introduced in Section 4.3, the thick blue line indicates the level set $\mathcal{W}_{c_2}$. The thin blue line represents the level set $\mathcal{W}_{c_1}$, and the dashed red line is the level set $V_P(x) \leq c$. In this setting, the regions of attraction provided by the quadratic function and the neural Lyapunov value function are comparable, as both are clearly limited by the chosen domain X. The objective here is to improve controller performance rather than maximize the region of attraction. Readers interested in maximizing verified regions of attraction using neural Lyapunov methods by solving Zubov's equation with the tool LyZNet are referred to Liu et al. (2023, 2024).

Figure 2 shows trajectories of the closed-loop system under the learned optimal controller starting from various initial conditions in the domain. We see in Figure 3 that the learned optimal controller indeed significantly improves the initial controller in terms of accumulated cost.

6. CONCLUSION

In this paper, we introduce the tool LyZNet, now equipped with functionalities for solving optimal nonlinear control problems that ensure both performance and stability with physics-informed neural networks. Currently, this is achieved through policy iteration, assuming the availability of an initial stabilizing controller. This controller is presumed to stabilize the system within a given domain. A logical next step is to incorporate model-free reinforcement learning techniques to expand the local domain where learning for optimal stabilization is effective. The tool can also be readily extended to explore various approaches for tackling the HJB equation directly. Furthermore, we are interested in employing LyZNet to address

[1] `https://git.uwaterloo.ca/hybrid-systems-lab/lyznet`

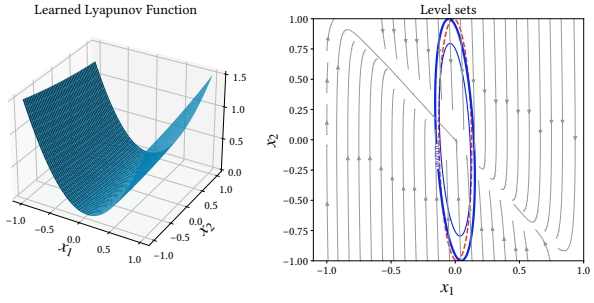

Fig. 1. Learned optimal value function and verified level sets as domain of attraction for asymptotic stability for inverted pendulum (Example 1).

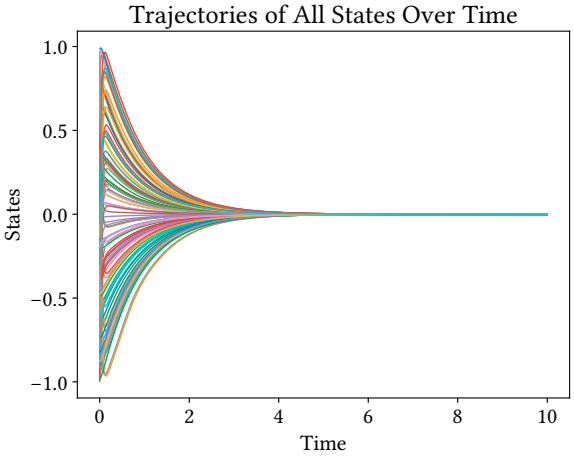

Fig. 2. Simulated trajectories of the closed-loop system under the learned optimal controller are shown, starting from various random initial conditions.

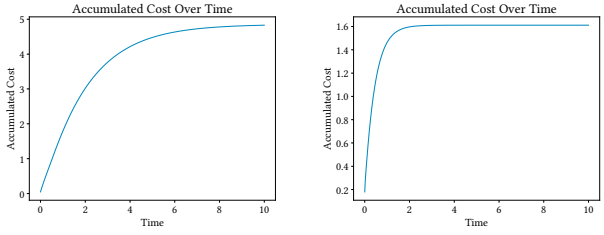

Fig. 3. Comparison of costs between the initial controller (left) and the final learned optimal controller (right) from a given initial condition.

nonlinear control problems beyond mere stabilization, such as stabilization with safety constraints (Meng et al., 2022) or temporal logic specifications (Li et al., 2022; Li and Liu, 2022).

REFERENCES

Abate, A., Ahmed, D., Edwards, A., Giacobbe, M., and Peruffo, A. (2021). FOSSIL: a software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks. In *Proc. of HSCC*, 1–11.

