Constructive Learning for Agile Underactuated Control

Thesis by
Ivan Dario Jimenez Rodriguez

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended December 19, 2025

Ivan Dario Jimenez Rodriguez
ORCID: 0000-0002-7230-6733

ACKNOWLEDGEMENTS

To my **parents**, thank you for your unwavering support and love throughout this journey.

To my grandmother, **Cota**, thank you for making my education possible, I would not be here without your help and love.

To my co-advisor, **Yisong Yue**, thank you for your invaluable mentoring and for always providing the right gradient steps for growth.

To my co-advisor, **Aaron Ames**, thank you for your guidance and for giving me the freedom to roam across so many robot platforms.

To **Albert**, thank you for your helpful conversations and your remarkable clarity of thought.

To **Noel**, thank you for helping me think through the ideas that would become Neural Gaits, your open demeanor and clarity of thought for robotics were invaluable.

To **Will**, thank you for your theoretical insights; whiteboarding Neural Gaits together in the lab was a highlight of this work.

This work was funded in part by AeroVironment, DARPA, the National Science Foundation, and the Caltech Center for Autonomous Systems and Technology.

ABSTRACT

Learning-based methods have achieved remarkable performance in robotic control, yet providing formal guarantees for learned controllers remains an open challenge. The central question motivating this work is how control-theoretic structure can serve as an inductive bias for learning, producing controllers that are both flexible and certifiable.

The key insight is that global properties such as stability and safety reduce to pointwise conditions that neural networks can learn to satisfy. A Lyapunov function certifies stability through a condition that must hold at every state in a region; a barrier function certifies safety through a condition that must hold throughout the safe set. Rather than optimizing over expensive trajectory rollouts, one can sample states and penalize violations of these pointwise conditions. If the conditions are satisfied throughout the relevant region of state space, the global guarantee follows.

Three complementary strategies are developed for integrating control structure with learning: training on pointwise certificate conditions, learning inputs to structured controllers that provide guarantees conditional on quantities like uncertainty estimates, and enforcing structure architecturally so that desired properties hold by construction. These strategies are validated on neural ordinary differential equations for certified stability and safety, stereo vision systems that learn perception uncertainty for safe quadruped navigation, and Koopman operator learning with guaranteed linear latent dynamics for predictive control of floating platforms.

The primary application domain is underactuated legged locomotion. Zero dynamics, the residual dynamics that remain when controlled outputs have been driven to zero, provide a framework for coordinating actuated degrees of freedom with passive dynamics. A zero dynamics policy is a learned mapping from unactuated to actuated states that defines a stable invariant manifold. Theoretical results establish existence of such policies for locally controllable systems and prove that manifold stability implies full-state stability. Hardware experiments on a bipedal walker and a hopping robot validate these results: the hopper achieves over three thousand hops across stairs, ramps, and narrow bridges with robust disturbance rejection. These experiments demonstrate that control-theoretic structure and neural network learning can be combined to achieve both formal guarantees and robust physical performance.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Albert H. Li, Ivan Dario Jimenez Rodriguez, Joel W. Burdick, Yisong Yue, and Aaron D. Ames. "KALIKO: Kalman-Implicit Koopman Operator Learning For Prediction of Nonlinear Dynamical Systems". In: *Submitted to IEEE International Conference on Robotics and Automation (ICRA)* (2026). URL: https://arxiv.org/abs/2512.03256.
A.H.L. and I.D.J.R. contributed equally as co-first authors. I.D.J.R. contributed the use of the Takens embedding and the Koopman operator as well as substantial portions of code and manuscript.

[2] William D. Compton, Ivan Dario Jimenez Rodriguez, Noel Csomay-Shanklin, Yisong Yue, and Aaron D. Ames. "Constructive Nonlinear Control of Underactuated Systems via Zero Dynamics Policies". In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. IEEE. 2024. URL: https://ieeexplore.ieee.org/document/10886411.
W.D.C. N.C.S. and I.D.J.R. contributed equally as co-first authors. I.D.J.R. contributed substantial portions of the code for training the zero dynamics policies, co-developed the algorithm and wrote sections of the manuscript.

[3] Noel Csomay-Shanklin, William D. Compton, Ivan Dario Jimenez Rodriguez, Eric R. Ambrose, Yisong Yue, and Aaron D. Ames. "Robust Agility via Learned Zero Dynamics Policies". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024. URL: https://ieeexplore.ieee.org/document/10801322.
N.C.S. W.D.C. and I.D.J.R. contributed equally as co-first authors. I.D.J.R. contributed substantial portions of the code for training the zero dynamics policies, co-developed the algorithm and wrote sections of the manuscript.

[4] Yujia Huang, Ivan Dario Jimenez Rodriguez, Huan Zhang, Yuanyuan Shi, and Yisong Yue. "FI-ODE: Certifiably Robust Forward Invariance in Neural ODEs". In: *Proceedings of the 6th Annual Learning for Dynamics and Control Conference*. PMLR. 2024. URL: https://arxiv.org/abs/2210.16940.
Y.H. and I.D.J.R. contributed equally as co-first authors. I.D.J.R. developed the control-based parts of the algorithm and theory as well as wrote sections of the manuscript.

[5] Ryan K. Cosner, Ivan D. Jimenez Rodriguez, Tamas G. Molnar, Wyatt Ubellacker, Yisong Yue, Aaron D. Ames, and Katherine L. Bouman. "Self-Supervised Online Learning for Safety-Critical Control using Stereo Vision". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 11487–11493. URL: https://ieeexplore.ieee.org/document/9812183.
R.K.C. and I.D.J.R. contributed equally as co-first authors. I.D.J.R. con-

tributed the stereo vision uncertainty estimation, substantial portions of code and manuscript.

[6] Ivan Dario Jimenez Rodriguez, Aaron D. Ames, and Yisong Yue. "LyaNet: A Lyapunov Framework for Training Neural ODEs". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 18687–18703. URL: https://proceedings.mlr.press/v162/rodriguez22a.html.
I.D.J.R. conceived the project, developed the theory, implemented the experiments and wrote the manuscript.

[7] Ivan Dario Jimenez Rodriguez, Noel Csomay-Shanklin, Yisong Yue, and Aaron D. Ames. "Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies". In: *Proceedings of the 4th Annual Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 1060–1072. URL: https://proceedings.mlr.press/v168/rodriguez22a.html.
N.C.S. and I.D.J.R. contributed equally as co-first authors. I.D.J.R. contributed substantial portions of the code for training the neural gaits, co-developed the algorithm and wrote sections of the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Part I

Introduction

Chapter 1

INTRODUCTION

This thesis develops methods for learning controllers and models that combine the flexibility of neural networks with the formal guarantees of classical control theory. The central insight is that global properties such as stability and safety reduce to pointwise conditions that neural networks can learn to satisfy. We call this approach *constructive* because it produces controllers with certificates: provable guarantees that the learned system will behave as specified. The methods are validated on agile underactuated robots including bipedal walkers and hopping platforms.

Model-based control has demonstrated its value in safety-critical applications. The propulsive landing of orbital-class rockets provided a striking example of what model predictive control could achieve when accurate dynamics models were available [2]. Lyapunov analysis certifies stability and characterizes regions of attraction. Barrier functions certify constraint satisfaction. When the control problem admits a convex formulation, optimization-based controllers can guarantee feasibility and compute globally optimal solutions in real time. These certificates provide assurance before deployment that the system will behave as specified. Deploying an uncertified controller on a rocket or spacecraft is usually considered an unacceptable risk.

Providing such guarantees for learned controllers remains an open challenge. While substantial progress has been made on neural network verification, Lyapunov neural networks, and learned barrier functions [6] (including the contributions in this thesis), these methods have not yet become standard practice in deployed learning-based systems. The gap between the formal guarantees routinely achieved in model-based control and those available for learned policies motivates the central question of this thesis: how can control-theoretic structure serve as an inductive bias for learning, producing controllers that are both flexible and certifiable?

The formal guarantees of control theory differ in kind from the probabilistic statements of statistical learning theory. A Lyapunov certificate proves that all trajectories starting in a region converge to an equilibrium. A PAC bound states that with high probability over the training distribution, the expected loss is bounded [17]. The former is a deterministic statement about a dynamical system; the latter is a probabilistic statement about an algorithm's behavior over a distribution of problems.

For safety-critical systems, the deterministic guarantee is often what is required: we need to know that this controller on this robot will satisfy this constraint, not that a typical controller trained by this procedure will likely perform well.

1.1 A Paradigm Shift in Robotic Control

This section describes how learning-based methods became competitive with model-based control for agile robotics. Results such as ANYmal [13] showed that combining domain randomization [16] and massively parallel simulation could achieve neural network policies with robust locomotion on physical hardware across varied terrains. Quadrupeds were traversing challenging terrain, recovering from disturbances and generalizing across environmental conditions without explicit model-based planning. In manipulation, learned visuomotor policies, including recent diffusion-based approaches [5], have enabled robots to perform dexterous tasks directly from camera observations. The paradigm of learning policies that map pixels to torques, once a niche research direction, became the dominant approach for many cutting edge problems in robotics.

The specific characteristics of these problems explain why learning-based methods proved so effective. In dexterous manipulation, cameras serve as the primary sensor, and mapping from images to actions is difficult to specify analytically. In the manipulation of deformable objects, while simulation fidelity has improved substantially, the sim-to-real gap for contact-rich deformable manipulation remains significant [8]. The sensitivity of deformable dynamics to material properties, friction, and contact geometry means that policies trained in simulation often fail to transfer reliably to physical systems without extensive real-world fine-tuning. In agile locomotion, contact-implicit trajectory optimization [14] can in principle handle hybrid dynamics, but the computational cost grows combinatorially with the number of potential contact modes, making real-time replanning infeasible for rich contact sequences. Hybrid zero dynamics approaches [18] circumvent this by precomputing periodic gaits, but sacrifice the ability to adapt to novel terrain or recover from large disturbances. Learned policies, by contrast, amortize computation at training time and can represent reactive feedback strategies that would require prohibitively fast replanning in an MPC framework. For such problems, the flexibility of neural network function approximators combined with large scale simulation data, offers advantages that model-based methods cannot match.

1.2 The Relevance of Structure in Learning-Based Control

This section argues that despite empirical advances, formal guarantees remain important and structure provides the path to achieving them. The observation that general-purpose learning algorithms tend to outperform methods incorporating domain-specific structure (i.e the bitter lesson) [15] has found substantial empirical support in robotics. Yet this observation concerns empirical performance, not formal guarantees. A learned policy may outperform a model-based controller on average while offering no certificate of behavior in the worst case. A full understanding of structure in learning-based control would enable deployment of learned controllers in safety-critical applications where formal guarantees are required. Certifying a controller's behavior demonstrates that we understand its fundamental structure and limitations. Without such understanding, deployments must proceed cautiously and empirically.

The gradual rollout of autonomous vehicle systems illustrates this point [11]. Despite extensive testing and strong empirical performance, these systems encounter edge cases that were not represented in their training distributions. Augmenting the dataset with unexpected scenarios is a practical response, but it does not provide the formal guarantees that would allow confident deployment in novel environments. For those who wish to not solely build effective systems but to understand and prove properties about how robots make decisions, the purely empirical approach remains insufficient.

1.3 Challenges in Agile Robotic Control

This section identifies the structural features of agile robots that make control difficult: underactuation and hybrid dynamics. Agile robots such as bipedal walkers, quadrupeds, hoppers, and dexterous manipulators are underactuated and hybrid. Underactuation means that one cannot directly command all degrees of freedom. The foot angle of a point-foot walker and the orientation of a grasped object are examples of states that cannot be directly actuated. Hybrid dynamics arise from contact: impacts with the ground, stick-slip transitions in manipulation, and the combinatorial nature of which contact points are active at any given time. These structural features make analysis difficult even before learning enters the picture.

The difficulty is not merely theoretical. Real deployments require safety under conditions that are difficult to anticipate. For example, autonomous vehicles must handle rare edge cases that do not appear in millions of miles of logged data.

Any approach that relies purely on empirical coverage of the state space will fail to provide guarantees in these tail regimes. The end result is slow and cautious deployment, where the robot datasets can be augmented with unexpected scenarios as they are encountered in the field.

Classical model-based control provides powerful tools but has structural limitations. Model Predictive Control (MPC), trajectory optimization, and Control Lyapunov Function/Control Barrier Function (CLF/CBF) methods [1] all require reasonably accurate models. Performance degrades under model mismatch and unmodeled contacts. Furthermore, real-time constraints limit the horizon and fidelity of MPC for agile locomotion. Handling perception uncertainty is difficult; most methods assume the state is accurately known. Practitioners often resort to reduced-order models (the Linear Inverted Pendulum for walking [9], the Spring-Loaded Inverted Pendulum for running [3]) that enable tractable computation but sacrifice fidelity. These approaches provide structure and, under favorable conditions, guarantees. Yet they struggle to simultaneously handle underactuation, perception uncertainty, and aggressive performance requirements.

1.4 Learning for Robotic Control: Opportunities and Limitations

This section reviews the capabilities and limitations of current learning-based approaches to robotic control. Reinforcement learning has achieved impressive results on locomotion and manipulation when abundant simulation is available. Quadrupeds like ANYmal learn to walk and recover from pushes via domain-randomized policies trained entirely in simulation [13]. Bipedal robots have demonstrated sim-to-real transfer for walking gaits [12]. These successes demonstrate what is possible in principle. However, the reliance on simulators and domain randomization means that sim-to-real transfer remains empirical rather than certified. Policies are black boxes with no explicit stability or safety conditions; failure modes are difficult to characterize. RL demonstrates the expressiveness of learned controllers but does not tell us how to systematically enforce the properties control theory cares about: stability, safety, and invariance.

Imitation learning and diffusion-based visuomotor policies have transformed manipulation and short-horizon skills. Diffusion policies can capture multi-modal action distributions from demonstration data [5]. Large vision-language-action models perform diverse manipulation tasks from natural language instructions [19]. These methods excel at mapping sensory context to desired trajectories. Yet they depend

on curated demonstrations with limited coverage of rare failure cases. Crucially, they delegate all stability and safety to whatever low-level controller happens to sit underneath, typically a PID or simple impedance controller. There is no explicit encoding of underactuated structure or safety constraints in the learned policy itself.

Foundation models add another layer to this picture. Vision-language models can reason over tasks and scenes, translating high-level instructions into waypoints or plans. But these models run at low control rates and are not designed for 1 kHz stabilization loops. They ultimately depend on a control stack beneath them to ensure physical feasibility, stability, and safety. Roboticists should then consider carefully what control structures should sit beneath these high-level models to make their actions safe and reliable.

The pattern across these approaches is consistent: current learning methods often treat the robot as a generic environment and ignore known control-theoretic structure. This leads to strong empirical performance in narrow settings but leaves open how to guarantee stability or safety, how to handle underactuation systematically, and how to reason about failure. The opportunity is to use control structure as an inductive bias or scaffold for learning, rather than discarding it.

1.5 Structure as Representation, Not Constraint

This section argues that the debate is not whether to use structure, but which structure provides the right representation for learning. The bitter lesson [15] suggests that hand-designed features are eventually outperformed by learned representations given sufficient compute and data. But in robotics, failure means hardware damage or in the case of nominally unstable systems (e.g. bipedal walkers), a total inability to collect data in the first place. We need some structure to enable data collection in the first place. Notice that successful sim-to-real RL already uses structure, just implicitly. Policies typically output target joint positions, not torques. A PID controller tracks these targets. The PID provides robustness: the learned policy provides behavior. This is already a form of structure harness for learning.

Thus we are not debating whether there should be structure or no structure. Instead we want to determine what structure provides the right representation which generalizes across the sim-to-real gap. The PID beneath an RL policy is one choice but there may be better representations which preserve control-theoretic guarantees while maintaining the flexibility of the learned components.

Control theory offers a vocabulary of structures with mathematical meaning. Lya-

punov functions certify stability: if $\dot{V} < 0$ along trajectories, the system converges to a stable set. Barrier functions certify safety: if $\dot{h} \geq -\alpha(h)$ on the boundary, the set remains invariant. Zero dynamics capture the residual dynamics when outputs are zeroed, which is the key to analyzing underactuated systems. Koopman operators linearize nonlinear dynamics by lifting to the right coordinates [10, 4]. These properties have been studied for decades and have precise mathematical characterizations.

The critical observation for this thesis is that these global properties are often characterized by pointwise conditions. For example, Lyapunov stability is a trajectory-level property, but one verifies it by checking $\dot{V} < 0$ at each state. Forward invariance is a set-level property, but one certifies it by checking barrier conditions pointwise on the boundary. If a neural network can be trained to satisfy these pointwise conditions across the relevant region of state space, the global guarantee follows. This insight connects the sampling-based nature of learning with the condition-based nature of control certificates. Figure 1.1 illustrates this approach: given user-specified goals, neural networks are trained to minimize losses derived from Lyapunov and barrier conditions, producing both controllers and certificates.

1.6 Strategies for Structure-Aware Learning

This section presents three strategies for integrating control structure with learning that organize the contributions of this thesis. These strategies are not mutually exclusive; they address different aspects of the learning-control interface.

Train on Pointwise Conditions to guarantee global properties. Trajectory-level properties such as stability, safety, and forward invariance reduce to pointwise algebraic conditions on Lyapunov or barrier functions. Rather than optimizing a task-specific loss over trajectories, one can train by sampling states and penalizing violations of these conditions. If the conditions are satisfied almost everywhere, the global guarantee holds. This approach respects the expressiveness of neural networks while using control structure in the evaluation criteria for what constitutes a good model or policy. The key advantage is that pointwise training avoids expensive trajectory rollouts during learning and provides a path to certification via verification of the trained model.

Learn Inputs to Structured Controllers. In many robotics problems, the controller architecture is well-understood. CBF-based safety filters, robust MPC, and

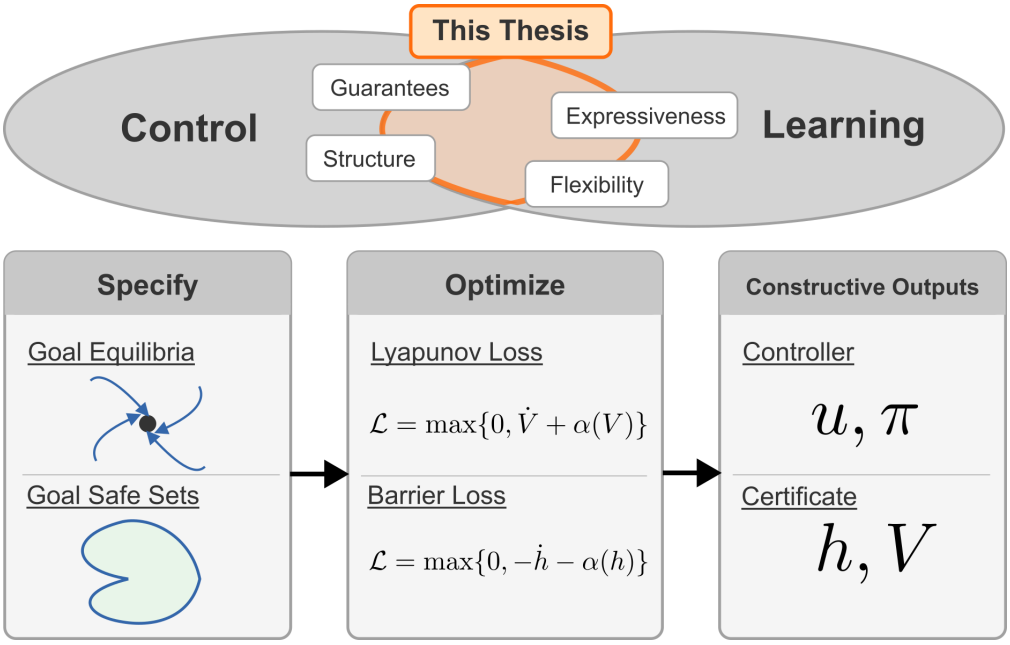


Figure 1.1: Conceptual overview of the constructive approach to learning-based control developed in this thesis. The approach combines the guarantees and structure of classical control with the expressiveness and flexibility of learning. Given user-specified goals (equilibrium points for stability, safe sets for safety), neural networks are trained to minimize Lyapunov and barrier losses that encode pointwise certificate conditions. The optimization produces both a controller (a policy π or control input u) and a certificate (a Lyapunov function V or barrier function h) that formally guarantees the desired behavior.

CLF-based stabilizers all have known guarantees if certain quantities are available: disturbance bounds, model error estimates, perception uncertainties. These quantities are often unavailable analytically but can be learned from data. Rather than replacing the controller with a neural network, one learns the quantities the controller needs to provide its guarantees. The structure lives in the control framework; learning fills the gaps. This approach is particularly valuable when the control guarantee depends on environment-specific information that varies across deployments.

Enforce Structure Architecturally. Sometimes the desired mathematical structure is known but difficult to obtain from raw data. Linear latent dynamics, Hamiltonian structure [7], passivity, and energy conservation are examples. One can build this structure into the model architecture so that any learned model automatically satisfies it, regardless of the learned parameters. Training then optimizes performance within the structural constraint. This approach trades some expressiveness for guaranteed structural properties, which can make downstream analysis and control

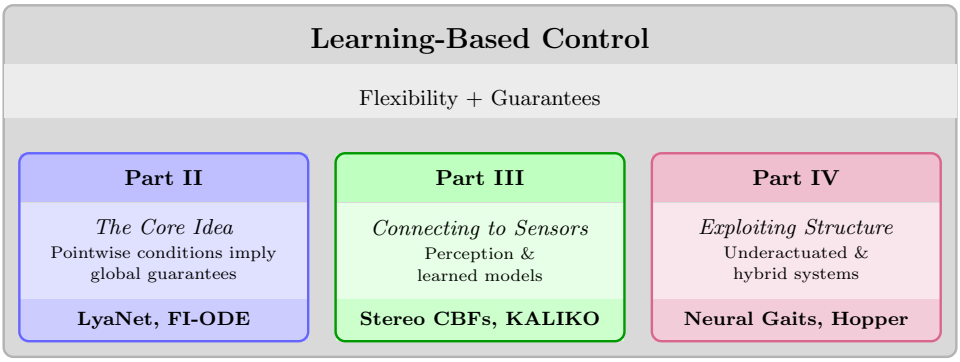


Figure 1.2: Overview of thesis contributions. Each part addresses a different aspect of the learning-based control problem. Part II develops the core idea of training neural networks to satisfy pointwise certificate conditions (Lyapunov and barrier functions). Part III addresses how to connect learned controllers to perception and how to learn dynamics models for structured control. Part IV develops Zero Dynamics Policies for underactuated systems with hybrid dynamics.

more tractable.