Beard, R.W. (1995). *Improving the closed-loop performance of nonlinear systems*. Ph.D. thesis, Rensselaer Polytechnic Institute.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
Bertsekas, D. (2012). *Dynamic Programming and Optimal Control: Volume I*, volume 4. Athena Scientific.
Bertsekas, D. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific.
Chang, Y.C., Roohi, N., and Gao, S. (2019). Neural Lyapunov control. *Advances in Neural Information Processing Systems*, 32.
Crandall, M.G. and Lions, P.L. (1983). Viscosity solutions of hamilton-jacobi equations. *Transactions of the American mathematical society*, 277(1), 1–42.
Darbon, J., Langlois, G.P., and Meng, T. (2020). Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures. *Research in the Mathematical Sciences*, 7, 1–50.
Darbon, J. and Osher, S. (2016). Algorithms for overcoming the curse of dimensionality for certain hamilton–jacobi equations arising in control theory and elsewhere. *Research in the Mathematical Sciences*, 3(1), 19.
Farsi, M. and Liu, J. (2022). *Model-Based Reinforcement Learning*. John Wiley & Sons, Ltd.
Gaby, N., Zhang, F., and Ye, X. (2022). Lyapunov-Net: A deep neural network architecture for Lyapunov function approximation. In *Proc. of CDC*, 2091–2096. IEEE.
Gao, S., Kong, S., and Clarke, E.M. (2013). dreal: an SMT solver for nonlinear theories over the reals. In *Proc. of CADE*, 208–214.
Grüne, L. (2021). Computing Lyapunov functions using deep neural networks. *Journal of Computational Dynamics*, 8(2), 131–152.
Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34), 8505–8510.
Harris, C.R., Millman, K.J., Van Der Walt, S.J., et al. (2020). Array programming with numpy. *Nature*, 585(7825), 357–362.
Howard, R.A. (1960). Markov processes and dynamic programming.
Hu, Z., Shukla, K., Karniadakis, G.E., and Kawaguchi, K. (2023). Tackling the curse of dimensionality with physics-informed neural networks. *arXiv preprint arXiv:2307.12306*.
Jiang, Y. and Jiang, Z.P. (2014). Robust adaptive dynamic programming and feedback stabilization of nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5), 882–893.
Jiang, Y. and Jiang, Z.P. (2017). *Robust Adaptive Dynamic Programming*. John Wiley & Sons.
Kang, W., Sun, K., and Xu, L. (2023). Data-driven computational methods for the domain of attraction and Zubov's equation. *IEEE Transactions on Automatic Control*.
Khalil, H.K. (2015). *Nonlinear control*. Pearson.
Lagaris, I.E., Likas, A., and Fotiadis, D.I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5), 987–1000.
Leake, R. and Liu, R.W. (1967). Construction of suboptimal control sequences. *SIAM Journal on Control*, 5(1), 54–63.
Li, Y. and Liu, J. (2022). *Formal Methods for Control of Nonlinear Systems*. CRC Press.
Li, Y., Sun, Z., and Liu, J. (2022). A specification-guided framework for temporal logic control of nonlinear systems. *IEEE Transactions on Automatic Control*, 68(4), 2002–2017.
Liu, J., Meng, Y., Fitzsimmons, M., and Zhou, R. (2023). Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification. *arXiv preprint arXiv:2312.09131*.
Liu, J., Meng, Y., Fitzsimmons, M., and Zhou, R. (2024). LyZNet: A lightweight python tool for learning and verifying neural Lyapunov functions and regions of attraction. In *Proc. of HSCC*.
McEneaney, W.M. (2007). A curse-of-dimensionality-free numerical method for solution of certain hjb pdes. *SIAM journal on Control and Optimization*, 46(4), 1239–1276.
Meng, Y., Li, Y., Fitzsimmons, M., and Liu, J. (2022). Smooth converse Lyapunov-barrier theorems for asymptotic stability with safety constraints and reach-avoid-stay specifications. *Automatica*, 144, 110478.
Meng, Y., Zhou, R., Mukherjee, A., Fitzsimmons, M., Song, C., and Liu, J. (2024). Physics-informed neural network policy iteration: Algorithms, convergence, and verification. In *Proc. of ICML*.
Meurer, A., Smith, C.P., Paprocki, M., et al. (2017). Sympy: symbolic computing in python. *PeerJ Computer Science*, 3, e103.
Milshtein, G. (1964). Successive approximations for solution of one optimal problem. *Automation and Remote Control*, 25(3), 298–306.
Papachristodoulou, A. and Prajna, S. (2005). A tutorial on sum of squares techniques for systems analysis. In *Proc. of ACC*, 2686–2700. IEEE.
Paszke, A., Gross, S., Massa, F., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.
Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019). 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.
Saridis, G.N. and Lee, C.S.G. (1979). An approximation theory of optimal control for trainable manipulators. *IEEE Transactions on systems, Man, and Cybernetics*, 9(3), 152–159.
Spong, M.W., Hutchinson, S., and Vidyasagar, M. (2020). *Robot Modeling and Control*. John Wiley & Sons.
Vaisbord, E. (1963). Concerning an approximate method for optimum control synthesis. *Avtomat. i Telemekh.*, 24(12), 1626–1632.
Virtanen, P., Gommers, R., Oliphant, T.E., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3), 261–272.
Zhou, R., Quartz, T., De Sterck, H., and Liu, J. (2022). Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*.
Zubov, V.I. (1964). *Methods of A. M. Lyapunov and Their Application*. Noordhoff.