These three strategies are complementary. Training on pointwise conditions uses control structure in the loss function. Learning inputs to structured controllers uses learning to estimate inputs to a fixed control structure. Enforcing structure architecturally uses structure in the architecture itself. This thesis explores all three.

1.7 Thesis Contributions and Positioning

This thesis studies how control-theoretic structure can be exploited in learning to produce controllers and models for agile underactuated robots. Figure 1.2 provides an overview of the thesis contributions, organized into three parts that address progressively more challenging aspects of the learning-based control problem. The term *constructive* has a specific meaning: a control method is constructive if it synthesizes a controller together with a certificate proving that the controller achieves the desired property. Classical constructive methods include Control Lyapunov Functions, which synthesize stabilizing controllers, and Control Barrier Functions, which synthesize safe controllers. This thesis extends constructive methods to learned controllers by training neural networks to satisfy the pointwise conditions that underlie these certificates.

Contributions to Training on Pointwise Conditions to Guarantee Global Properties. The main arc of this thesis develops pointwise-condition training for in-

creasingly challenging robotic settings. LyaNet introduces a framework for training Neural ODEs to satisfy Lyapunov conditions, demonstrating that pointwise training works and that adversarial robustness emerges as a consequence of stability. FI-ODE extends this to barrier conditions for certifiable safety, producing the first provably safe Neural ODE controllers. Together these contributions lay the groundwork for using pointwise conditions to guarantee global properties in learning-based control.

Contributions to Learning Inputs to Structured Controllers. The stereo vision for safety-critical control work addresses perception uncertainty for safety-critical control. CBF-based obstacle avoidance requires knowing how uncertain the depth estimates can be, which varies with environment and sensor configuration. We learn disparity uncertainty via self-supervised multibaseline stereo consistency and use this learned uncertainty as an input to a robust CBF controller. The CBF framework provides safety; learning provides the missing model of perception error.

Contributions to Enforcing Structure Architecturally. KALIKO enforces globally linear latent dynamics (a finite-dimensional Koopman model) by construction. A Kalman filter and smoother play the role of an implicit encoder while the latent dynamics matrix and decoder are learned. This linear structure allows for interpretability of the learned nonlinear dynamics while maintaining the performance of other learned components under closed-loop control.

Zero Dynamics Policies (ZDPs) as the Exemplar of Structured Learning. The ZDP framework addresses the hardest case: underactuated systems with hybrid (impact) dynamics. We first provide a theoretical foundation for ZDPs as mappings from underactuated to actuated states that result in globally stable behavior. ZDPs show how to extend pointwise-condition training to systems where direct Lyapunov methods fail. Hardware validation on walking and hopping robots demonstrates that these results go beyond theory and simulation.

This work does not directly address integration with vision-language models or diffusion policies. The contribution is foundational: what are the right structured controllers to put beneath learned high-level policies? If PID is one answer, ZDPs may offer a more principled alternative with formal guarantees.

1.8 Thesis Structure

Chapter 2: Pointwise Conditions for Stability and Safety. This chapter introduces LyaNet, a training framework using Lyapunov loss where Monte Carlo sampling avoids trajectory rollouts. The approach is extended in FI-ODE to forward invariance via barrier conditions, with interval propagation for certification. The main results are the first certifiably safe Neural ODE controller (on a planar segway) and certified robust classification on CIFAR. The takeaway is that pointwise conditions can be trained and certified, and neural networks can provably satisfy control-theoretic properties.

Chapter 3: Learning Models for Structured Control. This chapter presents two works demonstrating how to learn inputs to structured controllers and how to enforce structure architecturally. The stereo vision for safety-critical control work develops self-supervised stereo uncertainty learning via multibaseline consistency, with a robust CBF formulation. Hardware experiments on a quadruped demonstrate maintained safety without conservatism in novel environments. KALIKO develops Koopman learning via Kalman filtering with linear latent dynamics by construction and an implicit encoder. The learned models produce interpretable eigenfunctions (orbits, limit cycles) and enable MPC for stabilization under wave disturbances. The takeaway is that when the control structure is known, learning provides missing model components such as uncertainty and dynamics.

Chapter 4: Zero Dynamics Policies for Underactuated Locomotion. This chapter develops the theoretical and practical foundations of ZDPs. First, it establishes why zero dynamics are the right reduction for underactuation, proves existence of stabilizing ZDPs, and shows that pointwise conditions on zero dynamics imply full-state stability. Neural Gaits applies CBFs on zero dynamics for bipedal locomotion with episodic learning and hardware dynamics refinement, demonstrating walking on the AMBER-3M platform with formal stability certificates. The hopper work develops discrete-time ZDPs for hybrid systems with impacts, learning via trajectory optimization and demonstrating over 3000 hops on the ARCHER hardware platform across stairs, ramps, and narrow bridges. The takeaway is that zero dynamics provide a principled dimensionality reduction, and learning policies that satisfy stability conditions on this reduced space yields guaranteed stable behavior for the full underactuated system.

Chapter 5: Discussion and Conclusion. This chapter places the contributions of the thesis in the context of the broader literature on learning-based control and robotic safety. It discusses the limitations of the proposed methods and potential future directions.

References

[1] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876.

[2] Lars Blackmore. "Autonomous precision landing of space rockets". In: *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*. Vol. 46. The Bridge Washington, DC, USA. 2016, pp. 15–20.

[3] Reinhard Blickhan. "The spring-mass model for running and hopping". In: *Journal of biomechanics* 22.11-12 (1989), pp. 1217–1227.

[4] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. "Modern Koopman Theory for Dynamical Systems". In: *SIAM Review* 64.2 (2022), pp. 229–340. DOI: 10.1137/21M1401243. eprint: https://doi.org/10.1137/21M1401243. URL: https://doi.org/10.1137/21M1401243.

[5] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *Robotics: Science and Systems (RSS)*. 2023.

[6] Charles Dawson, Sicun Gao, and Chuchu Fan. "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control". In: *IEEE Transactions on Robotics* (2023).

[7] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. "Hamiltonian neural networks". In: *Advances in neural information processing systems* 32 (2019).

[8] Feida Gu, Yanmin Zhou, Zhipeng Wang, Shuo Jiang, and Bin He. "A Survey on Robotic Manipulation of Deformable Objects: Recent Advances, Open Challenges and New Frontiers". In: *CoRR* (2023).

[9] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 1. IEEE. 2001, pp. 239–246.

[10] Bernard O Koopman. "Hamiltonian systems and transformation in Hilbert space". In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318.

[11] Philip Koopman and Michael Wagner. “Autonomous vehicle safety: An interdisciplinary challenge”. In: *IEEE Intelligent Transportation Systems Magazine* 9.1 (2017), pp. 90–96.

[12] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. “Reinforcement learning for robust parameterized locomotion control of bipedal robots”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2811–2817.

[13] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science robotics* 7.62 (2022), eabk2822.

[14] Michael Posa, Cecilia Cantu, and Russ Tedrake. “A direct method for trajectory optimization of rigid bodies through contact”. In: *The International Journal of Robotics Research* 33.1 (2014), pp. 69–81.

[15] Richard Sutton. “The bitter lesson”. In: *Incomplete Ideas (blog)* 13.1 (2019), p. 38.

[16] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[17] Leslie G Valiant. “A theory of the learnable”. In: *Communications of the ACM* 27.11 (1984), pp. 1134–1142.

[18] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. “Hybrid zero dynamics of planar biped walkers”. In: *IEEE Transactions on Automatic Control* 48.1 (Jan. 2003), pp. 42–56. ISSN: 1558-2523. DOI: 10.1109/TAC.2002.806653.

[19] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. “Rt-2: Vision-language-action models transfer web knowledge to robotic control”. In: (2023), pp. 2165–2183.

Part II

Learning Through Lyapunov-Like Conditions

Chapter 2

A LYAPUNOV FRAMEWORK FOR TRAINING NEURAL ODES

2.1 Introduction

This chapter establishes the core methodology of this thesis: training neural networks to satisfy pointwise conditions that imply global dynamical properties. Control-theoretic properties such as stability and safety can be characterized by conditions that must hold at every state in a region. A Lyapunov function certifies stability through one such condition: if $\dot{V} < 0$ everywhere in a region, then all trajectories in that region converge. This local-to-global structure suggests a natural training procedure. Rather than optimizing over trajectories, one can sample states and penalize violations of the pointwise condition. If the condition is satisfied throughout the relevant region, the global guarantee follows.

To develop the approach in a controlled setting, this chapter studies Neural Ordinary Differential Equations (Neural ODEs) for classification. Neural ODEs interpret the forward pass of a deep network as the solution to a differential equation, where the hidden state evolves continuously through a learned vector field [7]. This interpretation, first identified in connections between ResNets and ODEs [15, 14, 31, 20], enables the direct application of dynamical systems analysis to neural network inference. In particular, one can ask whether the inference dynamics are stable in the control-theoretic sense: do all trajectories converge to states that yield correct predictions?

The classification setting provides a useful testbed for several reasons. The dynamics are fully specified by the learned parameters, without the complications of physical constraints, contact, or underactuation that arise in robotic systems. Large labeled datasets allow systematic evaluation of whether Lyapunov-based training achieves competitive accuracy. Adversarial robustness provides a measurable proxy for the kind of robustness to perturbations that matters in physical control systems. If pointwise Lyapunov training fails to work in this simplified setting, it is unlikely to succeed in the harder settings that follow. Conversely, success here provides evidence that the methodology extends to more challenging domains.

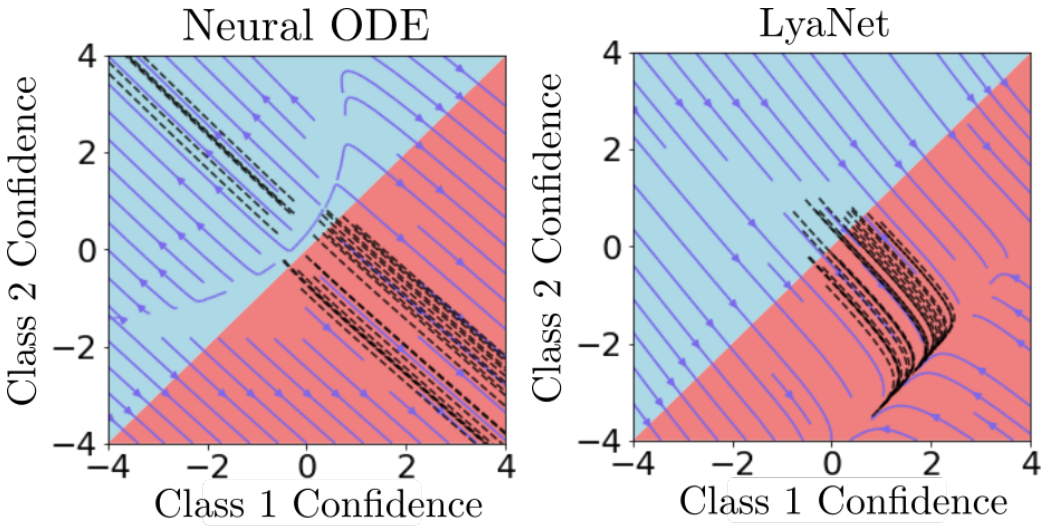


Figure 2.1: **Comparing learned dynamics** on a toy prediction task. The quiver lines show the dynamics (Equation (2.2)) of a 2-D state space, the dotted black lines show state trajectories from 100 initial conditions (Equations (2.1) and (2.2)), and the background coloring shows the class label from the output layer (Equation (2.3)), with red being the correct label. **Left:** a Neural ODE does not learn stable dynamics, where some dynamics point towards the incorrect prediction (blue region), and are especially sensitive around the initial conditions. **Right:** an identical model trained with LyaNet has much more stable dynamics that always smoothly point towards the correct prediction (red region).

Neural ODE Model Class. Inference or the "forward pass" uses a continuous-time ODE, parameterized by $\boldsymbol{\theta}$:

$$\boldsymbol{x}(0) = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{o}), \qquad \text{(input layer)} \tag{2.1}$$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}, t), \quad \text{(continuum of hidden layers)} \tag{2.2}$$

$$\hat{\boldsymbol{y}}(t) = \boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}(t)). \quad \text{(output layer)} \tag{2.3}$$

Given input $\boldsymbol{o}$, one makes a prediction (i.e., inference) by solving the ODE specified by Equations (2.1) & (2.2), and computing the output via Equation (2.3). Without loss of generality, we assume (2.2) evolves in the time interval $[0, 1]$, i.e., $\hat{\boldsymbol{y}}(t)$ is typically computed at $t = 1$, although one could in principle compute $\hat{\boldsymbol{y}}(t)$ at any $t \in [0, 1]$. We also assume that $\boldsymbol{o} \in \mathbb{R}^n$, $\boldsymbol{x} \in H \subset \mathbb{R}^k$, $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^l$, and $\boldsymbol{y} \in \mathbb{R}^m$. The state space H is assumed to be bounded and path connected.

Motivation. The standard approach for training Neural ODEs is to differentiate through the ODE solution using the adjoint method [7, 17]. This approach does not impose any structure on the learned dynamics, and as a result the inference dynamics can exhibit unstable behavior. Figure 2.1 illustrates the problem. In the

left plot, a Neural ODE trained with standard methods has learned dynamics that do not reliably converge toward the correct prediction (red region) throughout the state space. Some trajectories wander through regions of incorrect prediction before eventually arriving at the correct answer; others are highly sensitive to perturbations of the initial condition. Such instability leads to slower convergence and fragile predictions. Our goal is to learn dynamics like those in the right plot, which reliably converge to the correct prediction from a robust set of initial conditions.

The connection to control theory is direct. In control, a Lyapunov function certifies that a dynamical system converges to a desired equilibrium or set. Here, the "desired set" consists of states that yield correct predictions under the output map (2.3). If we can train the inference dynamics to satisfy a Lyapunov condition with respect to this set, the resulting Neural ODE will have stable inference that converges to correct predictions. The Lyapunov condition is a pointwise inequality: it must hold at each state, but checking it does not require integrating trajectories. This enables a Monte Carlo training approach that samples states and penalizes violations, avoiding the computational cost of backpropagating through an ODE solver.

Contributions. This chapter develops LyaNet, a framework for training Neural ODEs with provably stable inference dynamics. The contributions are as follows:

- We formulate a Lyapunov loss that captures the degree to which the learned dynamics violate exponential stability conditions. Minimizing this loss encourages the dynamics to converge quickly toward correct predictions.
- We prove that dynamics satisfying the Lyapunov condition are exponentially stable, meaning trajectories converge to correct predictions at a guaranteed rate. We further show that Lyapunov stability implies a form of adversarial robustness: bounded perturbations to the initial condition produce bounded deviations in the trajectory.
- We develop practical algorithms for optimizing Lyapunov loss, including a Monte Carlo approach that samples states rather than integrating trajectories. This exploits the local-to-global structure of Lyapunov conditions: the global convergence guarantee follows from a pointwise condition holding throughout the state space.
- We evaluate LyaNet on image classification benchmarks. Compared to standard Neural ODE training, LyaNet achieves competitive or superior prediction

accuracy with improved adversarial robustness. The inference dynamics converge faster, allowing early termination without sacrificing accuracy.

Broader Context. The methodology developed here extends beyond classification to the robotic control problems that motivate this thesis. The next chapter applies the same pointwise-condition approach to barrier functions for safety, producing the first certifiably safe Neural ODE controllers. Part IV applies these ideas to underactuated locomotion, where the dynamics are physical rather than learned and the stakes of instability are hardware damage rather than misclassification. The classification experiments in this chapter establish that pointwise training works in principle; the subsequent chapters demonstrate that it scales to the systems that motivated this thesis.

2.2 Preliminaries

Additional Details on Neural ODEs

To have unique solutions for all time, it is sufficient for an ODE to have an initial condition and a globally Lipschitz time derivative as shown in Equation (2.1) and Equation (2.2), respectively. We thus assume that the dynamics functions we learn are globally uniformly Lipschitz. This assumption is not overly onerous since most neural networks are compositions of globally Lipschitz preserving functions such as ReLUs, convolutions, max or affine functions.

Equation (2.2) generalizes the original Neural ODE formulation by making $\boldsymbol{f}$ directly depend on $\boldsymbol{o}$. This generalization is sometimes referred to as an Augmented Neural ODE [13] or Data-controlled Neural ODE [22].

Connection to ResNet. One can think of the hidden layers in a ResNet architecture as a discrete-time Euler approximation to Equation (2.2) [15, 14], where each discretized hidden layer $\boldsymbol{x}_t$ is modeled as:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-\delta} + \delta \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-\delta}, \boldsymbol{o}, t). \quad \text{(ResNet hidden layer)} \tag{2.4}$$

One obtains $\frac{\boldsymbol{x}_t - \boldsymbol{x}_{t-\delta}}{\delta} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-\delta}, \boldsymbol{o}, t)$ by isolating $\boldsymbol{f}$ in Equation (2.4) . As $\delta \to 0$ we recover Equation (2.2) (assuming continuity of $\boldsymbol{f}$). Although the original ResNet architecture does not have an explicit time step δ, each ResNet layer can be thought of as learning discrete-time dynamics with a fixed step size (i.e., an Euler approximation to the continuous-time dynamics in Equation (2.2)).

Supervised Learning as Inverse Control

We consider the standard supervised learning setup, where we are given a training set of input/output pairs, $(\boldsymbol{o}, \boldsymbol{y}) \sim D$, and the goal is to find a parameterization of our model that minimizes a supervised loss over the training data:

$$\underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} \quad \sum_{(\boldsymbol{o},\boldsymbol{y})\sim D} \mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{o}}(1), \boldsymbol{y}), \tag{2.5}$$

where $\hat{\boldsymbol{y}}_{\boldsymbol{o}}(1)$ is shorthand for Equations (2.1) to (2.3).

As is typical in deep learning, the standard approach for training Neural ODEs is via backpropagation through Equation (2.5). The "end-to-end" training optimization problem is equivalent to the following finite-time optimal control problem (using just a single $(\boldsymbol{o}, \boldsymbol{y})$ for brevity):

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \quad & \mathcal{L}(\hat{\boldsymbol{y}}(1), \boldsymbol{y}), \\ \text{s.t.} \quad & \frac{\partial \boldsymbol{x}}{\partial t} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}(t), \boldsymbol{o}, t), \\ & \boldsymbol{x}(0) = \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{o}), \\ & \hat{\boldsymbol{y}}(1) = \boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}(1)). \end{aligned} \tag{2.6}$$

Differentiating $\boldsymbol{\theta}$ through (2.6) requires computing $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}}$, which was shown to be possible from rollouts of the dynamics using either backpropagation through the solver or the adjoint method [7, 14].

Challenges. In Equation (2.6), there is no explicit penalty or regularization for intermediate states of the dynamical system. As such, even if (2.6) is optimized, the resulting dynamics $\boldsymbol{x}(t)$ can exhibit problematic behavior. Indeed, one can observe such issues in Figure 2.1 where the Neural ODE dynamics are unstable, and in Figure 2.4 where the Neural ODE learns a fragile solution. Furthermore, training in this fashion can have high computational costs from generating roll-outs and the inherent numerical difficulty of integrating unstable dynamics. Our LyaNet approach addresses these limitations via a control-theoretic learning objective.

Lyapunov Conditions for Stability

In control theory, a stable dynamical system implies that all solutions in some region around an equilibrium point flow to that point. Lyapunov theory generalizes this concept by reasoning about convergence to states that minimize a potential function V. These potential functions are a special case of dynamic projections [34].

Definition 1 (Dynamic Projection (adapted from [34])). *Let $H \subseteq \mathbb{R}^k$ be compact. A continuously differentiable function $V : H \to \mathbb{R}_{\geq 0}$ is a dynamic projection if the optimal set $X^* = \{\boldsymbol{x} \in H : V(\boldsymbol{x}) = 0\}$ is non-empty, and there exist constants $\underline{\sigma}, \overline{\sigma} > 0$ satisfying:*

$$\forall \boldsymbol{x} \in H : \ \underline{\sigma} \cdot \text{dist}(\boldsymbol{x}, X^*) \leq V(\boldsymbol{x}) \leq \overline{\sigma} \cdot \text{dist}(\boldsymbol{x}, X^*), \tag{2.7}$$

where $\text{dist}(\boldsymbol{x}, X^*) = \min_{\boldsymbol{x}^* \in X^*} \|\boldsymbol{x} - \boldsymbol{x}^*\|$.

We can now define exponential stability with respect to V. As is common in many nonlinear convergence analyses, studying the behavior of a potential function is typically much easier than directly reasoning about the full dynamics.

Definition 2 (Exponential Stability). *We say that the ODE defined in Equations* (2.1) *and* (2.2) *is* ***exponentially stable*** *if there exists a dynamic projection V (with optimal set X^*) and a constant $\kappa > 0$, such that all solution trajectories $\boldsymbol{x}(t)$ of the ODE for all $t \in [0, 1]$ satisfy:*

$$V(\boldsymbol{x}(t)) \leq V(\boldsymbol{x}(0))e^{-\kappa t}. \tag{2.8}$$

By Equation (2.7)*, this implies* $\text{dist}(\boldsymbol{x}(t), X^*) \leq \frac{V(\boldsymbol{x}(0))}{\underline{\sigma}} e^{-\kappa t}$*, i.e., exponential convergence to the optimal set X^*.*

Exponential stability implies that the dynamics converge exponentially fast to states with minimal V. Later, we will instantiate V using supervised loss.[1] Exponential stability is desirable because: 1) it guarantees fast convergence to desired states (as defined by V) after integrating for finite time (e.g., for $t \in [0, 1]$); and 2) it has implications for adversarial robustness, discussed later.

In order to guarantee exponential stability, we will impose additional structure, as described in the following theorem.

Theorem 1 (Exponentially Stabilizing Control Lyapunov Function (ES-CLF) Implies Exponential Stability (Ames et al. [1])). *For the ODE in Equations* (2.1) *and* (2.2)*, a continuously differentiable dynamic projection V is an ES-CLF if there is a constant $\kappa > 0$ such that:*

$$\min_{\boldsymbol{\theta} \in \Theta} \left[\left. \frac{\partial V}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}}^{\top} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}, t) + \kappa V(\boldsymbol{x}) \right] \leq 0 \tag{2.9}$$

[1] For instance, in Figure 2.1, we want $V \approx 0$ only within the red region. Our definition of V will also involve Equation (2.3).

holds for all $\boldsymbol{x} \in H$ *and* $t \in [0, 1]$. *The existence of an ES-CLF implies that there is a* $\boldsymbol{\theta} \in \Theta$ *that can achieve:*

$$\left.\frac{\partial V}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}}^{\top} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}, t) + \kappa V(\boldsymbol{x}) \leq 0, \tag{2.10}$$

and furthermore the ODE using $\boldsymbol{\theta}$ *is exponentially stable with respect to* V *(and constant* κ*).*

In other words, we require the ODE to satisfy additional structure as specified by Equations (2.9) and (2.10), in order for the ODE to be exponentially stable w.r.t. V. In Section 2.3, we will develop a learning framework that encourages finding parameters $\boldsymbol{\theta}$ that satisfy Equation (2.10).

Local-to-Global Contraction Structure. Equation (2.10) is essentially a contraction condition on V with respect to time (with κ controlling the rate of contraction), whereby the rate of decrease in V should be (at least) proportional to its current value. One can further interpret Equation (2.10) as a local invariance property: the condition only depends on the local state $\boldsymbol{x}$ rather than, say, the entire trajectory. In essence, Lyapunov theory exploits this local-to-global structure so that establishing a local contraction-based invariance implies global stability.

Control Lyapunov Functions. The potential function used in Theorem 1 is called a control Lyapunov function (CLF). The main difference between CLFs and classic Lyapunov functions is the additional minimization over $\boldsymbol{\theta}$. From the perspective of control theory, one can think of the parameters $\boldsymbol{\theta}$ as a "controller" and Theorem 1 establishes conditions when there exists such a controller that can render the dynamics exponentially stable.[2] An exponentially stabilizing CLF (ES-CLF) is a CLF where κ in Equations (2.9) and (2.10) is strictly greater than zero.

Connection to Learnability. The minimization in Equation (2.9) can be interpreted as a statement about learnability or realizability. Satisfying Equation (2.9) equates to the family of parameters Θ realizing exponential stability with respect to V (i.e., some $\boldsymbol{\theta} \in \Theta$ satisfies Equation (2.9)). If this potential function corresponds to a supervised loss, this means that supervised learning problem is learnable by this function class (i.e. some $\boldsymbol{\theta} \in \Theta$ achieves low supervised loss). A natural way to prove that a V is an ES-CLF is to find (i.e., learn) a parameter $\boldsymbol{\theta}$ that satisfies Equation (2.10). Note that stability is a much stronger condition than achieving low

[2] Conventional applications of Theorem 1 focus on designing controllers to stabilize a given physical system [1].

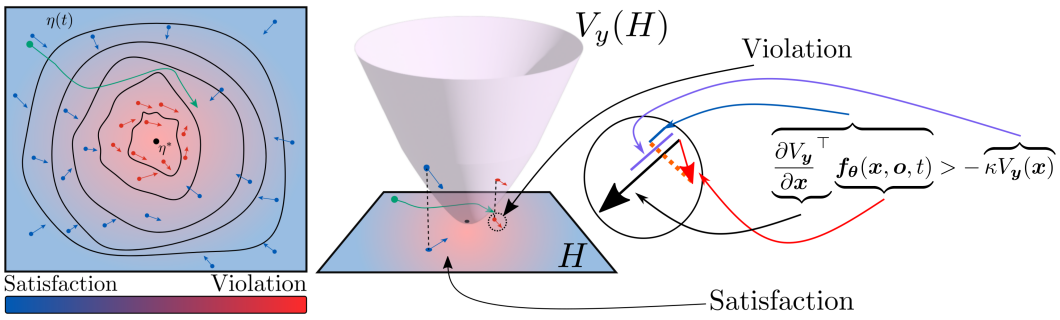


Figure 2.2: **Left:** a phase space plot a dynamical system (Equation (2.2)), along with level sets of the potential function V_y (Equation (2.11)) which in this example is minimized at x^*. The blue arrows represent flows that locally satisfy the Lyapunov exponential stability condition Equation (2.10)), while red arrows violate it. The background coloring indicates a local measure of this violation, as captured in the point-wise Lyapunov loss (Equation (2.12)). The green line denotes an example trajectory $x(t)$ (Equation (2.2)), which in this case does not stabilize to x^* which has minimal V_y. **Right:** depicting the geometric correspondence between the point-wise Lyapunov loss and the 1-D projected dynamics of V_y, where the inequality is a re-arrangement of terms. At any state x, if the point-wise Lyapunov loss is positive (i.e., the depicted inequality is satisfied) then the 1-D projected dynamics at $V_y(x)$ is not guaranteed to be exponentially stabilizing to 0. Conversely, choosing a θ to break the depicted inequality (and thus achieve zero Lyapunov loss) will guarantee exponential stability.

supervised loss, which implies that the set of stably fitting models is a subset of all models that fit the data.

Learning Exponentially Stable Systems. In this paper, we aim to shape a (highly overparameterized) system to satisfy a Lyapunov condition for stabilizing to predictions with minimal supervised loss. Prior work has explored learning dynamical systems that are stable but not explicitly to a correct prediction [31, 21, 4, 5], which can lead to a tension between stability and accuracy. Compared to conventional work in Lyapunov analysis, our goal can be viewed as the dual of the more common goal of finding a Lyapunov function to associate with a pre-specified system.

2.3 LyaNet Framework

We now present LyaNet, our Lyapunov framework for training ODEs of the form specified by Equations (2.1) to (2.3). As alluded to in Section 2.2, our goal is to find parameters θ of the ODE to satisfy the Lyapunov exponential stability condition in Theorem 1 with respect to a potential function V. We develop the formulation in two steps:

1. Section 2.3: For a given supervised loss, we define an appropriate potential function V.

2. Section 2.3: For that V, we define the Lyapunov loss which captures the degree of violation from satisfying the contraction condition in Equation (2.10) that implies exponential stability.

Theoretically, we show that optimizing the Lyapunov loss implies the learned ODE exponentially stabilizes to predictions with minimal supervised loss (Theorem 2), which in turn implies a novel adversarial robustness guarantee (Theorem 3).[3] In other words, if we find a $\boldsymbol{\theta} \in \Theta$ that achieves zero Lyapunov loss, then Equation (2.10) will be satisfied. We present practical learning algorithms in Section 2.4.

Potential Function for Supervised Loss

For a given input-output pair $(\boldsymbol{o}, \boldsymbol{y})$ and supervised loss $\mathcal{L}$, define the following potential function:

$$V_{\boldsymbol{y}}(\cdot) := \mathcal{L}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\cdot), \boldsymbol{y}), \tag{2.11}$$

where the input to $V_{\boldsymbol{y}}$ depends on $\boldsymbol{o}$, and $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ is the output layer from Equation (2.3). Typically, one would input the hidden state at some time t, $\boldsymbol{x}_{\boldsymbol{\theta}}(t)$, where the $\boldsymbol{\theta}$ subscript denotes the parameters in Equations (2.1) and (2.2). An additional technical requirement is the continuous-differentiability of $\boldsymbol{\psi}_{\boldsymbol{\theta}}$, which is satisfied by typical network architectures.

Implications. Recall that the model's prediction is defined as $\hat{\boldsymbol{y}}(t) = \boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}(t))$. Therefore, V in Equation (2.11) is minimized when the dynamical system converges to states $\boldsymbol{x}$ with zero supervised loss i.e. $(V_{\boldsymbol{y}}(\boldsymbol{x}) = \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})) = 0$).[4] In the following, we further develop the formulation so that we can interpret V as an exponentially stabilizing control Lyapunov function, and thus invoke Theorem 1 to prove exponential stability of the inference dynamics to a loss-minimizing prediction.

Truncated Cross Entropy Loss. In this paper, we focus on the standard cross entropy loss that is widely used for classification. As a theoretical technical detail, our formulation and analysis use the truncated cross entropy loss defined as: $\mathcal{L}(\cdot) := \max\{0, \mathcal{L}_{ce}(\cdot) - \gamma\}$, where $\mathcal{L}_{ce}$ is cross entropy, and $\gamma > 0$ is a small constant. The

[3] Our results assume the function class is expressive enough to minimize the Lyapunov loss, which we expect from overparameterized dynamical systems.

[4] We later show in Figure 2.3 the convergence rate of inference, which we can exactly interpret as V for cross entropy loss.

main difference is that the truncated version attains $\mathcal{L} = 0$ at a finite point (whereas cross entropy only attains 0 at infinity), which simplifies the stability analysis. In practice, we can choose γ to be machine precision, and ignore it in the learning algorithms.

Dynamic Projections. Following the arguments in Section 2.2, the first step is showing that V in Equation (2.11) is a dynamic projection, which is analyzed in the following for truncated cross entropy. See Section 2.A.1 for proof.

Lemma 1. *For the dynamical system described in Equations* (2.1) *to* (2.3) *with compact state space H,* $V_{\boldsymbol{y}}(\boldsymbol{x})$ *in Equation* (2.11) *with* $\mathcal{L}$ *as truncated cross entropy loss is a dynamic projection with optimal set* $X^* = \{\boldsymbol{x} \in H : \mathcal{L}_{ce}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) \leq \gamma\}$.

Lyapunov Loss

In order to use the potential function V from Section 2.3 to impose stability on ODE training, the remaining step is to satisfy the condition in Equation (2.10). To do so, we define the Lyapunov loss, starting with the point-wise version.

Definition 3 (Point-wise Lyapunov Loss). *For the dynamical system defined in Equations* (2.1) *to* (2.3)*, a single input-output pair* $(\boldsymbol{o}, \boldsymbol{y})$*, and dynamic projection* $V_{\boldsymbol{y}} : H \to R_{\geq 0}$ *from Equation* (2.11)*, the point-wise Lyapunov loss is:*

$$\mathscr{V}(\boldsymbol{o}, \boldsymbol{y}, \boldsymbol{x}, t) \coloneqq \max\left\{0, \frac{\partial V_{\boldsymbol{y}}}{\partial \boldsymbol{x}}^{\top} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}, t) + \kappa V_{\boldsymbol{y}}(\boldsymbol{x})\right\}. \tag{2.12}$$

Note that the point-wise Lyapunov loss $\mathscr{V}$ is exactly the violation of the local invariance property in Equation (2.10). Intuitively, for an input-output $(\boldsymbol{o}, \boldsymbol{y})$, if $\mathscr{V}$ is zero for all $\boldsymbol{x} \in H$ and $t \in [0, 1]$, then Equation (2.9) holds everywhere, and thus Theorem 1 implies that the inference dynamics converge exponentially to a loss minimizing prediction. Figure 2.2 provides a depiction of this intuition, and Theorem 2 formalizes it. The Lyapunov loss then applies the point-wise Lyapunov loss to all training examples and possible states.

Definition 4 (Lyapunov Loss). *For the dynamical system defined in Equations* (2.1) *to* (2.3)*, a dataset of input-output pairs* $(\boldsymbol{o}, \boldsymbol{y}) \sim D$*, dynamic projection* $V_{\boldsymbol{y}} : H \to R_{\geq 0}$*, and* $\mathscr{V}$ *from Equation* (2.12)*, the Lyapunov loss is:*

$$\mathscr{L}(\boldsymbol{\theta}) \coloneqq \mathop{\mathbb{E}}_{(\boldsymbol{o}, \boldsymbol{y}) \sim D}\left[\int_0^1 \mathscr{V}(\boldsymbol{o}, \boldsymbol{y}, \boldsymbol{x}_{\boldsymbol{\theta}}(\tau), \tau) d\mu(\tau)\right]. \tag{2.13}$$

Theorem 2. *Consider the setting in Definition 4. If there exists a parameter* $\boldsymbol{\theta}^* \in \Theta$ *of the dynamical system that attains* $\mathscr{L}(\boldsymbol{\theta}^*) = 0$*, then for each* $(\boldsymbol{o}, \boldsymbol{y}) \sim D$*:*

1. *The potential function* $V_{\boldsymbol{y}}(\cdot)$ *in Equation* (2.11) *is an exponentially stabilizing control Lyapunov function with* $\boldsymbol{\theta}^*$ *satisfying Equation* (2.10)*.*

2. *For* $t \in [0, 1]$*, the inference dynamics satisfy the following convergence rate w.r.t. the supervised loss* $\mathcal{L}$*:*

$$\mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{\theta}}(t), \boldsymbol{y}) \leq \mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{\theta}}(0), \boldsymbol{y}) e^{-\kappa t}, \tag{2.14}$$

   where $\hat{\boldsymbol{y}}_{\boldsymbol{\theta}}$ *is the output of Equation* (2.3) *with subscript* $\boldsymbol{\theta}$ *denoting all parameters in Equations* (2.1) *to* (2.3)*.*

Theorem 2 is essentially a consistency result between minimizing Lyapunov loss and guaranteeing exponential stability on converging to states $\boldsymbol{x}$ that have low supervised loss on the training examples. See Section 2.A.1 for proof. Future directions include analyzing when the Lyapunov loss is only approximately minimized, as well as generalization.

Choosing κ. The hyperparameter κ corresponds to the convergence rate of the loss dynamics. A larger κ enables faster convergence. However, as will come up in Section 2.4, larger κ can also make the learning problem more challenging, since the dynamics will have larger magnitude.

Adversarial Robustness

One attractive aspect of connecting learnable ODEs to control theory is the ability to leverage concepts in robust control. Fundamentally, robust control provides guarantees that a system will remain stable under perturbations, which is the same kind of guarantee studied in adversarially robust machine learning [38, 26, 11, 29]. One interesting contrast with prior work on adversarially robust learning is that LyaNet does not explicitly optimize for adversarial robustness, but rather the robustness guarantee presented here directly follows from optimizing Lyapunov loss.

Definition 5 (δ-Stable Inference Dynamics for $(\boldsymbol{o}, \boldsymbol{y})$). *A dynamical system defined in Equations* (2.1) *to* (2.3)*, with global Lipschitz constant L on* $\boldsymbol{x}(t)$ *and associated dynamic projections* $V_{\boldsymbol{y}}$ *(Equation* (2.11)*) has* δ*-stable inference dynamics for example* $(\boldsymbol{o}, \boldsymbol{y})$ *if it satisfies the following conditions:*

1. ***Correct Classification:***

$$\underset{i\in\{1\ldots m\}}{\operatorname{argmax}}\, \boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}(1))_i = \underset{i\in\{1\ldots m\}}{\operatorname{argmax}}\, \boldsymbol{y}_i.$$

2. ***Exponential Stability:*** *$V_{\boldsymbol{y}}(\boldsymbol{x}(t))$ satisfies the condition in Equation* (2.10).

3. ***δ-Final Loss:***

$$V_{\boldsymbol{y}}(\boldsymbol{x}(1)) \leq e^{-\kappa} V_{\boldsymbol{y}}(\boldsymbol{x}(0)) \leq \delta.$$

Definition 5 captures the general properties needed to analyze adversarial robustness. First, the learned model must correctly classify the unperturbed example (implied by minimizing $V_{\boldsymbol{y}}$). Second, the learned dynamics must be exponentially stable w.r.t. $V_{\boldsymbol{y}}$ (implied by minimizing Lyapunov loss). Third, at termination of inference ($t = 1$), the classification loss encoded in $V_{\boldsymbol{y}}$ is within an additive constant $\delta \geq 0$ from optimal (implied by exponential stability).

Theorem 3 (Adversarial Robustness of LyaNet). *Consider an ODE defined in Equations* (2.1) *to* (2.3) *that has δ-stable inference dynamics for input-output pair $(\boldsymbol{o}, \boldsymbol{y})$. Then the system given a perturbed input $\boldsymbol{o} + \boldsymbol{\epsilon}$ with $\|\boldsymbol{\epsilon}\|_\infty \leq \overline{\epsilon}$ will produce a correct classification (Definition 5) so long as:*

$$\delta \leq \log(2) - \frac{L\overline{\epsilon}}{\kappa}\left(1 - e^{-\kappa}\right). \tag{2.15}$$

See Section 2.A.1 for proof. Essentially, if a system is exponentially stable, then under perturbation, the system will be exponentially stable to a relaxed set of states (the radius of the relaxation is proportional to the perturbation magnitude). So long as that relaxation is contained in the part of the state space that outputs the correct classification, the final prediction is also adversarially robust. Another consequence of Theorem 3 is that the guarantee is stronger the more accurate the learned model is, since δ is exactly the nominal cross entropy loss of the prediction.

2.4 Learning Algorithms

We now present two algorithms for (approximately) optimizing Lyapunov loss (Equation (2.13)). The first approach is based on Monte Carlo sampling, and is suitable when the dimension of $\boldsymbol{x}$ is small-to-moderate (e.g., tens to hundreds). The

Algorithm 1 Monte Carlo LyaNet Training

hyperparameters:
  $\mu_H, \mu_{[0,1]}$ ▷ Measures on $\boldsymbol{x}$ and t
  Γ ▷ Number of MC samples
  α, M ▷ Learning rate and max iterations
initialize $\boldsymbol{\theta}$ ▷ Any standard NN initialization
for $i = 1 : M$ **do**
  $(\boldsymbol{o}, \boldsymbol{y}) \sim \mathcal{D}$ ▷ Sample training data
  $\{(\boldsymbol{x}_j, t_j)\}_{j=1}^{\Gamma} \sim \mu_H \times \mu_{[0,1]}$ ▷ Sample Γ $(\boldsymbol{x}, t)$ pairs
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \sum_j \mathscr{V}(\boldsymbol{o}, \boldsymbol{y}, \boldsymbol{x}_j, t_j)$ ▷ From Equation (2.12)
end for
return $\boldsymbol{\theta}$

second approach is based on discretized path integrals, and is suitable when the dimension of $\boldsymbol{x}$ is very large (e.g., hundreds of thousands). The main benefit of the MC approach is that it is easily parallelized (since it does not require solving an ODE during training), but may require too many samples to be practical in high dimensions.

The difficulty in optimizing Lyapunov loss is that the distributions for $\boldsymbol{x}$ and t are coupled, due to the inner integral in Equation (2.13). If one can effectively sample from this joint $(\boldsymbol{x}, t)$ distribution, then one can minimize the (expected) Lyapunov loss by optimizing the parameters $\boldsymbol{\theta}$ w.r.t. the point-wise Lyapunov loss (Equation (2.12)) at those samples.

Restriction on Initial State. To simplify algorithm design, we restrict to learning ODEs that always initialize at $\boldsymbol{x}_0 = \boldsymbol{0}$, i.e., Equation (2.1) is a constant function. The trained ODEs still perform well in practice.

Monte Carlo Training (Algorithm 1). The key idea is to sample $\boldsymbol{x}$ and t independently from measures μ_H and $\mu_{[0,1]}$, which is efficient so long as sampling from these measures is efficient. In practice, we choose uniform for $\mu_{[0,1]}$ and discuss μ_H next. The resulting (implicit) learning objective approximates the Lyapunov loss (Section 2.A.2) so that zero learning objective implies the Lyapunov Loss is zero.

One subtlety in defining μ_H is choosing the support set H of the state space. Earlier, H had been used solely as a theoretical object, but now must be made explicit. The key requirement is that H covers the actual states visited by the ODE during inference. Since the ODE function class is globally Lipschitz, we can bound how far states can evolve from the origin. In practice, we choose μ_H to be a either a uniform on a hypercube or on a hypersphere biased towards the origin, with radius of H

Algorithm 2 Path Integral LyaNet Training

hyperparameters:
  Γ ▷ Time discretization resolution
  α, M ▷ Learning rate and max iterations
initialize $\boldsymbol{\theta}$ ▷ Any standard NN initialization
define $t_0, t_1, \ldots, t_\Gamma$ ▷ Evenly spaced from 0 to 1
for $i = 1 : M$ **do**
  $(\boldsymbol{o}, \boldsymbol{y}) \sim \mathcal{D}$ ▷ Sample training data
  $\boldsymbol{x}(t_j) \leftarrow \int_{t_{j-1}}^{t_j} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{o}, \tau) d\tau + \boldsymbol{x}(t_{j-1}) \quad \forall_{1 \leq j \leq \Gamma}$ ▷ Generate path integral discretization
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \sum_j \mathtt{V}(\boldsymbol{o}, \boldsymbol{y}, \boldsymbol{x}(t_j), t_j)$ ▷ See Equation (2.16)
end for
return $\boldsymbol{\theta}$

being the distance from the initial condition to states corresponding to the corners of the hypercube that achieve losses of $e^{\pm\kappa}$ (Section 2.A.2).

Due to the parallel nature of random sampling, this method often outperforms conventional Neural ODE training w.r.t. compute time, since it avoids the sequential integration steps needed for standard backpropagation.[5] For instance, we found that even for 100-dimensional state spaces, only $\Gamma = 500$ samples were required to reliably approximate the integral, which can be very efficient with modern GPUs.

Path Integral Training (Algorithm 2). For systems with extremely high dimensional state spaces (e.g., hundreds of thousands), or for systems that are better approximated in discrete time, we consider an alternative approach based on collecting roll-outs of the original dynamics.

The basic idea is to approximate the inner integral in the Lyapunov loss using Euler integration, which uniformly discretizes the integration into Γ segments. We can also define a discrete-time version of the point-wise Lyapunov loss:[6]

$$\mathtt{V}(\boldsymbol{o}, \boldsymbol{y}, \boldsymbol{x}, t_j, t_{j-1}) = \max\left\{0, V_{\boldsymbol{y}}(\boldsymbol{x}(t_i)) + (\kappa - 1) V_{\boldsymbol{y}}(\boldsymbol{x}(t_{j-1}))\right\}. \tag{2.16}$$

The resulting discrete-time Lyapunov loss is then:

$$\mathop{\mathbb{E}}_{(\boldsymbol{o},\boldsymbol{y})\sim\mathcal{D}} \left[\sum_{i=1}^{\Gamma} \mathtt{V}(\boldsymbol{o}, \boldsymbol{y}, \boldsymbol{x}, t_i, t_{i-1}) \right]. \tag{2.17}$$

[5] The computational gains remain even after significantly reducing the precision of the ODE solver.

[6] There also exist analogous results to Theorem 1 for discrete-time systems [39], a thorough treatment of which is beyond the scope of this work.

Dataset	Method	Nominal	$\ell_\infty(\epsilon = 8/255)$	$\ell_2(\epsilon = 127/255)$
FashionMNIST	ResNet18	8.17	84.55	33.69
	Continuous	7.67	91.85	45.87
	Continuous LyaNet	**7.18**	**93.96**	**46.54**
	Neural ODE	8.66	77.9	29.85
	LyaNet	**7.9**	**46.33**	**25.14**
CIFAR10	ResNet18	19.73	76.44	40.06
	Continuous	13.12	88.79	41.53
	Continuous LyaNet	**12.99**	**91.03**	**39.82**
	Neural ODE	18.5	75.7	38.91
	LyaNet	**17.22**	**58.97**	**38.25**
CIFAR100	ResNet18	41.07	91.08	83.72
	Continuous	35.96	97.12	80.34
	Continuous LyaNet	**35.81**	**97.77**	**78.68**
	Neural ODE	41.57	92.77	81.7
	LyaNet	**41.07**	**91.08**	**81.91**

Table 2.1: Test error percentage comparison for networks trained with our approach and the equivalent network trained with other methods. Continuous LyaNet uses the Continuous Net architecture, and LyaNet uses the Neural ODE architecture. Adversarial robustness results are trained with PGD. LyaNet trained models have similar performance to their counterparts trained by back-propagating through solutions. We note that LyaNet trained with the Monte Carlo approach has a significant robustness enhancement in low-dimensional datasets.

Optimizing Equation (2.17) has the advantage of being more computationally efficient than Monte Carlo training in high-dimensional systems such as ResNet-inspired Continuous-in-Depth architectures [25]. This is due to the Monte Carlo method placing very little density on the states actually traversed by the ODE, thus requiring many samples to learn reliably. Notice, however, that the underlying inference dynamics remain continuous, we simply apply a discrete-time condition to it. Furthermore, the integral in Algorithm 2 can be replaced with discrete-time dynamics for application in discrete systems.

2.5 Experiments

In our experiments we address the following questions:

- How well do LyaNets perform compared to their baseline counterparts, including under adversarial attacks?
- How quickly can LyaNet inference converge (i.e., does exponential stability hold), thus allowing early inference termination?

- How does the decision boundary of LyaNet compare with that of other methods?
- How does the computational cost of training LyaNets compare with regular backpropagation?
- How does varying the paremter κ affect model generalization?

Experiment Setup

We compare with three model classes:

- ResNet-18 [16], which can be viewed as an Euler integrated dynamical system.
- A Continuous-in-Depth Network, as presented by Queiruga et al. [25], trained with both the adjoint method (labeled Continuous) and Path-Integral LyaNet (labeled Continuous LyaNet).
- A Data-Controlled Neural ODE where ResNet-18 is used to learn a parameterization of the dynamics trained with both the adjoint method (labeled Neural ODE) and Monte Carlo LyaNet (labeled LyaNet). In both cases the number of hidden dimensions correspond to the number of classes.

We evaluate primarily on three computer vision datasets: FashionMNIST, CIFAR-10 and CIFAR-100. More details on the training can be found in Section 2.A.3

Benchmark Experiments

Table 2.1 shows the primary quantitative results for both standard test prediction error as well as prediction error under PGD bounded adversarial attacks. For standard or nominal test error, we see that our LyaNet variants achieve competitive or superior performance compared to their counterparts trained via direct backpropagation.

For our robustness results, LyaNet trained using the Monte Carlo methods exhibits improved robustness for CIFAR-10 and FashionMNIST, despite not being explicitly trained to handle adversarial attacks. CIFAR-100 has a significantly larger hidden state dimensionality, and this larger hidden state also hurt nominal performance. These results are consistent with Theorem 3, which requires a low nominal error (δ) relative to perturbation size (ϵ) to guarantee adversarial robustness.

We finally note that the Path Integral method used to train the continuous-in-depth network was not able to improve adversarial robustness. This may be due to

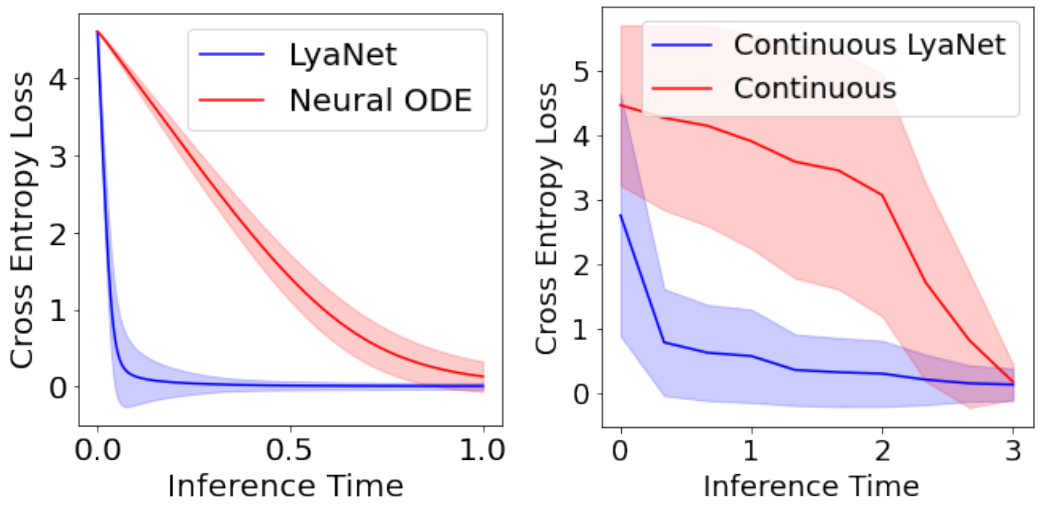


Figure 2.3: Plotting inference time vs prediction loss (if we stopped inference early and made a classification) on 512 correctly classified test examples from CIFAR-100. We see across two model classes that the LyaNet inference dynamics converge much faster.

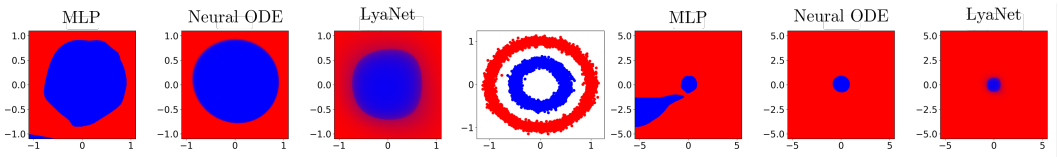


Figure 2.4: **Comparing softmax outputs** of learned dynamical systems. **Center:** depicting the training data with two classes (red and blue) in a 2-D input space. **Left 3 Plots:** comparing the softmax outputs, where we see that MPL and Neural ODE have sharp transitions between the two classes, while LyaNet has much smoother transitions. **Right 3 Plots:** showing a zoomed out version of left 3 plots.

the coarseness of the path integration, which breaks the continuous exponential stability condition needed to guarantee robustness in Theorem 3 (since we only proved exponential stability for continuous-time integration and not discrete-time).

These results suggest that the LyaNet framework offers promising potential to improve the reliability of Neural ODE training, and more gains may be possible with improved training algorithms for optimizing Lyapunov loss.

Inference Convergence Speed

Early inference termination refers to stopping the dynamics before $t = 1$, and output $\hat{\boldsymbol{y}}(t)$. Intuitively, one might expect LyaNets to perform well under early inference termination.

Figure 2.3 shows the results for the CIFAR-100 dataset. We see that both the continuous-in-depth models and the data-controlled models show significantly better performance during early termination when trained with LyaNet compared to standard backpropagation. These results also demonstrate that the internal dynamics of the learned models are fundamentally different when trained with LyaNet versus standard backpropagation.

Inspecting Decision Boundaries

To provide some intuition on the the type of decision boundaries being learned, we run an experiment on a low-dimensional dataset as shown in Figure 2.4. All models were trained with the same number of data-points and tested on a uniform grid. The Neural ODE, MLP and (Monte Carlo) LyaNet have the same number of parameters and the models are trained to similar accuracy. We note that because of the low-dimensional setting, this is a best-case scenario for the LyaNet approach. The MLP exhibits the fragility we expect from conventional neural networks. Although the Neural ODE captures the data well, its decision boundary is very sharp and biased towards the outer circle. Meanwhile, the LyaNet is able to capture the uncertainty between the inner and outer circles, and this uncertain boundary can be interpreted as related to adversarial robustness.

Computation Time

For all models we compared training times for one epoch using the CIFAR-100 dataset. As you can see in Table 2.2, training with LyaNet can cause a marginal increase in training time for both models. This is expected since we have to evaluate the loss at all the evaluation times for the path integral method and over a large number of samples for the Monte Carlo approach. Still, the difference in training time is about 1 minute per epoch for the Monte Carlo approach and less than 10 seconds for Path Integral approach.

Model	Batch Size 64	Batch Size 128
ResNet18	44.29	23.23
Neural ODE	183.82	104.95
LyaNet	151.08	170.64
Continuous	109.41	57.33
Continuous LyaNet	112.98	62.37

Table 2.2: Amount of time in seconds needed to complete an epoch of training for each method on the CIFAR100 dataset.

κ Ablation Study

The parameter κ corresponds to the exponential rate of convergence in Equation (2.9). We trained models with values of κ different to those used in Table 2.1 to observe how changing κ affects the performance of the model. Recall that in the continuous setting (LyaNet) $\kappa > 0$ is the only constraint on the rate of convergence while in the discrete-time setting (Continuous LyaNet) $0 < \kappa < 1$ must be satisfied.

κ	LyaNet	κ	Continuous LyaNet
0.1	43.18	0.9	37.12
1.0	39.83	0.09	39.85
10.0	40.46	0.009	47.73

Table 2.3: Test error percentage for models trained with different exponential convergence parameter κ on the CIFAR-100 dataset.

As you can see in Table 2.3, the LyaNet experiments have a minimum at $\kappa = 1.0$. Recall that the Continuous LyaNet architecture has continuous dynamics but we apply the discrete-time Lyapunov condition. In this case, progressively larger κ appear to improve the models performance. For reference, in Table 2.1 LyaNet is using $\kappa = 5$ and Continuous LyaNet is using $\kappa = 0.999955$.

In the continuous case, κ is a form of regularization for the dynamics of the model. As you can see in the LyaNet experiment, increased stability can come at the cost of expressiveness. This trade-off between expressiveness and stability is expected given Theorem 3(which that implies a larger κ makes the model more robust) and the fact that robustness decreases model expressiveness.

2.6 Related Work

Prior Work in Learning Dynamical Systems. Most prior work has focused on using the adjoint method to infer dynamics [7, 3]. The proposal by E [14] discusses properties like controllability but ultimately frames inference as a standard supervised learning, or optimal control, problem. Although optimal control is a powerful framework, for ODE training it may be challenging to avoid fragile solutions and attain strong stability guarantees.

Prior work have also studied how to impose stability in various forms, including guaranteeing stability around an equilibrium point [21, 5, 4, 32], and designing architectures that stable by construction Haber and Ruthotto [15]. Drawbacks of these approaches are that they do not guarantee exponential stability everywhere in

the state space, or they can create a tension between accuracy and stability by using less expressive models.

Prior Work in Learning and Control. There have been many other intersection points between learning and control theory, although none have studied using control theory to influence the inference procedure of a learned model as we do. Wilson, Recht, and Jordan [37] use Lyapunov theory to analyze the dynamical system implicit in the momentum updates of stochastic gradient descent, Amos et al. [2] and Williams et al. [36] differentiate through controllers such as MPC, and Peng et al. [24] learn control policies directly for a real system. Richards, Berkenkamp, and Krause [28] safely learn physical dynamics by taking into account Lyapunov-like conditons during training. Chang, Roohi, and Gao [6] use an adversarial approach with a similar loss condition to learn controllers using Lyapunov functions. Cheng et al. [9] uses a control prior to regularize reinforcement learning, in some cases with robustness properties of the learned policy. Chow et al. [10] uses a Lyapunov condition to improve safety conditions in reinforcement learning. Dean et al. [12] study the sample complexity of learning controllers such as for LQR. Rosolia and Borrelli [30] study how to learn cost-to-go value functions for MPC under safety constraints. Learning Lyapunov functions for controlling hybrid dynamical systems Chen et al. [8]. Learning Lyapunov stability certificate for a black-box neural network policy Richards, Berkenkamp, and Krause [27]. Neural networks have also been used to learn contraction metrics (a similar concept to Lyapunov) for the control and estimation of stochastic systems Tsukamoto, Chung, and Slotine [35].

2.7 Discussion

This chapter established that pointwise Lyapunov conditions can be used as a training objective for dynamical systems. The key insight is structural: global stability is a trajectory-level property, but it can be certified by checking a pointwise inequality throughout the state space. This local-to-global relationship enables Monte Carlo training that samples states rather than integrating trajectories. The empirical results on image classification demonstrate that the approach works in principle: LyaNet achieves competitive accuracy while conferring adversarial robustness as a consequence of stability, without explicit adversarial training.

The classification setting served as a controlled testbed, but several aspects distinguish it from the robotic control problems that motivate this thesis. In Neural ODEs for classification, the dynamics are fully parameterized and the state space

has no physical meaning. We can design the entire system, which allows relatively simple control tools to suffice. In robotic systems, by contrast, the dynamics are given by physics and we can only influence the system through actuation. The state has physical significance: joint angles, velocities, and contact forces. Underactuation constrains which degrees of freedom can be directly commanded. Contact introduces hybrid dynamics that switch between continuous evolution and discrete impacts. These complications require extensions beyond what LyaNet provides.

The connection between Lyapunov stability and adversarial robustness has a natural interpretation for control. Consider a stabilizing controller for a physical system. If the closed-loop dynamics satisfy a Lyapunov condition, the system converges to an equilibrium. Theorem 3 shows that under bounded perturbations to the dynamics, the system remains in a neighborhood of the nominal trajectory. This is the standard notion of robustness in control: a stable system should tolerate noise, model uncertainty, and disturbances without diverging. The adversarial perturbations studied in classification are analogous to these disturbances. The theoretical guarantee is the same: Lyapunov stability implies bounded deviation under bounded perturbation.

Several limitations of the current approach point toward the extension developed in Chapter 3. First, LyaNet provides stability but not safety. Stability guarantees convergence to an equilibrium or limit set; safety guarantees that trajectories remain within a constraint set. For robotic systems operating near obstacles or joint limits, safety is often the more relevant property. The natural extension is to replace Lyapunov conditions with forward invariance conditions, which certify that trajectories never leave a specified safe set rather than converging to an equilibrium.

Second, the certification in this chapter is empirical rather than formal. We verify that the trained network satisfies the Lyapunov condition on sampled test points, but we do not prove that it holds everywhere. The FI-ODE framework addresses this gap by developing verification tools that certify conditions hold throughout the boundary of the safe set, not just at sampled points. This yields the first Neural ODE controller with certified forward invariance, demonstrated on a planar Segway stabilization task.

Third, the Monte Carlo training approach requires sampling a bounded region of state space. In this chapter, that region was chosen heuristically based on the classification problem structure. For physical systems, the relevant region is determined by the task: the robot's workspace, the range of feasible velocities, or the configurations

reachable from a given initial condition. FI-ODE addresses this through adaptive sampling that focuses training on the boundary of the forward invariant set, which is the region where the safety conditions must hold most stringently.

Appendix 2.A

2.A.1 Proofs

Proof of Lemma 1

Proof. We must show that the truncated cross entropy loss $V_y(\boldsymbol{x}) = \max\{0, \mathcal{L}_{ce}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) - \gamma\}$ satisfies the dynamic projection bounds in Equation (2.7) on the compact state space H.

Step 1: The optimal set X^* is non-empty and compact. Define $X^* = \{\boldsymbol{x} \in H : V_y(\boldsymbol{x}) = 0\} = \{\boldsymbol{x} \in H : \mathcal{L}_{ce}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) \leq \gamma\}$. Since $\mathcal{L}_{ce} \circ \boldsymbol{\psi}_{\boldsymbol{\theta}}$ is continuous and H is compact, the sublevel set X^* is closed. As a closed subset of a compact set, X^* is compact. For $\gamma > 0$, X^* is non-empty provided there exists some $\boldsymbol{x} \in H$ with sufficiently low cross-entropy loss.

Step 2: Upper bound (Lipschitz property). Since V_y is continuous on the compact set H, it is Lipschitz with some constant $L > 0$. For any $\boldsymbol{x} \in H$ and $\boldsymbol{x}^* \in X^*$:

$$V_y(\boldsymbol{x}) = V_y(\boldsymbol{x}) - V_y(\boldsymbol{x}^*) \leq L\|\boldsymbol{x} - \boldsymbol{x}^*\|. \tag{2.18}$$

Taking the infimum over $\boldsymbol{x}^* \in X^*$ yields $V_y(\boldsymbol{x}) \leq L \cdot \text{dist}(\boldsymbol{x}, X^*)$. Thus $\overline{\sigma} = L$.

Step 3: Lower bound (growth away from X^*). For $\boldsymbol{x} \notin X^*$, we have $V_y(\boldsymbol{x}) > 0$ and $\text{dist}(\boldsymbol{x}, X^*) > 0$. On the compact set $H \setminus \text{int}(X^*)$, consider the continuous function:

$$g(\boldsymbol{x}) = \frac{V_y(\boldsymbol{x})}{\text{dist}(\boldsymbol{x}, X^*)}. \tag{2.19}$$

On the boundary ∂X^* where $\mathcal{L}_{ce}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) = \gamma$, the gradient of the (untruncated) cross-entropy loss is non-zero since the unconstrained minimum of cross-entropy lies outside H (at infinity). By the convexity of cross-entropy in the logits and the chain rule, there exists $c > 0$ such that for $\boldsymbol{x}$ near ∂X^*:

$$V_y(\boldsymbol{x}) = \mathcal{L}_{ce}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) - \gamma \geq c \cdot \text{dist}(\boldsymbol{x}, X^*). \tag{2.20}$$

Since $g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in H \setminus X^*$ and H is compact, we can define:

$$\underline{\sigma} = \inf_{\boldsymbol{x} \in H \setminus X^*} g(\boldsymbol{x}) > 0. \tag{2.21}$$

This gives us $V_y(\boldsymbol{x}) \geq \underline{\sigma} \cdot \text{dist}(\boldsymbol{x}, X^*)$ for all $\boldsymbol{x} \in H$.

Combining Steps 2 and 3, V_y satisfies Equation (2.7) and is therefore a dynamic projection. □

Proof of Theorem 2

Proof. We begin by rearranging the terms of the integral:

$$\mathscr{L}(\boldsymbol{\theta}) = E_{(\boldsymbol{o},\boldsymbol{y})\sim D}\left[\int_0^1 \mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}_{\boldsymbol{\theta}}(\tau),\tau)d\mu(\tau)\right] \tag{2.22}$$

$$= \int_D \int_0^1 \mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}_{\boldsymbol{\theta}}(\tau),\tau)d\mu(\tau)dD((\boldsymbol{o},\boldsymbol{y})) \tag{2.23}$$

$$= \int_{D\times[0,1]} \mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}_{\boldsymbol{\theta}}(\tau),\tau)d\mu \times D(\tau,(\boldsymbol{o},\boldsymbol{y})) \tag{2.24}$$

$$\tag{2.25}$$

Recall that

$$\mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x},t) = \max\left\{0, \frac{\partial V_y}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{o},t) + \kappa V_y(\boldsymbol{x})\right\} \tag{2.26}$$

We note that that $\mathscr{V}$ is being integrated over a bounded domain and it satisfies the following properties:

1. $\mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}(t),t) \geq 0$ for all values of $\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}$ and t.
2. $\mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}(t),t)$ is continuous since it is the maximum of two continuous functions: the 0 function and $\frac{\partial V_y}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{o},t) + \kappa V_y(\boldsymbol{x})$ which is continuous since it is differentiable.

From these two facts we can conclude that $\mathscr{L}(\boldsymbol{\theta}) = 0$ implies that for all $t, (\boldsymbol{o},\boldsymbol{y}) \in [0,1]\times D$ the function $\mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x}(t),t) = 0$. This follows from the standard calculus argument that if the function weren't zero at a point there would be an ϵ region surrounding that point that would integrate to a strictly positive value. Since $\mathscr{V} \geq 0$, integrating it over any region must also be non negative which would be a contradiction given that we assumed the integral is non-negative. □

Proof of Theorem 3

Theorem 3 is inspired by concepts underlying Input-to-State and Projection-to-State stability [33]. The key technical detail is leveraging the Comparison Lemma in a similar fashion as Taylor et al. [34].

Proof. We will use the following notation for the time derivative of $V_{\boldsymbol{y}}$:

$$\dot{V}_{\boldsymbol{y}}(\boldsymbol{x}(t), \boldsymbol{o}, t) = \frac{d}{dt} V_{\boldsymbol{y}}(\boldsymbol{x}(t)) = \left.\frac{\partial V_{\boldsymbol{y}}}{\partial \boldsymbol{x}}\right|^{\top}_{\boldsymbol{x}=\boldsymbol{x}(t)} \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{o}, t) \tag{2.27}$$

We now begin the derivation.[7]

$$\dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o} + \boldsymbol{\epsilon}, t) = \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) + \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o} + \boldsymbol{\epsilon}, t) - \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) \tag{2.28}$$

$$\leq \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) + |\dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o} + \boldsymbol{\epsilon}, t) - \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t)| \tag{2.29}$$

$$\leq \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) + \left| \frac{\partial V_{\boldsymbol{y}}}{\partial \boldsymbol{x}}^{\top} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{o} + \boldsymbol{\epsilon}, t) - \frac{\partial V_{\boldsymbol{y}}}{\partial \boldsymbol{x}}^{\top} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{o}, t) \right| \tag{2.30}$$

From the Global Uniform Lipschitz property of V and $\boldsymbol{f}$:

$$\leq \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) + \underbrace{L_V L_f}_{L} \|\boldsymbol{\epsilon}\| \tag{2.31}$$

Using the adversarial disturbance bound $\|\boldsymbol{\epsilon}\| \leq \overline{\epsilon}$:

$$\leq \dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) + L\overline{\epsilon} \tag{2.32}$$

From the Lyapunov Exponential Stability condition in Theorem 1:

$$\leq -\kappa V_{\boldsymbol{y}}(\boldsymbol{\psi}(\boldsymbol{x})) + L\overline{\epsilon} \tag{2.33}$$

We now consider the dynamical system that achieves the upper bound Equation (2.33) in preparation to apply the comparison lemma:

$$\dot{\gamma} = -\kappa\gamma(t) + L\overline{\epsilon} \tag{2.34}$$

This dynamical system is linear and therefore has solutions of the following form:

[7] Note that the choice of norm is arbitrary given equivalence of norms in finite dimensional spaces

$$\gamma(t) = e^{-\kappa t}c + \frac{L\overline{\epsilon}}{\kappa} \tag{2.35}$$

Now we solve for c in terms of the initial condition of this system:

$$\gamma(0) = c + \frac{L\overline{\epsilon}}{\kappa} \tag{2.36}$$

$$\rightarrow c = \gamma(0) - \frac{L\overline{\epsilon}}{\kappa} \tag{2.37}$$

$$\rightarrow \gamma(t) = e^{-\kappa t}\gamma(0) + \frac{L\overline{\epsilon}}{\kappa}(1 - e^{-\kappa t}) \tag{2.38}$$

Where the last line comes from plugging the resulting value of c back into the solution $\gamma(t)$.

To apply the Comparison Lemma, we need a choice of $\gamma(0)$ that satisfies $V_y(\boldsymbol{x}(0)) \leq \gamma(0)$. Recall from the δ-Final Loss condition of Definition 5 that $e^{-\kappa}V_y(\boldsymbol{x}(0)) \leq \delta$. This implies that $V_y(\boldsymbol{x}(0)) \leq \delta e^{\kappa}$ so that we can pick $\gamma(0) = \delta e^{\kappa}$ to satisfy the initial condition inequality of the comparison lemma. Plugging this value into our form of the solution for $\gamma(t)$ results in the following:

$$\gamma(t) = e^{\kappa(1-t)}\delta + \frac{L\overline{\epsilon}}{\kappa}(1 - e^{-\kappa t}) \tag{2.39}$$

Therefore, since the following conditions hold:

1. $\dot{V}_y(\boldsymbol{x}(t))$ and $\dot{\gamma}(\gamma(t))$ are continuous functions of time and state.
2. $\dot{V}_y(\boldsymbol{x}(t)) \leq \dot{\gamma}(\gamma(t))$
3. $V_y(\boldsymbol{x}(0)) \leq \gamma(0)$

We can conclude from the comparison lemma that $V_y(\boldsymbol{x}(t)) \leq \gamma(t)$.

Recall from the definition of V_y as the cross entropy loss, that $V_y(\boldsymbol{x}(t)) \leq -\log(\frac{1}{2}) = \log(2)$ is sufficient for to guarantee Correct Classification as defined in Definition 5. Therefore, it suffices to show that $\gamma(t) \leq \log(2)$. Plugging in the definition of $\gamma(t)$ evaluated at the end of the integration time $t = 1$ we obtain:

$$\delta + \frac{L\overline{\epsilon}}{\kappa}(1 - e^{-\kappa}) \leq \log(2) \tag{2.40}$$

$$\delta \leq \log(2) - \frac{L\overline{\epsilon}}{\kappa}(1 - e^{-\kappa}) \tag{2.41}$$

Which results in the desired bound.

□

2.A.2 Details on Learning Algorithms

Monte Carlo Method. We can formulate the objective of the training optimization problem as a Monte Carlo integral as follows:

$$\min_{\boldsymbol{\theta}\in\Theta} \mathscr{L}(\boldsymbol{\theta}) \approx \min_{\boldsymbol{\theta}\in\Theta} \mathop{\mathbb{E}}_{\substack{(\boldsymbol{o},\boldsymbol{y})\sim\mathcal{D} \\ \boldsymbol{x}\sim\mu_H \\ t\sim\mu_{(0,1)}}} [\mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x},t)] \tag{2.42}$$

$$\approx \min_{\boldsymbol{\theta}\in\Theta} \mathop{\mathbb{E}}_{(\boldsymbol{o},\boldsymbol{y})\sim\mathcal{D}} \left[\int_{H\times(0,1)} \mathscr{V}(\boldsymbol{o},\boldsymbol{y},\boldsymbol{x},t) d\mu_{H\times(0,1)}(\boldsymbol{x},t) \right]. \tag{2.43}$$

The expectation in Equation (2.42) has the property that 0 loss implies the Lyapunov Loss ,$\mathscr{L}$, is also 0 provided the PDF implied by μ is non-zero everywhere in H. This is a straight forward implication from the continuity of $\mathscr{V}$ with respect to $\boldsymbol{x}$.

Some of the stated assumptions require justification. First, we know that perfect classifications using the cross entropy loss function require infinite inputs in one coordinate. Exponential stability to a correct classification under the cross entropy lyapunov dynamics would thus imply exponential growth for the hidden state. To avoid numerical stability issues we bound the state-space where the dynamics can evolve through our choice κ and by noting that most neural network architectures are globally Lipschitz. This implies uniqueness and existence of solutions for all time. When taking into account that we only compute a finite time integral, we can conclude that the image of a bounded set of initial conditions through the dynamics's evolution will remain be bounded. Therefore, we only need to enforce a bounded set of initial condition to have a bounded reachable set. This means our predictions can only have a maximum confidence but with a sufficiently a large bounded region where the dynamics can evolve, this error can become arbitrarily low. In practice, we initialized the dynamics with the constant zero function.

The choice of measure for both $\boldsymbol{x}$ and t has the potential to significantly impact the convergence integral; however, in practice we found that a uniform distribution over the n-cube performed well for most problems and in high-dimensional settings a slightly different distribution can be used to mitigate the curse of dimensionality.

Picking H to sample from:

We will sample from a k-dimensional hypercube with corners at $(\pm s, \pm s, \ldots, \pm s)$. We now have to pick s so that $\boldsymbol{x}$ does not evolve outside of the set H. In principle we could instead select a k-hypersphere but it is well known that most of the volume of the hypersphere is concentrated towards the surface in higher dimensions. In

reality, the dynamics will spend most of the time evolving in the interior since our prediction dynamics are initialized at the origin. Although the hypercube also has most of its volume near the corners, we expect systems to evolve near the corners that correspond to a classification. We note that this choice of H is unlikely to be optimal.

To select and s, begin by considering the bound we are trying to enforce on the dynamics:

$$\dot{V}_{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{o}, t) \leq -kV(\boldsymbol{x}) \tag{2.44}$$

In a comparison lemma style analysis consider the system that achieves the upper bound:

$$\dot{\gamma} = -k\gamma \tag{2.45}$$

We hypothesise that in the case of an incorrect classification, the system evolves as if lower bounded by $\underline{\gamma} = k\underline{\gamma}$. This assumption is based on the supposition that the dynamical system will evolve to a corner of the hypercube corresponding to an admissible but incorrect classification. Therefore we only have to solve for an s that satisfies $V_{\boldsymbol{y}}(\boldsymbol{x}^*) = e^{-k}$ where

$$\boldsymbol{x}_i^* = \begin{cases} s & \text{if } 1 \leq i \leq k \text{ is the correct class.} \\ -s & \text{otherwise} \end{cases} \tag{2.46}$$

Although this choice of H worked in practice, this choice of H is not optimal and could be improved by better exploiting the structure of the dynamics and the classification problem more generally.

Sampling on Hypersphere In high-dimensional spaces sampling in the hypercube is highly inefficient for retrieving samples around the origin. Our models are also all initialized at the origin and should end close to a vertex of the simplex that corresponds to a classification. It's thus imperative to bias sampling towards the origin in higher dimensions. To do this we use the following procedure to generate samples:

Algorithm 3 Biased Sampling on Hypersphere

Require: A hyper sphere of radius r_{max} inscribes the hypercube for H

1: $r \sim U(0, r_{max})$ ▷ sample a radius
2: $\hat{\boldsymbol{x}} \sim \mathcal{N}(0, 1)^k$ ▷ sample k dimensional normally distributed random vector
3: **return** $r \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|_2}$ ▷ Method for generating point on surface of unit sphere Muller [23].

Although this method significantly biases the samples towards zero, we note that that the region near zero is the area that implies the smallest prior for a classification. If we are very far away from the origin we would expect to keep moving in the direction opposite of the origin (since we have a strong prior). The dynamics should learn to exploit this by having relatively simple dynamics in points far away from the origin that get more complex as they approach the point with lowest prior (zero).

2.A.3 Experimental Details

To simplify tuning, we trained our models using Nero [19] with a learning rate of 0.01 with a batch size of 64 for models trained with LyaNet and 128 for models trained with regular backpropagation. We found this by performing a grid search on learning rates and batch sizes over $(0.1, 0.001, 0.001) \times (32, 64, 128)$, validated on a held out set of 10% of training data. All models were trained for a total of 120 epochs. For our adversarial attack we used PGD as implemented by Kim [18] for 10 iterations with a step size $\alpha = \frac{2}{255}$. Our experiments ran on a cluster 6 GPUs: 4 GeForce 1080 GPUs, 1 Titan X and Titan RTX. All experiments were able to run on less than 10GB of VRAM.

References

[1] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[2] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. "Differentiable MPC for End-to-end Planning and Control". In: *CoRR* (2018). arXiv: 1810.13400. URL: http://arxiv.org/abs/1810.13400.

[3] Eric Aislan Antonelo, Eduardo Camponogara, Laio Oriel Seman, Eduardo Rehbein de Souza, Jean P. Jordanou, and Jomi F. Hubner. *Physics-Informed Neural Nets for Control of Dynamical Systems*. 2021. arXiv: 2104.02556 [cs.LG].

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *Deep Equilibrium Models*. 2019. arXiv: 1909.01377 [cs.LG].

[5] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. *Stabilizing Equilibrium Models by Jacobian Regularization*. 2021. arXiv: 2106.14342 [cs.LG].

[6] Ya-Chien Chang, Nima Roohi, and Sicun Gao. "Neural lyapunov control". In: *Advances in neural information processing systems* 32 (2019).

[7] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. "Neural Ordinary Differential Equations". In: *Neural Information Processing Systems (NeurIPS)*. 2019. URL: http://arxiv.org/abs/1806.07366.

[8] Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J Pappas, and Victor M Preciado. "Learning lyapunov functions for hybrid systems". In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 2021, pp. 1–11.

[9] Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. "Control regularization for reduced variance reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1141–1150.

[10] Yinlam Chow, Ofir Nachum, Edgar A Duéñez-Guzmán, and Mohammad Ghavamzadeh. "A Lyapunov-based Approach to Safe Reinforcement Learning". In: *NeurIPS*. 2018.

[11] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1310–1320.

[12] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. "On the sample complexity of the linear quadratic regulator". In: *Foundations of Computational Mathematics* 20.4 (2020), pp. 633–679.

[13] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. *Augmented Neural ODEs*. 2019. arXiv: 1904.01681 [stat.ML].

[14] Weinan E. "A proposal on machine learning via dynamical systems". In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11.

[15] Eldad Haber and Lars Ruthotto. "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (Dec. 2017), p. 014004. DOI: 10.1088/1361-6420/aa9a90. URL: https://doi.org/10.1088/1361-6420/aa9a90.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[17] Patrick Kidger, Ricky T. Q. Chen, and Terry Lyons. *"Hey, that's not an ODE": Faster ODE Adjoints via Seminorms*. 2021. arXiv: 2009.09457 [cs.LG].

[18] Hoki Kim. "Torchattacks: A pytorch repository for adversarial attacks". In: *arXiv preprint arXiv:2010.01950* (2020).

[19] Yang Liu, Jeremy Bernstein, Markus Meister, and Yisong Yue. *Learning by Turning: Neural Architecture Aware Optimisation*. 2021. arXiv: 2102.07227 [cs.NE].

[20] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. *Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations*. 2020. arXiv: 1710.10121 [cs.CV].

[21] Gaurav Manek and J. Zico Kolter. *Learning Stable Deep Dynamics Models*. 2020. arXiv: 2001.06116 [cs.LG].

[22] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. "Dissecting neural odes". In: *arXiv preprint arXiv:2002.08071* (2020).

[23] Mervin E. Muller. "A Note on a Method for Generating Points Uniformly on n-Dimensional Spheres". In: *Commun. ACM* 2.4 (Apr. 1959), pp. 19–20. ISSN: 0001-0782. DOI: 10.1145/377939.377946. URL: https://doi.org/10.1145/377939.377946.

[24] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. *Learning Agile Robotic Locomotion Skills by Imitating Animals*. 2020. arXiv: 2004.00784 [cs.RO].

[25] Alejandro F. Queiruga, N. Benjamin Erichson, Dane Taylor, and Michael W. Mahoney. *Continuous-in-Depth Neural Networks*. 2020. arXiv: 2008.02389 [cs.LG].

[26] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified Defenses against Adversarial Examples". In: *International Conference on Learning Representations*. 2018.

[27] Spencer M Richards, Felix Berkenkamp, and Andreas Krause. "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems". In: *Conference on Robot Learning*. PMLR. 2018, pp. 466–476.

[28] Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. *The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems*. 2018. arXiv: 1808.00924 [cs.SY].

[29] Alexander Robey, Luiz Chamon, George Pappas, Hamed Hassani, and Alejandro Ribeiro. "Adversarial robustness with semi-infinite constrained learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[30] Ugo Rosolia and Francesco Borrelli. "Learning model predictive control for iterative tasks. a data-driven control framework". In: *IEEE Transactions on Automatic Control* 63.7 (2017), pp. 1883–1896.

[31] Lars Ruthotto and Eldad Haber. *Deep Neural Networks Motivated by Partial Differential Equations*. 2018. arXiv: 1804.04272 [cs.LG].

[32] Andreas Schlaginhaufen, Philippe Wenk, Andreas Krause, and Florian Dorfler. "Learning Stable Deep Dynamics Models for Partially Observed or Delayed Dynamical Systems". In: *Advances in Neural Information Processing Systems* 34 (2021).

[33] Eduardo D Sontag and Yuan Wang. "On characterizations of the input-to-state stability property". In: *Systems & Control Letters* 24.5 (1995), pp. 351–359.

[34] Andrew J. Taylor, Victor D. Dorobantu, Meera Krishnamoorthy, Hoang M. Le, Yisong Yue, and Aaron D. Ames. "A Control Lyapunov Perspective on Episodic Learning via Projection to State Stability". In: *2019 IEEE 58th Conference on Decision and Control (CDC)* (Dec. 2019). DOI: 10.1109/cdc40024.2019.9029226. URL: http://dx.doi.org/10.1109/CDC40024.2019.9029226.

[35] Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jacques E Slotine. "Neural stochastic contraction metrics for learning-based control and estimation". In: *IEEE Control Systems Letters* 5.5 (2020), pp. 1825–1830.

[36] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. "Information theoretic MPC for model-based reinforcement learning". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1714–1721. DOI: 10.1109/ICRA.2017.7989202.

[37] Ashia C. Wilson, Benjamin Recht, and Michael I. Jordan. *A Lyapunov Analysis of Momentum Methods in Optimization*. 2018. arXiv: 1611.02635 [math.OC].

[38] Eric Wong and Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5286–5295.

[39] Wei Zhang, Alessandro Abate, Jianghai Hu, and Michael P Vitus. "Exponential stabilization of discrete-time switched linear systems". In: *Automatica* 45.11 (2009), pp. 2526–2536.

Chapter 3

CERTIFIABLY ROBUST FORWARD INVARIANCE IN NEURAL ODES

3.1 Introduction

The previous chapter established that Lyapunov conditions can be used to train Neural ODEs with stable inference dynamics. Training on pointwise conditions produced networks whose trajectories converge toward correct predictions, with adversarial robustness emerging as a consequence of stability. That work demonstrated the viability of pointwise-condition training but left open the question of certification: given a trained network, can one prove that the Lyapunov condition holds everywhere in the relevant region, rather than merely at sampled points?

This chapter addresses certification by developing FI-ODE, a framework for training Neural ODEs that are provably forward invariant. Forward invariance is a property from control theory that guarantees trajectories of a dynamical system never leave a specified set [5, 34]. In control applications, forward invariant sets correspond to safe operating regions; in classification, they correspond to regions of state space that produce correct predictions. The key insight is that forward invariance, like Lyapunov stability, reduces to a pointwise condition. One need not analyze entire trajectories to certify that a set is forward invariant. Instead, one verifies that the dynamics point inward at every point on the boundary of the set. This pointwise structure enables both efficient training and tractable certification.

The certification challenge motivates the technical contributions of this chapter. Training a Neural ODE to approximately satisfy a pointwise condition is straightforward: sample states, evaluate the condition, and penalize violations. Certifying that the condition holds everywhere is harder. Sampling can miss small regions where the condition fails, and even a small violation can compromise the global guarantee. We address this through a combination of techniques. First, we constrain the hidden states of the NODE to evolve on a compact set by projecting the dynamics to satisfy barrier conditions. Second, we develop an interval propagation technique compatible with optimization layers that provides sound overapproximations of the dynamics. Together, these allow us to verify forward invariance with formal guarantees.

Contributions. The main contribution of this chapter is the first Neural ODE controller with certified robust forward invariance. We demonstrate this on a planar segway, a canonical unstable nonlinear system where maintaining balance requires active control. The certified guarantee states that the controller keeps the system within a verified region of attraction despite bounded perturbations to the system parameters. Prior work on Neural ODE control [29, 9, 36] demonstrated forward invariant policies empirically but without formal certificates. To our knowledge, FI-ODE produces the first such certificate for a Neural ODE controller.

To validate that the framework generalizes beyond control, we also evaluate on image classification. This setting provides a well-benchmarked domain where certified robustness methods can be compared systematically [51, 42, 13]. Even impressive empirical robustness often fails under unforeseen stronger attacks [7], which motivates the pursuit of formal certificates. The image classification experiments demonstrate superior ℓ_2 certified robustness versus other certifiably robust ODE-based models. The classification results serve primarily to validate the methodology; the control experiments are the primary contribution toward the thesis goal of certified controllers for robotic systems.

Several limitations of this chapter are addressed in subsequent chapters. The planar segway is a relatively simple system: it is low-dimensional and fully actuated in the sense that control authority exists over all degrees of freedom relevant to stability. The certification techniques developed here apply to systems where the dynamics are smooth and the state space is bounded. Extending these methods to underactuated systems with hybrid dynamics, such as legged robots with intermittent ground contact, requires additional theoretical machinery. Chapter 6 develops Zero Dynamics Policies to address these challenges, using the pointwise-condition framework established here as a foundation.

Our code is available at `https://github.com/yjhuangcd/FI-ODE.git`.

3.2 Preliminaries

Neural ODEs. We consider the following Neural ODE (NODE) model class, where $\boldsymbol{o}$ are the inputs to the dynamics, and $\boldsymbol{x} \in \mathcal{H} \subset \mathbb{R}^n$ are the states of the NODE ($\mathcal{H}$ is compact and connected). Let $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^l$ denote the parameters of the learned model. In general, we assume the overparameterized setting, where $\boldsymbol{\theta}$ is expressive

enough to fit the dynamics.

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad \text{(initial condition)} \tag{3.1a}$$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}(t), \boldsymbol{o}) \quad \text{(continuum of hidden layers).} \tag{3.1b}$$

An important setting for NODEs is continuous control, where we can explicitly compose the known dynamics of the physical system with a neural network controller parameterized by $\boldsymbol{\theta}$. The closed-loop system is denoted by $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}(t), \boldsymbol{o})$, where $\boldsymbol{\theta}$ is the neural controller and $\boldsymbol{o}$ are the system parameters. Other settings include image classification, where $\boldsymbol{o}$ are the images, and we evolve the system over $t \in [0, T]$ to get the final prediction $\boldsymbol{x}(T)$.[1]

Forward Invariance & Robust Forward Invariance. Forward Invariance refers to sets of states of a dynamical system (e.g., Equation (3.1b)) where the system can enter but never leave. Formally:

Definition 6 (Forward Invariance). *A set $\mathcal{S} \subseteq \mathcal{H}$ is forward invariant with respect to the system (Equation* (3.1b)*) if $\boldsymbol{x}(t) \in \mathcal{S} \Rightarrow \boldsymbol{x}(t') \in \mathcal{S}, \forall t' \geq t$.*

Forward invariance can be applied generally in NODEs: we can choose the dynamics in Equation (3.1b) to render almost any set we choose forward invariant. For instance, in control we often want to keep the states of the system within a safe set, while in classification we will be concerned with the set of states that produce a correct classification. Mathematically, these settings can be captured by shaping the ODE dynamics $\boldsymbol{f}_{\boldsymbol{\theta}}$ to achieve forward invariance within a specified set (i.e., training the ODE to satisfy Definition 6 for some specified set $\mathcal{S}$).

Definition 7 (Robust Forward Invariance). *A set $\mathcal{S} \subseteq \mathcal{H}$ is robust forward invariant with respect to $\boldsymbol{o}$ if $\mathcal{S}$ is forward invariant with respect to the system $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}(t), \boldsymbol{o} + \boldsymbol{\epsilon})$, $\forall \boldsymbol{\epsilon} \in \mathbb{R}^n$ with $\|\boldsymbol{\epsilon}\| \leq \overline{\epsilon}$.*

Robust forward invariance is attractive when one seeks performance guarantees under input perturbations. Here, we consider norm-bounded perturbations. In control, when there are mis-specifications for system parameters, we still hope the controller to be able to keep the system safe. In classification, we would want the system to classify correctly despite noisy inputs.

[1] In this setting, one can think of a NODE as a ResNet [26] with a continuum of hidden layers.

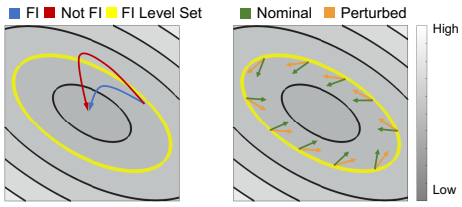


Figure 3.1: Depicting trajectories (Left) and dynamics (Right) of the state-space of a NODE. The contours show a quadratic potential, and the yellow-line is the target sublevel set. Left: trajectories that violate (red) or satisfy (blue) forward invariance. Right: flow field (dynamics) of the NODE, under both nominal and perturbed inputs. The perturbed flow field still satisfies forward invariance, implying robust forward invariance.

Trajectory-wise versus Point-wise Certification Analysis. Figure 3.1 depicts two ways of certifying forward invariance. On the left, we consider entire trajectories that result from running the ODE (i.e., running the forward pass) and determine forward invariance by checking whether the *trajectories* leave the target set. Such trajectory-level analyses are computationally expensive due to running ODE integration to generate trajectories. This approach also poses a challenge for verification since trajectories can only be integrated for finite time T and the dynamics may be close to leaving the set shortly thereafter ($T + \varepsilon$), which translates into vulnerability to perturbations.

An alternative approach, depicted on Figure 3.1(right), relies on point-wise conditions: we look at the dynamics *point-wise* over the state-space and infer whether the set within the yellow line is forward invariant. Here, robust certification can be significantly easier because we only need to verify that the perturbed dynamics are point-wise still pointing in the right direction, rather than analyzing the perturbed dynamics over an entire trajectory (i.e., we do not need to do ODE integration).

Lyapunov Functions & Sublevel Sets.

As discussed further in Section 3.3, we use Lyapunov potential functions from control theory to define sets to render forward invariant. A potential function $V : \mathcal{H} \to \mathbb{R}_{\geq 0}$ is a Lyapunov function for the ODE if for all reachable states $\boldsymbol{x}$ we have:

$$\dot{V} \equiv \frac{d}{dt} V(\boldsymbol{x}(t)) \equiv \frac{\partial V}{\partial \boldsymbol{x}}^{\top} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) \leq 0. \tag{3.2}$$

Intuitively, Equation (3.2) means that the dynamics of the ODE within the states $\mathcal{H}$ are always flowing in the direction that reduces V, i.e., Equation (3.2) establishes a contraction condition on ODE. Note that for the system to be strictly contracting, the RHS of Equation (3.2) needs to be strictly negative.

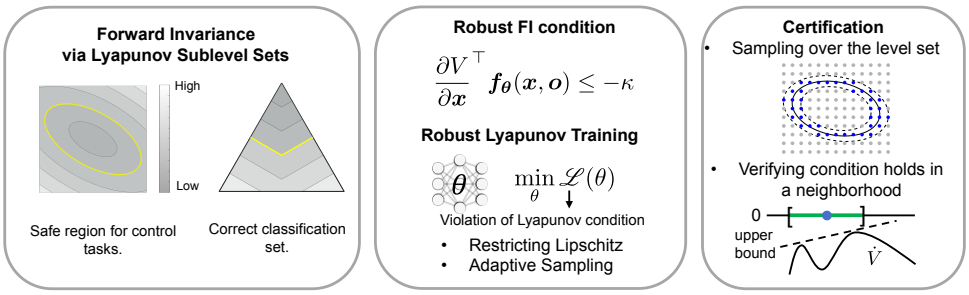


Figure 3.2: Overview of our FI-ODE framework. We first pick a Lyapunov function based on the shape of the forward invariant set: the boundaries of the Lyapunov sublevel sets are parallel to the boundary of the forward invariant set. Then we show that robust forward invariance implies robust control and classification. We train the dynamics to satisfy robust FI conditions via robust Lyapunov training. To certify the forward invariance property, we sample points on the boundary of the forward invariant set and verify conditions hold everywhere on the boundary.

Since V is always decreasing in time, we can use it to define a forward invariant set (Definition 6): $V(\boldsymbol{x}) \leq c$ for some constant c, which is known as a Lyapunov sublevel set. Once a state enters a Lyapunov sublevel set it remains there for all time. The potential function depicted in Figure 3.1 can be viewed as a Lyapunov function, and the yellow line the boundary of the corresponding sublevel set. At training time, one would specify a desired forward invariance condition using a potential function V and threshold c, and optimize the NODE to satisfy the forward invariance condition.

3.3 FI-ODE: Robust Forward Invariance for Neural ODEs

We now present our FI-ODE framework to enforce forward invariance on NODEs (Figure 3.2). We define forward invariance using Lyapunov sublevel sets (Section 3.3), and show that robust forward invariance implies robust control and classification (Section 3.3) To enforce forward invariance, we first train to encourage the Lyapunov conditions to hold on the boundary of the target Lyapunov sublevel set (e.g., yellow line in Figure 3.1), and then verify. We develop a robust Lyapunov training algorithm (Section 3.3) that extends the LyaNet framework [28] to enable efficiently training NODEs that provably satisfy forward invariance. Finally, we develop certification tools to verify the Lyapunov conditions everywhere in the region of interest (Section 3.3).

Forward Invariance via Lyapunov Sublevel Sets

We first define the set that we would like to render forward invariant, and then choose a Lyapunov function whose level sets are parallel to the boundary of the set.[2] For our main application in control, we define the forward invariant set to be a region around an equilibrium point (i.e., straying far from the equilibrium point can be unsafe), and use the standard quadratic Lyapunov function:

$$V(\boldsymbol{x}) = \boldsymbol{x}^\top P \boldsymbol{x}, \tag{3.3}$$

where P is a (learnable) positive definite matrix, and assuming WLOG that the equilibrium point is at the origin. The forward invariant set has the form $\mathcal{S} = \{\boldsymbol{x} | V(\boldsymbol{x}) \leq c\}$, for $c > 0$. The level sets of this quadratic Lyapunov function are shown in Figure 3.1. The boundary is then $\mathcal{D} = \partial\mathcal{S} = \{\boldsymbol{x} | V(\boldsymbol{x}) = c\}$. This forward invariance condition is commonly used in safety-critical control [6].

We also explore an application to multi-class classification. Here, we define the forward invariant set to be the correct classification region. For an input $\boldsymbol{o}$ with label y, the output of a NODE after integrating for T time is $\boldsymbol{x}(T)$. The NODE correctly classifies $\boldsymbol{o}$ if $y = \text{argmax}\, \boldsymbol{x}(T)$. Then the correct classification region for class y is $\mathcal{S}_y = \{\boldsymbol{x} | \boldsymbol{x} \in \triangle, y = \text{argmax}\, \boldsymbol{x}\}$ (Figure 3.2, left panel) where $\triangle$ stands for the n-class probability simplex: $\{\boldsymbol{x} \in \mathbb{R}^n | \sum_{i=1}^n \boldsymbol{x}_i = 1, \boldsymbol{x}_i \geq 0\}$. The boundary of this set is known to be the decision boundary for class y: $\mathcal{D}_y = \{\boldsymbol{x} \in \triangle | \boldsymbol{x}_y = \max_{i \neq y} \boldsymbol{x}_i\}$. We define a Lyapunov function whose level sets are parallel to the decision boundary:

$$V_y(\boldsymbol{x}) = 1 - (\boldsymbol{x}_y - \max_{i \neq y} \boldsymbol{x}_i) \tag{3.4}$$

We can check that V_y is positive definite: since $0 \leq \boldsymbol{x}_i \leq 1$ for all i, we have $V_y \geq 0$. In addition, $V_y = 0$ only when $\boldsymbol{x}_y = 1$ and $\boldsymbol{x}_i = 0$ for $i \neq y$. For the simplicity of notations, we use V to refer to the Lyapunov function, but note that the Lyapunov function for classification depends on class y.

Robust Forward Invariance for Robust Control and Classification

A NODE satisfies robust forward invariance if the forward invariance condition holds despite (norm-bounded) perturbations on the dynamics (e.g. due to perturbed inputs). Our framework uses $\boldsymbol{o}$ for *system parameters* in control, and for *input*

[2] Usually, Lyapunov stability uses a potential function to prove the stability of a given dynamical system. In our setting, the potential function is pre-defined to be positive definite, and we find a dynamical system (e.g. by training a Neural ODE) that is stable with respect to this potential function (i.e. making this potential function a Lyapunov function). This is possible because the NODEs are typically overparameterized.

images in classification. To ensure robust forward invariance for perturbed $\boldsymbol{o}$, the dynamics for the perturbed input $\boldsymbol{f_\theta}(\boldsymbol{x}, \boldsymbol{o} + \boldsymbol{\epsilon})$ needs to satisfy the standard forward invariance condition in (3.2) (as informally depicted in Figure 3.1, right). In other words, the condition in (3.2) needs to hold in a neighborhood of $\boldsymbol{o}$ for robust control or classification. Thanks to the Lipschitz continuity of the Lyapunov function V and the dynamics $\boldsymbol{f_\theta}$, this can be achieved by a more strict condition than (3.2) on the dynamics (Theorem 4).

Theorem 4 (Robust Forward Invariance). *Consider the dynamical system in Equations* (3.1a) *and* (3.1b)*, the set $\mathcal{S}$ will be robust forward invariant with respect to $\boldsymbol{o}$ if the following conditions hold:*

$$\frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f_\theta}(\boldsymbol{x}, \boldsymbol{o}) \leq -\bar{\epsilon} L_V L_f^o, \quad \forall \boldsymbol{x} \in \partial \mathcal{S}. \tag{3.5}$$

where $\bar{\epsilon}$ is the perturbation magnitude on $\boldsymbol{o}$ (i.e. $\|\boldsymbol{\epsilon}\| \leq \bar{\epsilon}$), $\partial\mathcal{S}$ is the boundary of $\mathcal{S}$, L_V is the Lipschitz constant of V and L_f^o is the Lipschitz constant of the dynamics with respect to $\boldsymbol{o}$.

Remark 1 (Implications). With $\mathcal{S}$ defined as in Section 3.3, if the dynamics satisfy (3.5), then we have a robust controller that always keeps the system in the desired region despite perturbed system parameters and inputs.

Remark 2 (Non-Robust Variant). The non-robust version of Theorem 4 is where the RHS of Equation (3.5) is 0 instead of $-\bar{\epsilon} L_V L_f^o$. I.e., Equation (3.5) need not be a strictly contracting condition.

Robust Lyapunov Training

We now present our robust Lyapunov training approach to satisfy the conditions in Theorem 4 (Algorithm 4). Our method extends the LyaNet framework [28] in two ways: 1) restricting the Lipschitz constant of the NODE with respect to the input; and 2) adaptive sampling to focus learning on the states necessary for forward invariance certification.

Training loss.

Our training loss encourages the dynamics to satisfy the conditions in Theorem 4. Specifically, we use a modified Monte Carlo Lyapunov loss from [28]:

$$\mathscr{L}(\boldsymbol{\theta}) \approx \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mu(\mathcal{H})} \left[\max \left\{ 0, \frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f_\theta}(\boldsymbol{x}, \boldsymbol{o}) + \kappa(V(\boldsymbol{x})) \right\} \right], \tag{3.6}$$

Algorithm 4 Robust Lyapunov Training

Require: Lyapunov function V, Sampling scheduler, dataset $\mathcal{D}$, hinge-like function κ.

initialize Model parameters $\boldsymbol{\theta}$, Lyapunov parameters P and system parameters $\boldsymbol{o}$ (for control).
for $i = 1 : M$ **do**
    Sample $\boldsymbol{x}$ based on the training progress and the level sets of V: $\boldsymbol{x} \sim$ `Sampling_scheduler` (i, V)
    For control, find adversarial samples of $\boldsymbol{o}$ and $\boldsymbol{x}$
    For classification, sample $(\boldsymbol{x}, y) \sim \mathcal{D}$
    Update model parameters to minimize Lyapunov loss $\mathscr{L}(\boldsymbol{\theta})$ (Equation (3.6)).
    $P \leftarrow P - \beta' \nabla_P \mathscr{L}(\boldsymbol{\theta})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta})$
end for
return $\boldsymbol{\theta}$

which can be interpreted as a hinge-like loss on the Lyapunov contraction condition for each state $\boldsymbol{x}$ of the NODE (i.e., the loss encourages $(\partial V/\partial \boldsymbol{x})^\top \boldsymbol{f} \leq -\kappa(V(\boldsymbol{x})) < 0$ for some non-negative non-decreasing function κ). Intuitively, if the loss in Equation (3.6) is 0 for some given $\boldsymbol{o}$, then we know that the contraction condition is satisfied with the RHS being $\kappa(V(\boldsymbol{x}))$. As long as $\kappa(V(\boldsymbol{x})) \geq \bar{\epsilon} L_V L_f^o$, then Theorem 4 holds. If instead $\kappa(V(\boldsymbol{x})) \geq 0$, then the non-robust variant holds.

Restricting the Lipschitz constant.

To obtain a non-vacuous guarantee from Theorem 4, we need to restrict the Lipschitz of $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o})$ with respect to both $\boldsymbol{x}$ and $\boldsymbol{o}$. For image classification, we can estimate L_f^o easily by the product of matrix norms of the weight matrices in the neural network [47]. Then we certify condition Equation (3.5) holds for the clean image $\boldsymbol{o}$. For control problems, since the dynamics of the physical system is usually known, the closed loop dynamics is not purely parameterized by neural networks and it is not straightforward to estimate L_f^o. Therefore, instead of directly certifying condition Equation (3.5), we certify the LHS of it to be smaller than 0 for $\boldsymbol{x} \in \partial \mathcal{S}$ and all $\boldsymbol{o}$ within the perturbation range (not only on the nominal parameter). We use adversarial training to make $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o})$ smooth with respect to both $\boldsymbol{x}$ and $\boldsymbol{o}$, and in fact certifiable.

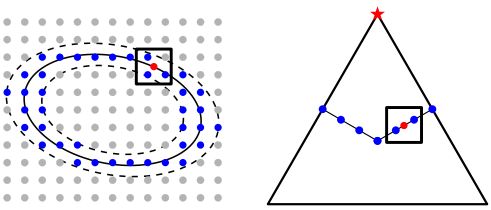


Figure 3.3: Sampling to cover the level sets of the Lyapunov functions.

Adaptive sampling.

To minimize the Lyapunov loss (Equation (3.6)), we need to choose a sampling distribution μ. A simple choice is uniform (as was done in [28]), but that may require an intractable number of samples to guarantee minimizing Equation (3.6) everywhere in the state space. We address this challenge with an adaptive sampling strategy that focuses training samples on the region of the state space necessary for forward invariance: the boundary of the Lyapunov sub-level set. For control problems, since we jointly learn matrix P in the Lyapunov function, the shape of its level set changes during training, and the sampled points change accordingly. For classification, we switch from uniform sampling in the simplex to sampling only within the forward invariant set, with the switching time being a hyper-parameter of the sampling scheduler.

Certification

In the previous section, we minimize the empirical Lyapunov loss (Equation (3.6)) on some finite set of samples to encourage the dynamics to satisfy conditions in Theorem 4. However, zero empirical Lyapunov loss on a finite sample is not necessarily a certificate that the conditions hold everywhere on the boundary of the forward invariant. This section develops tools to certify the forward invariance conditions hold *everywhere* on the boundary of the safe set.

Certification procedures.

The certification procedures are as follows: 1) sample points on the boundary of the forward invariant set (blue dots in Figure 3.3), and check Lyapunov condition holds on all the sampled points; 2) verify the condition holds in a small neighborhood around those points.

Procedure 1: Sampling techniques.

Rigorous certification is challenging because it requires the set of samples and their neighborhoods to cover the whole boundary of the forward invariant set (not guaranteed by random sampling). We construct a set for the quadratic Lyapunov function (Equation (3.3)) and the classification Lyapunov function (Equation (3.4)) respectively, and show that the proposed set can cover the level set of the corresponding Lyapunov function in Theorem 5 below, i.e. for any point on the level set (red dot in Figure 3.3), there exists a sampled point nearby (blue dot).

To sample on the level set of the quadratic Lyapunov function $\mathcal{D} = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x}^\top P \boldsymbol{x} = c\}$, we first create a uniform grid $\mathcal{G}$ (with spacing r) in the ambient space that covers the Lyapunov level set. We pick r to be at most $\sqrt{\frac{c}{\lambda_1}}$, where λ_1 is the maximum eigenvalue of P. Then we do rejection sampling to keep the points that are close to the Lyapunov level set via $\mathcal{G} \cap \mathcal{B}$ for $\mathcal{B}$ defined below:

$$\mathcal{B} = \{\boldsymbol{x} | \underline{c} \leq \boldsymbol{x}^\top P \boldsymbol{x} \leq \overline{c}\} \tag{3.7}$$

where $\underline{c} = (\sqrt{c} - \frac{\sqrt{n}}{2} r \sqrt{\lambda_1})^2$ and $\overline{c} = (\sqrt{c} + \frac{\sqrt{n}}{2} r \sqrt{\lambda_1})^2$. We show that $\mathcal{B} \cap \mathcal{G}$ (points inside dashed lines in Figure 3.3, Left) covers the c-level set of the quadratic Lyapunov function (Theorem 5 (a)).

To sample on the 1-level set of the n-class classification Lyapunov function (the decision boundary), we consider the following set $\tilde{S}_y$ (Figure 3.3, Right) (Theorem 5 (b)):

$$\tilde{S}_y = \{\tilde{\boldsymbol{s}} \in \mathbb{R}^n | \tilde{\boldsymbol{s}} = \frac{\boldsymbol{s}}{N}, \boldsymbol{s} \in S_y\} \tag{3.8}$$

where $S_y = \{\boldsymbol{s} \in \mathbb{Z}^n | \sum_{i=1}^n \boldsymbol{s}_i = N, \boldsymbol{s}_y = \max_{i \neq y} \boldsymbol{s}_i, \boldsymbol{s}_i \geq 0, \forall i = 1, ..., n\}$, and N represents sample density, and needs to be a positive even integer and $N \not\equiv 1 \pmod{n}$.

Theorem 5. (Sampling on the boundary of a FI set).

- **(a)** For any $\boldsymbol{x} \in \mathcal{D}$, there exist an $\boldsymbol{s} \in \{\mathcal{B} \cap \mathcal{G}\}$ such that $|\boldsymbol{x}_i - \boldsymbol{s}_i| \leq \frac{r}{2}$ for all $i = 1, ..., n$.
- **(b)** For any $\boldsymbol{x} \in \mathcal{D}_y$, there exists an $\tilde{\boldsymbol{s}} \in \tilde{S}_y$ such that $|\boldsymbol{x}_i - \tilde{\boldsymbol{s}}_i| \leq \frac{1}{N}$ for all $i = 1, ..., n$.

Procedure 2: Verification in a neighborhood around the sampled points.

Since we only sample a finite number of points on the level set, certifying robust forward invariance requires verifying that the condition holds in a small neighborhood

around each of the points. We do so by bounding the range of the output given the range of the input. This bound can be obtained by estimating the Lipschitz constant of the LHS of Equation (3.5), and the norm of output difference can be bounded by the norm of the input difference. This is convenient for cases where it is simple to bound the Lipschitz of the Lyapunov function and the dynamics. For instance, for classification problems, the Lipschitz constant of the Lyapunov function (3.4) is $\sqrt{2}$, and the Lipschitz constant of the dynamics with respect to both $\boldsymbol{o}$ and $\boldsymbol{x}$ is 1 because we use orthogonal layers in the neural network. For more general Lyapunov functions and dynamics, the Lipschitz bound is often either intractable or vacuous. Instead, we use a popular linear relaxation based verifier CROWN [57] to bound the output of any general computation graph. In the control case, we verify both $\boldsymbol{o}$ and $\boldsymbol{x}$ since we want the robustness robustness for a set of perturbations over a set in the state-space.

3.4 Experiments

Our main evaluation is in an application of certified robust forward invariance in nonlinear continuous control (Section 3.4). We also explore the generality of our approach by studying a second application in certified robustness for image classification (Section 3.4).

Certifying Safety for Robust Continuous Control

Setup.

We evaluate our framework on a planar segway system, which is a highly unstable nonlinear system whose dynamics is sensitive to its system parameters and therefore hard to train certifiably robust nonlinear controllers (see Section 3.A.9 for the details). We train a neural network controller (a 3-layer multi-layer perceptron (MLP)) to keep the system forward invariant within the 0.15-sublevel set of a jointly learned Lyapunov function under ±2% perturbations on each system parameter. This guarantees that the segway will not fall under adversarial system perturbations. We evaluate the its performance under both nominal and adversarial system parameters for 1000 adversarially selected initial states within the safe set. The adversarial parameters and states are optimized jointly via projected gradient descent for 100 steps to maximize violations of the forward invariance condition. We also provide provable certificates using certification approach in Section 3.3.

Results.

We compare with two competitive robust control baselines, and perform ablation studies on different training algorithms in Table 3.1. We make three main observations. First, it is difficult to certify even non-robust forward invariance using previous methods, highlighting the need for more advanced methods. Second, our robust Lyapunov training approach (Algorithm 4) is able to train NODE controllers with robust forward invariance certificates. Third, certifying non-robust forward invariance is easier than certifying robust forward invariance. Overall, these results suggest that our approach is able to train nonlinear ODE controllers in non-trivial settings where existing approaches cannot. To our knowledge, this is also the first instance of training NODE policies with such non-vacuous certified guarantees.

Table 3.1: Robustness of controllers trained with different methods. The numbers are the percentage of trajectories that stay within the forward invariant set under the nominal and adversarial system parameters on 1000 adversarially selected initial states. The certificate column indicates whether the (robust) FI property is certified.

Method	**Empirical**		**Certificate**	
	Nominal	**Adv**	**FI**	**Robust FI**
Robust LQR	96.2	92.8	✗	✗
Robust MBP [18]	94.3	93.6	✗	✗
Standard Backprop Training	58.0	50.4	✗	✗
Basic Lyapunov Training [28]	90.2	52.6	✗	✗
+ Adaptive Sampling	100	68.9	✓	✗
+ Adversarial Training	100	97.8	✓	✗
+ Both (Robust FI-ODE, Ours)	100	100	✓	✓

Visualizations.

We show trajectories that start within the safe set (gray ellipse) and the corresponding Lyapunov functions in Figure 3.4. The left shows the trajectories of a certifiably *non-robust* FI controller. While the system is safe 100% under nominal system parameters, it fails for adversarial system parameters. The right shows the trajectories of a certifiably *robust* FI controller. Even under adversarial system parameters, it keeps the trajectories within the safe set 100% of the time.

Certified Robustness for Image Classification

We also apply our approach to train certifiably robust NODEs for image classification to explore the generality of the framework. We treat the input image as system

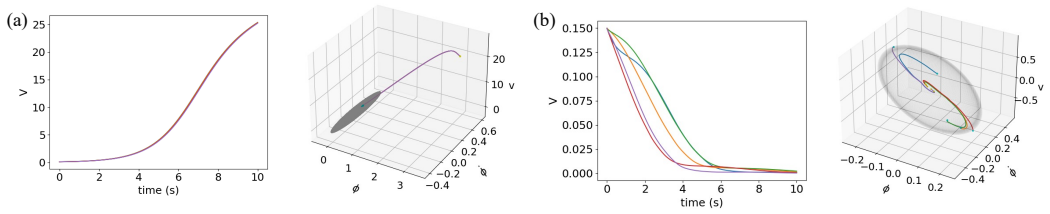


Figure 3.4: Showing Lyapunov function value V along the trajectories of a planar segway. The forward invariant set is the 0.15-sublevel set. All trajectories start within the forward invariant set (gray ellipse). Each system parameter are perturbed adversarially within ±2% of their original value. (a) Shows the Lyapunov function values and system trajectories of a certifiably *non-robust* FI controller. (b) Shows the Lyapunov function values and system trajectories of a certifiably *robust* FI controller.

parameters, and set all the initial states to be $\boldsymbol{x}(0) = \mathbb{1}\frac{1}{n}$. Table 3.2 shows the results. The main metric is certified accuracy, the percentage of test set inputs that are certifiably robust. Our approach achieves the strongest overall certified robustness results compared to prior ODE-based approaches. We also reported clean and adversarial accuracy for references. In addition, the ablation studies shows similar trends as in the robust control experiments: all the learning components: Lyapunov training, Lipschitz restriction and adaptive sampling are needed for good performance, and the model that is trained with all of them (Robust FI-ODE) achieved the highest certified accuracy.

Table 3.2: Evaluating certified robustness for image classification. ϵ is the ℓ_2 norm of the input perturbations. We report the classification accuracy (%) on clean & adversarial inputs, and the percentage of inputs that are certifiably robust (Certified). Semi-MonDeq results are on 100 test images [95% CI in bracket] due to high cost, and other results are on all test images (10,000).

Dataset	**Method**	ϵ	**Clean**	**Adversarial**	**Certified**
MNIST	Lipschitz-MonDeq [41]	0.1	95.60	94.42	83.09
	Semi-MonDeq [12] †	0.1	99 [>94]	99 [>94]	99 [>94]
	Robust FI-ODE(Ours)	0.1	99.35	99.09	95.75
	Lipschitz-MonDeq [41]	0.2	95.60	93.09	50.56
	Robust FI-ODE(Ours)	0.2	99.35	98.83	81.65
CIFAR-10	Lipschitz-MonDeq [41]	0.141	66.66	50.51	<7.37
	NODE w/o Lyapunov training	0.141	69.05	56.94	16.81
	LyaNet [28] + Lipschitz restriction	0.141	73.15	64.87	41.43
	LyaNet [28] + Sampling scheduler	0.141	82.83	74.81	0
	Robust FI-ODE(Ours)	0.141	78.34	67.45	42.27

3.5 Related Works

Robust control and robust learning-based control methods. Robust control involves creating feedback controllers for dynamic systems that sustain performance under adverse conditions [59, 8], often relying on basic (linear) controllers. Our work learns *nonlinear* controllers parameterized by neural networks, while maintaining the robust forward invariance guarantees. There have been recent works for learning-based control with robustness guarantees, such as focusing on $\mathcal{H}_\infty$ robust control [1, 38, 21, 25, 58], or linear differential inclusions systems [18]. In comparison, our framework could be used for general nonlinear systems and norm-bounded input/system parameter perturbations.

Learning Lyapunov functions and controllers for nonlinear control problems. Various studies focus on learning neural network Lyapunov functions, barrier functions, and contraction metrics for nonlinear control [16]. For stability or safety certification, Chang, Roohi, and Gao [11] employ SMT solvers [22], Jin et al. [30] use Lipschitz methods, and Dai et al. [15] apply mixed integer programming. Our approach uses a linear relaxation-based verifier [57], balancing tightness and computational efficiency, to certify nonlinear control policies (unlike the linear policies in Chang, Roohi, and Gao [11] and Jin et al. [30]) on actual dynamics, contrasting with Dai et al. [15]'s neural network dynamic approximations.

Verification and Certified robustness of NODEs. Many studies (e.g., Yan et al. [56], Kang et al. [31], and Huang et al. [27]) demonstrate improved empirical robustness of NODEs, yet certifying this robustness is challenging. Prior NODE analyses primarily address reachability: Grunbacher et al. [23] proposes a stochastic bound on the reachable set of NODEs, while Lopez et al. [37] computes deterministic reachable set of NODEs via zonotope and polynomial-zonotope based methods implemented in CORA [3]. However, these methods are limited to low-dimension or linear NODEs. MonDEQ [50], akin to implicit ODEs, has seen ℓ_2 robustness certification efforts [41, 12], but these struggle beyond MNIST. Xiao et al. [53] propose invariance propagation for stacked NODEs that provides guarantees for output specifications by controller/input synthesis. While their approach focuses more on interpretable causal reasoning of stacked NODEs, our work provides a framework for training and provably certifying general NODEs.

Formal verification of neural networks. Formal verification of neural networks aim to prove or disprove certain specifications of neural networks, and a canonical problem of neural network verification is to bound the output of neural networks given specified input perturbations. Computing the exact bounds is a NP-complete problem [32] and can be solved via MIP or SMT solvers [45, 20], but they are not scalable and often too expensive for practical usage. In the meanwhile, incomplete neural network verifiers are developed to give sound outer bounds of neural networks [43, 19, 48, 44], and bound-propagation-based methods such as CROWN [57] are a popular approach for incomplete verification. Recently, branch-and-bound based approaches [10, 49, 17] are proposed to further enhance the strength of neural network verifiers. Our work utilizes neural network verifiers as a sub-procedure to prove forward invariance of NODEs, and is agnostic to the verification algorithm used. We used CROWN because it is efficient, GPU-accelerated and has high quality implementation [54].

3.6 Discussion

This chapter extended the pointwise-condition framework from stability (LyaNet) to safety, producing the first Neural ODE controller with a formal certificate of robust forward invariance. The planar segway demonstrates that learned nonlinear controllers can achieve guarantees traditionally associated with model-based control. The classification experiments validate that the certification machinery generalizes beyond control, though the control results are the primary contribution toward the thesis goal of certified controllers for robotic systems.

The thesis introduction identified three strategies for integrating control structure with learning. FI-ODE exemplifies two of them. Training on pointwise Lyapunov and barrier conditions exemplifies the first strategy: global properties reduced to local conditions that can be sampled during training and verified over regions during certification. The CBF-QP layer that constrains dynamics to the probability simplex exemplifies the third strategy: architectural enforcement of structure that holds regardless of learned parameters. These strategies complement each other; the architectural constraint simplifies certification by reducing the space over which invariance must be verified.

Several limitations motivate subsequent chapters. The interval propagation approach degrades in tightness with state dimension and integration horizon; scaling to high-dimensional systems would require tighter relaxations or alternative certi-

fication strategies. More fundamentally, the planar segway is fully actuated and has smooth dynamics. Legged robots are underactuated and experience impacts with the ground that violate standard ODE assumptions. Chapter 6 addresses both limitations by applying pointwise conditions on zero dynamics rather than full state, and by developing discrete-time analysis that handles the Poincaré map structure of periodic locomotion with impacts. The perception side of safety-critical control, namely quantifying uncertainty in obstacle estimates, is addressed in Chapter 5.

Appendix 3.A

3.A.1 Definitions for Class $\mathcal{K}$ functions

Definition 8 (Class $\mathcal{K}$ Function). *A continuous function $\alpha : [0, a) \to [0, \infty)$ for $a \in \mathbb{R}_{>0} \cup \{\infty\}$ belongs to class $\mathcal{K}$ ($a \in \mathcal{K}$) if it satisfies:*

1. ***Zero at Zero:*** $\alpha(0) = 0$
2. ***Strictly Increasing:*** *For all $r_1, r_2 \in [0, a]$ we have that $r_1 < r_2 \Rightarrow \alpha(r_1) < \alpha(r_2)$*

Definition 9 (Class $\mathcal{K}_\infty$ Function). *A function belongs to $\mathcal{K}_\infty$ if it satisfies:*

1. $\alpha \in \mathcal{K}$
2. ***Radially Unbounded:*** $\lim_{r\to\infty} \alpha(r) = \infty$

Definition 10 (Extended Class $\mathcal{K}^e_\infty$ Function). *A continuous function $\alpha : \mathbb{R} \to \mathbb{R}$ belongs to extended $\mathcal{K}^e_\infty$ if it satisfies:*

1. ***Zero at Zero:*** $\alpha(0) = 0$
2. ***Strictly Increasing:*** *For all $r_1, r_2 \in [0, a]$ we have that $r_1 < r_2 \Rightarrow \alpha(r_1) < \alpha(r_2)$*

Definition 11 (Class $\mathcal{KL}$ Function). *A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ belongs to $\mathcal{KL}$ if it satisfies:*

1. ***Class $\mathcal{K}$ on first argument:*** $\forall s \in [0, \infty) \beta(\cdot, s) \in \mathcal{K}$

2. ***Asymptotically*** 0 ***on second argument:*** $\forall r \in [0, a) \lim_{s \to \infty} \beta(r, s) = 0$

3.A.2 Forward Invariance on a Probability Simplex

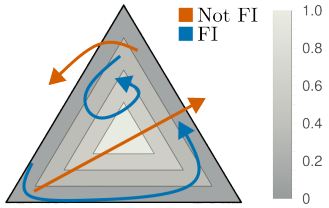


Figure 3.5: The color contours show level-sets of a barrier function in a 3-class probability simplex.

For the purposes of certification and training, it is often useful to make the state space $\mathcal{H}$ be a bounded set, as certifying over unbounded sets is typically intractable. For multi-class classification, a natural choice is the probability simplex. Since we initialize $\boldsymbol{x}$ within the simplex, it suffices to render the simplex to be forward invariant. We explicitly constrain the states to a probability simplex using a Control-Barrier Function based Quadratic Program (CBF-QP)[3] [5], implemented as a differentiable optimization layer [2].

Barrier functions can be viewed as a variant of Lyapunov functions that only require the state to stay within a set rather than always make progress towards some minimum. Specifically, we choose a potential function h with a 0-super level set (i.e. $\{\boldsymbol{x} \in \mathcal{H} | h(\boldsymbol{x}) \geq 0\}$) equal to the desired forward invariant set $\mathcal{S}$ (see Figure 3.5 for an example). Similarly to the Lyapunov case, there is a point-wise inequality condition that must be true over the forward invariant set:

$$\frac{d}{dt} h(\boldsymbol{x}(t)) \geq -\alpha(h(\boldsymbol{x})) \tag{3.9}$$

where $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}_\infty$ function. Intuitively, all the flows on the boundary of the forward invariant set must have a positive time-derivative (otherwise there could be a point on the boundary that decreases the value of h and thus exits the forward invariant set). This is the essence of Nagumo's theorem [40]. Barriers extend this idea with a condition that can be applied everywhere in the target forward invariant set without being overly conservative. As trajectories approach the boundary of the set, Equation (3.9) ensures the time derivative increases until it

[3] This is analogous to projected gradient descent (PGD) where we project the dynamics instead of the states.

is positive at the boundary. We use a variation of barrier functions called Control Barrier Functions (CBF). We formalize this concept with Theorem 7.

In our case, the unconstrained dynamics is the output of a neural network and we denote it as $\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{o})$. To make the dynamics satisfy the barrier conditions, we use a Control Barrier Function Quadratic Program (CBF-QP) Safety Filter [24]:

$$\boldsymbol{f}(\hat{\boldsymbol{f}}) = \underset{\mathtt{f} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|\mathtt{f} - \hat{\boldsymbol{f}}\|_2^2 \tag{3.10a}$$

$$\text{s.t} \qquad \mathbb{1}^\top \mathtt{f} = 0 \tag{3.10b}$$

$$\mathtt{f} \geq -\alpha(\boldsymbol{x}) \tag{3.10c}$$

where the arguments to the function $\hat{\boldsymbol{f}}$ are omitted for brevity.

Recall that an n-class probability simplex is defined as $\triangle = \{\boldsymbol{x} \in \mathbb{R}^n | \sum_{i=1}^n \boldsymbol{x}_i = 1, \boldsymbol{x}_i \geq 0\}$. Now we show that Equation (3.10b) ensures that the sum of the state stays to be 1 and Equation (3.10c) guarantees the state to be non-negative.

First, we need the sum of $\boldsymbol{x}$ stays the same as the initial condition. Taking time derivative of both sides of $\sum_{i=1}^n \boldsymbol{x}_i = 1$, we have $\frac{d}{dt}\left(\sum_{i=1}^n \boldsymbol{x}_i(t)\right) = \sum_{i=1}^n \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o})_i = \mathbb{1}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) = 0$, which is Equation (3.10b). This is natural because the dynamics summing up to zero means the changes from all dimensions summing up to zero, and thus the sum of all dimensions stays the same.

Next, we need each dimension of the state to be non-negative. Since the initial condition has non-negative entries, we just need the set $\{\boldsymbol{x} | \boldsymbol{x} \geq 0\}$ to be forward invariant. We define forward invariance via barrier functions. For each dimension i, we define $h_i(\boldsymbol{x}) = \boldsymbol{x}_i$. Then the 0-superlevel set of h_i equals the safe set $\{\boldsymbol{x} | \boldsymbol{x}_i \geq 0\}$. As long as the condition in Equation (3.9) holds, i.e. $\frac{dh_i}{d\boldsymbol{x}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) \geq -\alpha(h_i(\boldsymbol{x}))$ for some class $\mathcal{K}_\infty$ function α, the set $\{\boldsymbol{x} | \boldsymbol{x}_i \geq 0\}$ is forward invariant. Plugging in $h_i(\boldsymbol{x})$, we have $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) \geq -\alpha(\boldsymbol{x})$, which is Equation (3.10c).

To learn in this setting we differentiate through the QP layer using the KKT conditions as shown in [2]. Given the simple nature of the QP, we implemented a custom solver that uses binary search to efficiently compute solutions, detailed in the supplymentary materials.

To demonstrate the effectiveness of the CBF-QP layer, we visualize the learned trajectories on the CIFAR-3 dataset (a subset of CIFAR-10 with the first 3 classes) in Figure 3.6. Each colored line represents a trajectory of an input image from a specific class. As training progresses, the trajectories are trained to evolve to the

correct classes. All the trajectories stay in the simplex, implying that the learned dynamics satisfy the constraints in Equation (3.10b) and Equation (3.10c).

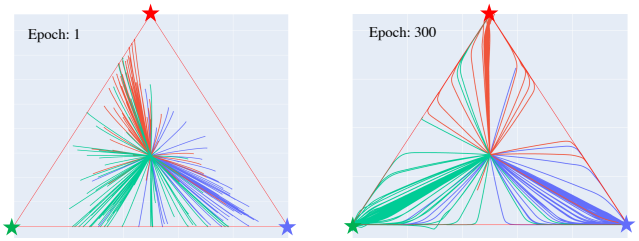


Figure 3.6: Depicting ODE trajectories that satisfy the simplex constraint for CIFAR-3 on epochs 1 and 300. Each colored line represents the trajectory of an input example of a specific class, and the stars at the corners are colored with the ground-truth class.

3.A.3 Theorems with Proof

Theorem 6 (ES-CLF Implies Exponential Stability [4]). *For the ODE in Equations* (3.1a) *and* (3.1b)*, a continuously differentiable function* $V : \mathbb{R}^k \to \mathbb{R}_{\geq 0}$ *is an Exponentially Stabilizing Control Lyapunov Function (ES-CLF) if there are class* $\mathcal{K}_\infty$ *functions* $\overline{\sigma}$ *and* κ *such that:*

$$V(\boldsymbol{x}) \leq \overline{\sigma}(\|\boldsymbol{x}\|), \tag{3.11}$$

$$\min_{\boldsymbol{\theta}\in\Theta}\left[\frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{o}) + \kappa(V(\boldsymbol{x}))\right] \leq 0 \tag{3.12}$$

holds for all $\boldsymbol{x} \in \mathcal{R} \subseteq H$ *and* $t \in [0,1]$*. The existence of an ES-CLF implies that there is a* $\boldsymbol{\theta} \in \Theta$ *that can achieve:*

$$\frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{o}) + \kappa(V(\boldsymbol{x})) \leq 0, \tag{3.13}$$

and furthermore the ODE using $\boldsymbol{\theta}$ *is exponentially stable with respect to* V*, i.e.,* $V(\boldsymbol{x}(t)) \leq V(\boldsymbol{x}(0))e^{-\underline{\kappa}t}$ *for some* $\underline{\kappa} > 0$*.*

Proof. Since Θ is compact, minimums are attained within Θ. Let $\boldsymbol{\theta}^*(\boldsymbol{x}) = \operatorname{argmin}_{\boldsymbol{\theta}\in\Theta} \frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{o})$. Therefore, from Equation (3.12) we can conclude that:

$$\frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}^*(\boldsymbol{x})}(\boldsymbol{x},\boldsymbol{o}) \leq -\kappa(V(\boldsymbol{x})) \qquad \forall \boldsymbol{x} \in \mathcal{H} \tag{3.14}$$

For simplicity we will omit the arguments of $\boldsymbol{\theta}^*$. Furthermore, in the case where $\boldsymbol{\theta}^*$ is a set we will select only one. Since $\mathcal{H}$ is compact then

$$\sigma^* = \max_{\boldsymbol{x}\in\mathcal{H}} \overline{\sigma}(\|\boldsymbol{x}\|). \tag{3.15}$$

This in turn implies that V in bounded in $\mathcal{H}$ which helps us conclude that

$$\overline{V} = \max_{\boldsymbol{x} \in \mathcal{H}} V(\boldsymbol{x}) \tag{3.16}$$

is well defined which in turn implies

$$\underline{\kappa} = \min_{r \in [0, \overline{V}]} \frac{d\kappa(r)}{dr} \tag{3.17}$$

is well defined. Since κ is strictly increasing, then $\underline{\kappa} > 0$. Notice that $\alpha(r) = \underline{\kappa} r$ satisfies $\alpha \in \mathcal{K}_\infty$ and, by the comparison lemma, $\forall r, \underline{\kappa} r \leq \kappa(r)$. Therefore:

$$\frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}^*(\boldsymbol{x})}(\boldsymbol{x}, \boldsymbol{x}) \leq -\kappa(V(\boldsymbol{x})) \leq -\underline{\kappa} V(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \mathcal{H} \tag{3.18}$$

In preparation for applying the comparison lemma we will consider the following Initial Value Problem (IVP):

$$y(0) = V(\boldsymbol{x}_0) \tag{3.19}$$

$$\dot{y} = -\underline{\kappa} y \tag{3.20}$$

Since this is a linear system solutions for y exist, are unique and take the form $y(t) = V(\boldsymbol{x}_0) e^{-\underline{\kappa} t}$. Furthermore, by the comparison lemma we can conclude that:

$$V(\boldsymbol{x}(t)) \leq y(t) = V(\boldsymbol{x}_0) e^{-\underline{\kappa} t} \tag{3.21}$$

□

Lemma 2 (Solution of Class $\mathcal{K}$ function systems (See Lemma 4.4 in [33])). *Let $\alpha \in \mathcal{K}^e_\infty$. Then consider the following IVP for $t \in [0, 1]$:*

$$y(0) = y_0 \tag{3.22}$$

$$\dot{y} = -\alpha(y) \tag{3.23}$$

This IVP has unique solutions $y(t) = \beta(y_0, t)$ where $\beta \in \mathcal{KL}$.

Theorem 7 (CBF Existence Implies Forward Invariance [55, 40]). *Let the set $\mathcal{S} \subset \mathcal{H}$ be the 0 superlevel set of a continuously differentiable function $h : \mathcal{H} \to \mathbb{R}$, i.e. $\mathcal{S} = \{\boldsymbol{x} \in \mathcal{H} | h(\boldsymbol{x}) \geq 0\}$. The set $\mathcal{S}$ is forward invariant with respect to the ODE Equations* (3.1a) *and* (3.1b)*, if h is a Control Barrier Function (CBF) i.e. it satisfies either of the following conditions:*

1. *[55] There exists a function α for all $\boldsymbol{x} \in \mathcal{S}$ so that:*

$$\max_{\boldsymbol{\theta}\in\Theta} \left[\frac{\partial h}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) + \alpha(h(\boldsymbol{x})) \right] \geq 0, \tag{3.24}$$

   where α is a class $\mathcal{K}_\infty^e$ function (this means $\alpha : \mathbb{R} \to \mathbb{R}$ is strictly increasing and satisfies $\lim_{r\to\infty} \alpha(r) = \infty$).

2. *[40] For all $\boldsymbol{x} \in \{\boldsymbol{x} \in \mathcal{S} | h(\boldsymbol{x}) = 0\}$:*

$$\frac{\partial h}{\partial \boldsymbol{x}} \neq \boldsymbol{0} \tag{3.25a}$$

$$\max_{\boldsymbol{\theta}\in\Theta} \left[\frac{\partial h}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) \right] \geq 0. \tag{3.25b}$$

Proof. Both conditions follow from fundamentally different arguments. Condition 1 follows from the comparison lemma. Condition 2 uses Nagumo's theorem. In either case we rely on the compactness of Θ to solve the following optimization problem:

$$\boldsymbol{\theta}^*(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \left[\frac{\partial h}{\partial \boldsymbol{x}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{o}) \right] \tag{3.26}$$

We will omit the parameters of $\boldsymbol{\theta}^*$ for brevity and choose a random solution in the case where $\boldsymbol{\theta}^*$ returns a set of solutions.

1. Consider the following IVP:

$$y(0) = h(\boldsymbol{x}(0)) \tag{3.27}$$

$$\dot{y} = -\alpha(y) \tag{3.28}$$

   which satisfies the conditions of Lemma 2. This implies that $y(t)$ is unique and $y(t) = \beta(h(\boldsymbol{x}(0)), t)$ where by the assumption of forward invariance $h(\boldsymbol{x}(0)) \geq 0$. The by a trivial variant of the comparison lemma we have that $\dot{h}(\boldsymbol{x}(t)) \geq -\alpha(h(\boldsymbol{x}))$ implies $h(\boldsymbol{x}(t)) \geq \beta(h(\boldsymbol{x}(0)), t)$ which implies $h(\boldsymbol{x}(t)) \geq 0$ for $t \in [0, 1]$

2. This is a direct application of Nagumo's theorem [40].

□

3.A.4 Proof of Theorem 4

Proof. Define barrier function h as $h = c - V$, then $\mathcal{S}$ is a 0-superlevel set of h. According to Nagumo's theorem [40], if $\dot{h}(\boldsymbol{o} + \boldsymbol{\epsilon}; \boldsymbol{x}) \geq 0$ on $\partial\mathcal{D} := \mathcal{S}$, then $\mathcal{S}$ is forward invariant. Since $\dot{V}(\boldsymbol{x}; \boldsymbol{o}) \leq -L_V L_f^o \bar{\epsilon}$ on $\mathcal{D}$, we have $\dot{h}(\boldsymbol{x}; \boldsymbol{o}) \geq L_h L_f^o \bar{\epsilon}$ on $\mathcal{D}$ (where L_h is the Lipschitz constant of h and notice that $L_V = L_h$). Then for the perturbed input, we have

$$\dot{h}(\boldsymbol{x}; \boldsymbol{o} + \boldsymbol{\epsilon}) = \dot{h}(\boldsymbol{x}; \boldsymbol{o}) + \dot{h}(\boldsymbol{x}; \boldsymbol{o} + \boldsymbol{\epsilon}) - \dot{h}(\boldsymbol{x}; \boldsymbol{o}) \tag{3.29}$$

$$\geq \dot{h}(\boldsymbol{x}; \boldsymbol{o}) - \|\dot{h}(\boldsymbol{x}; \boldsymbol{o} + \boldsymbol{\epsilon}) - \dot{h}(\boldsymbol{x}; \boldsymbol{o})\| \tag{3.30}$$

$$\geq \dot{h}(\boldsymbol{x}; \boldsymbol{o}) - L_h L_f^o \bar{\epsilon} \tag{3.31}$$

$$\geq \kappa \underline{V} - L_h L_f^o \bar{\epsilon} \geq 0 \tag{3.32}$$

Therefore, $\mathcal{S}$ is still forward invariant for the perturbed inputs with perturbation magnitude smaller than $\bar{\epsilon}$. □

3.A.5 Proof of Theorem 5

Proof. **(a) Sampling on the level set of a quadratic Lyapunov function.** Consider the Lyapunov function of the form $V(\boldsymbol{x}) = \boldsymbol{x}^\top P \boldsymbol{x}$, where P is a positive definite matrix. Let $\mathcal{D} = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x}^\top P \boldsymbol{x} = c\}$ be the c-level set of the Lyapunov function, and let $\mathcal{G}$ be a uniform grid (with spacing r) that covers the Lyapunov level set, i.e. $\min_{\boldsymbol{x} \in \mathcal{D}} \boldsymbol{x}_i \geq \min_{\boldsymbol{x} \in \mathcal{G}} \boldsymbol{x}_i$ and $\max_{\boldsymbol{x} \in \mathcal{D}} \boldsymbol{x}_i \leq \max_{\boldsymbol{x} \in \mathcal{G}} \boldsymbol{x}_i$, $\forall i = 1, ..., n$. Then $\forall \boldsymbol{x} \in \mathcal{D}$, there exists at least a point $\boldsymbol{g} \in \mathcal{G}$ such that $|\boldsymbol{x}_i - \boldsymbol{g}_i| \leq \frac{r}{2}$ for all $i = 1, ..., n$. Let $\mathcal{G}_D$ denote those grid points that are close to the decision boundary, i.e. $\mathcal{G}_D = \{\boldsymbol{g} \in \mathcal{G} | |\boldsymbol{x}_i - \boldsymbol{g}_i| \leq \frac{r}{2}, \text{ for all } i = 1, ..., n, \forall \boldsymbol{x} \in \mathcal{D}\}$.

Now we show that $\mathcal{G}_D \subseteq \{\mathcal{B} \cap \mathcal{G}\}$, where $\mathcal{B}$ is defined in Equation (3.7). Notice that the maximum ℓ_2 distance between a point $\boldsymbol{x} \in \mathcal{D}$ and its closest point $\boldsymbol{g} \in \mathcal{G}_D$ is $\frac{\sqrt{n}}{2} r$. Then we find the maximum and minimum Lyapunov function value by perturbing $\boldsymbol{x} \in \mathcal{D}$ within $\frac{\sqrt{n}}{2} r$ distance.

Consider the following maximization problem:

$$\max\ (\boldsymbol{x} + \boldsymbol{v})^\top P (\boldsymbol{x} + \boldsymbol{v}) \tag{3.33a}$$

$$\text{s.t.} \quad \boldsymbol{x}^\top P \boldsymbol{x} = c \tag{3.33b}$$

$$\boldsymbol{v}^\top \boldsymbol{v} = d^2 \tag{3.33c}$$

Let the eigenvalue decomposition of P be $P = U \Lambda U^\top$, where U is an orthonormal matrix, and $\Lambda = \text{diag}\{\lambda_1, \lambda_2, ..., \lambda_n\}$ with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$. Then the solution

of the above problem equals to the solution of the following problem (rotating the coordinates by U):

$$\max\ (\boldsymbol{x}+\boldsymbol{v})^\top \Lambda(\boldsymbol{x}+\boldsymbol{v}) \tag{3.34a}$$

$$\text{s.t.}\quad \boldsymbol{x}^\top \Lambda \boldsymbol{x} = c \tag{3.34b}$$

$$\boldsymbol{v}^\top \boldsymbol{v} = d^2 \tag{3.34c}$$

Then we can find the upper bound of the objective by the following:

$$\sum_{i=1}^{n}(\boldsymbol{x}_i+\boldsymbol{v}_i)\lambda_i = c + \sum_{i=1}^{n}\lambda_i \boldsymbol{v}_i^2 + 2\sum_{i=1}^{n}\lambda_i \boldsymbol{x}_i \boldsymbol{v}_i \tag{3.35}$$

$$\leq c + \lambda_1 d^2 + 2\sum_{i=1}^{n}\lambda_i \boldsymbol{x}_i \boldsymbol{v}_i \tag{3.36}$$

$$\leq c + \lambda_1 d^2 + 2\sqrt{\sum_{i=1}^{n}(\sqrt{\lambda_i}\boldsymbol{x}_i)^2 \sum_{i=1}^{n}(\sqrt{\lambda_i}\boldsymbol{v}_i)^2} \tag{3.37}$$

$$= c + \lambda_1 d^2 + 2\sqrt{c\sum_{i=1}^{n}\lambda_i \boldsymbol{v}_i^2} \tag{3.38}$$

$$\leq c + \lambda_1 d^2 + 2d\sqrt{c\lambda_1} \tag{3.39}$$

$$= (\sqrt{c} + d\sqrt{\lambda_1})^2 := \overline{c} \tag{3.40}$$

Similarly, we can find the lower bound of the objective by:

$$\sum_{i=1}^{n}(\boldsymbol{x}_i+\boldsymbol{v}_i)\lambda_i = c + \sum_{i=1}^{n}\lambda_i \boldsymbol{v}_i^2 + 2\sum_{i=1}^{n}\lambda_i \boldsymbol{x}_i \boldsymbol{v}_i \tag{3.41}$$

$$\geq c + \sum_{i=1}^{n}\lambda_i \boldsymbol{v}_i^2 - 2\sqrt{c\sum_{i=1}^{n}\lambda_i \boldsymbol{v}_i^2} \tag{3.42}$$

$$= \left(\sqrt{c} - \sqrt{\sum_{i=1}^{n}\lambda_i \boldsymbol{v}_i^2}\right)^2 \tag{3.43}$$

$$\geq (\sqrt{c} - d\sqrt{\lambda_1})^2 := \underline{c} \tag{3.44}$$

Therefore, the maximum and minimum Lyapunov function value the points in $\mathcal{G}_D$ can attain are $\overline{c}$ and $\underline{c}$ with $d = \frac{\sqrt{n}}{2}r$, and thus we have $\mathcal{G}_D \subseteq \{\mathcal{B} \cap \mathcal{G}\}$.

By definition of $\mathcal{G}_D$, we have that for any $\boldsymbol{x} \in \mathcal{D}$, there exist an $\boldsymbol{s} \in \mathcal{G}_D$ such that $|\boldsymbol{x}_i - \boldsymbol{s}_i| \leq \frac{r}{2}$ for all $i = 1, ..., n$. Since $\mathcal{G}_D \in \{\mathcal{B} \cap \mathcal{G}\}$, we have that for any $\boldsymbol{x} \in \mathcal{D}$, there exist an $\boldsymbol{s} \in \{\mathcal{B} \cap \mathcal{G}\}$ such that $|\boldsymbol{x}_i - \boldsymbol{s}_i| \leq \frac{r}{2}$ for all $i = 1, ..., n$.

(b) Sampling on the decision boundary. For any $\boldsymbol{x} \in \mathscr{D}_y$, let $z = [N\boldsymbol{x}_1, ..., N\boldsymbol{x}_n]$. By definition of $\mathscr{D}_y$, in addition to $\sum_i z_i = N$, we have

$$\sum_i z_i = N \tag{3.45}$$

$$z_y = \max_{j \neq y} z_j \tag{3.46}$$

Define $\tilde{z} = [z_1 - \lfloor z_1 \rfloor, ..., z_n - \lfloor z_n \rfloor]$ to be the vector that contains the fractional part of each element in z. Then we sort $\tilde{z}$ in a non-decreasing order. For the tied elements that equals to $\tilde{z}_y$, we put $\tilde{z}_y$ as the last. We denote the sorted vector as $\tilde{z}' = [\tilde{z}_{i_1}, ..., \tilde{z}_{i_n}]$, where $\tilde{z}_{i_1} \leq ... \leq \tilde{z}_{i_n}$. Let $v : \mathbb{Z}^+ \to \mathbb{Z}^+$ to be a function that maps the indices in $\tilde{z}$ to the indices in $\tilde{z}'$. For instance, if $\tilde{z}_1$ becomes the third element in $\tilde{z}'$, then $v(1) = 3$. If $\tilde{z}_j = \tilde{z}_y$, we have $v(y) > v(j)$.

Notice that $\sum_i \tilde{z}'_i = \sum_i \tilde{z}_i = \sum_i z_i - \sum_i \lfloor z_i \rfloor = N - \sum_i \lfloor z_i \rfloor$, and $\sum_i \tilde{z}'_i < n$ since $0 \leq \tilde{z}'_i < 1$ for $i = 1, ..., n$. Let $k = n - (N - \sum_i \lfloor z_i \rfloor)$, we have

$$\tilde{z}'_1 + ... + \tilde{z}'_k = (1 - \tilde{z}'_{k+1}) + ... + (1 - \tilde{z}'_n) \tag{3.47}$$

Define vector $\boldsymbol{q}$ as follows:

$$\boldsymbol{q}_{i_j} = \begin{cases} \lfloor N\boldsymbol{x}_{i_j} \rfloor, & j = 1, ..., k \\ \lceil N\boldsymbol{x}_{i_j} \rceil, & j = k+1, ..., n \end{cases} \tag{3.48}$$

Then we have $|\boldsymbol{q}_i - z_i| < 1$ for all $i = 1, ..., n$. Now we check $\boldsymbol{q}$ satisfies Equation (3.45) and a relaxed version of Equation (3.46). First, we have $\sum_i \boldsymbol{q}_i = N$ because of Equation (3.47). Next, we show $\boldsymbol{q}_y \geq \max_{i \neq y} \boldsymbol{q}_i$ by contradiction. Suppose there exists an index j such that $\boldsymbol{q}_j > \boldsymbol{q}_y$, then it has to be the case where $\lfloor z_j \rfloor = \lfloor z_y \rfloor$ and we take ceiling on z_j and take floor on z_y, i.e. $v(j) > k$ and $v(y) \leq k$. This means $\tilde{z}_j > \tilde{z}_y$, because v is the sorted indices of $\tilde{z}$ in a non-decreasing order and this gives $\tilde{z}_j \geq \tilde{z}_y$, and if $\tilde{z}_j = \tilde{z}_y$, we have $v(y) > v(j)$, which is contradictory to $v(j) > k$ and $v(y) \leq k$. Then we have $z_y = \lfloor z_y \rfloor + \tilde{z}_y < z_j = \lfloor z_j \rfloor + \tilde{z}_j$, which is contradictory to Equation (3.46). Therefore, there does not exist a j such that $\boldsymbol{q}_j > \boldsymbol{q}_y$, i.e. $\boldsymbol{q}_y \geq \max_{i \neq y} \boldsymbol{q}_i$. For the cases where $\boldsymbol{q}_y = \max_{i \neq y} \boldsymbol{q}_i$, we have $\boldsymbol{q} \in S_y$, i.e. $\boldsymbol{q}$ is a sampled point.

For the cases where $\boldsymbol{q}_y > \max_{i \neq y} \boldsymbol{q}_i$, we show that we can modify $\boldsymbol{q}$ to $\tilde{\boldsymbol{q}}$ such that $\tilde{\boldsymbol{q}} \in S_y$ and $|\tilde{\boldsymbol{q}}_i - z_i| \leq 1$ for all $i = 1, ..., n$. Let $\mathcal{I} = \{i \in \mathbb{Z}^+ | z \neq y, z_i = z_y\}$ be the set that contains the indices of all runner-up elements in z. If $\boldsymbol{q}_y > \max_{i \neq y} \boldsymbol{q}_i$, then we must have $\boldsymbol{q}_y = \lceil N\boldsymbol{x}_y \rceil$, and $\boldsymbol{q}_i = \lfloor N\boldsymbol{x}_i \rfloor$ for all $i \in \mathcal{I}$. We first let $\tilde{\boldsymbol{q}} = \boldsymbol{q}$,

and then pick an i^* from $\mathcal{I}$. Let $\mathcal{J} = \{j \in \mathbb{Z}^+ | \boldsymbol{q}_j \geq 1, j \neq i^*, j \neq y\}$. Notice that $\boldsymbol{q}_{i^*} + \boldsymbol{q}_y = 2\lfloor z_y \rfloor + 1$, which is an odd number. Since $\sum_i \boldsymbol{q}_i = N$ and N is an even number, $\mathcal{J} \neq \emptyset$. We discuss how to obtain $\tilde{\boldsymbol{q}}$ case by case.

Case 1: If there exists a $j \in \mathcal{J}$ such that $v(j) > k$, we set $\tilde{\boldsymbol{q}}_j = \lfloor N\boldsymbol{x}_j \rfloor$, and set $\tilde{\boldsymbol{q}}_{i^*} = \lceil N\boldsymbol{x}_{i^*} \rceil$. Then we have $\sum_i \tilde{\boldsymbol{q}} = \sum_{i \neq i^*, i \neq j} \boldsymbol{q}_i + \boldsymbol{q}_{i^*} + 1 + \boldsymbol{q}_j - 1 = N$ and $|\tilde{\boldsymbol{q}}_i - z_i| \leq 1$ for all $i = 1, ..., n$.

Case 2: If $v(j) \leq k$ for all $j \in \mathcal{J}$, there must exist a j such that $\boldsymbol{q}_j < \boldsymbol{q}_{i^*}$. Otherwise, $\boldsymbol{q}_y = \boldsymbol{q}_{i^*} + 1$, and $\boldsymbol{q}_i = \boldsymbol{q}_{i^*}$ for all $i \neq y$. Then $\sum_i \boldsymbol{q}_i = N = n\boldsymbol{q}_{i^*} + 1$, which is contradictory to the assumption that $N \not\equiv 1 (\mathrm{mod}\, n)$. Then set $\tilde{\boldsymbol{q}}_j = \lceil N\boldsymbol{x}_j \rceil$ and $\tilde{\boldsymbol{q}}_y = \lfloor N\boldsymbol{x}_y \rfloor$. Since $\boldsymbol{q}_j < \boldsymbol{q}_{i^*}$, we have $\tilde{\boldsymbol{q}}_j = \boldsymbol{q}_j + 1 \leq \tilde{\boldsymbol{q}}_y = \boldsymbol{q}_{i^*}$. □

Remark 3. The assumption that $N \not\equiv 1 (\mathrm{mod}\, n)$ is easy to satisfy. Since we also require N is an even number, as long as n is also an even number, we have $N \not\equiv 1 (\mathrm{mod}\, n)$. We can also relax this assumption by adding $[\frac{N}{n}, ..., \frac{N}{n}]$ to S_y.

3.A.6 Custom solver for the CBF-QP

Consider a CBF-QP in the following form:

$$\begin{aligned} \boldsymbol{f}(\hat{\boldsymbol{f}}) = \underset{\mathtt{f} \in \mathbb{R}^n}{\operatorname{argmin}} \; & \frac{1}{2} \|\mathtt{f} - \hat{\boldsymbol{f}}\|_2^2 \\ \text{s.t} \quad & \mathbb{1}^\top \mathtt{f} = b \\ & \underline{\mathtt{f}}(\boldsymbol{x}) \leq \mathtt{f} \leq \overline{\mathtt{f}}(\boldsymbol{x}) \end{aligned} \tag{3.49}$$

where $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$ are non-increasing function of $\boldsymbol{x}$. By the Karush–Kuhn–Tucker (KKT) conditions, the solution of (3.49) is as follows:

$$\boldsymbol{f}(\hat{\boldsymbol{f}}) = \left[\hat{\boldsymbol{f}} + \lambda^* \mathbb{1}\right]_{\underline{\mathtt{f}}}^{\overline{\mathtt{f}}} \tag{3.50}$$

where $[\cdot]_{\underline{\mathtt{f}}}^{\overline{\mathtt{f}}}$ stands for lower and upper clipping by $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$, and λ^* is the Lagrangian multiplier. We find λ^* such that $\mathbb{1}^\top \boldsymbol{f}(\hat{\boldsymbol{f}}) = b$ using binary search. Since $\boldsymbol{f}(\hat{\boldsymbol{f}})$ is clipped by $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$, the search range of λ^* is $[\min_i(\hat{\boldsymbol{f}}_i - \underline{\mathtt{f}}_i), \max_i(\overline{\mathtt{f}}_i - \hat{\boldsymbol{f}}_i)]$, where $\hat{\boldsymbol{f}}_i$ stands for the ith element in $\hat{\boldsymbol{f}}$, and $\underline{\mathtt{f}}_i$, $\overline{\mathtt{f}}_i$ stand for the ith element in $\underline{\mathtt{f}}(\boldsymbol{x})$ and $\overline{\mathtt{f}}(\boldsymbol{x})$ respectively. Here we consider a general constraint where there are both lower and upper bounds on $\mathtt{f}$. If there is only a lower bound constraint on $\mathtt{f}$ as in Equation (3.10c), we search λ^* in $[\min_i(\hat{\boldsymbol{f}}_i - \underline{\mathtt{f}}_i), -\min_i \hat{\boldsymbol{f}}_i]$, because if $\lambda^* > -\min_i \hat{\boldsymbol{f}}_i$, then $\mathbb{1}^\top \mathtt{f} > 0$, violating Equation (3.10b).

To differentiate through the solver in training, we derive the derivatives based on the binding conditions of the inequality constraints. First, we define the binding and

not binding sets as follows:

$$\mathcal{S} = \{i | \boldsymbol{f}_i = \underline{\mathtt{f}}_i \text{ or } \boldsymbol{f}_i = \overline{\mathtt{f}}_i\}, \quad \mathcal{S}^c = \Omega \backslash S \tag{3.51}$$

$$\mathcal{S}_l = \{i | \boldsymbol{f}_i = \underline{\mathtt{f}}_i\}, \quad \mathcal{S}_l^c = \Omega \backslash \mathcal{S}_l \tag{3.52}$$

$$\mathcal{S}_u = \{i | \boldsymbol{f}_i = \overline{\mathtt{f}}_i\}, \quad \mathcal{S}_u^c = \Omega \backslash \mathcal{S}_u \tag{3.53}$$

where $\Omega = \{i \in \mathbb{Z}^+ | i \leq n\}$. Then the derivatives of $\boldsymbol{f}$ with respect to the inputs $\hat{\boldsymbol{f}}$, $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$ are as follows:

$$\frac{d\boldsymbol{f}_i}{d\hat{\boldsymbol{f}}_j} = \begin{cases} 0, & i \in \mathcal{S}^c \text{ or } j \in \mathcal{S}^c \\ 1 - \frac{1}{n(\mathcal{S})}, & i = j \in \mathcal{S} \\ -\frac{1}{n(\mathcal{S})} & i \neq j, i \in \mathcal{S}, j \in \mathcal{S} \end{cases} \tag{3.54}$$

$$\frac{d\boldsymbol{f}_i}{d\underline{\mathtt{f}}_j} = \begin{cases} 0, & j \in \mathcal{S}_l, \forall i \in \Omega \\ 0, & j \in \mathcal{S}_l^c, i \in \mathcal{S}_l^c \backslash \{j\} \\ 1, & j \in \mathcal{S}_l^c, i = j \\ -\frac{1}{n(\mathcal{S}_l)} & j \in \mathcal{S}_l^c, i \in \mathcal{S}_l \end{cases} \quad \frac{d\boldsymbol{f}_i}{d\overline{\mathtt{f}}_j} = \begin{cases} 0, & j \in \mathcal{S}_u, \forall i \in \Omega \\ 0, & j \in \mathcal{S}_u^c, i \in \mathcal{S}_u^c \backslash \{j\} \\ 1, & j \in \mathcal{S}_u^c, i = j \\ -\frac{1}{n(\mathcal{S}_u)} & j \in \mathcal{S}_u^c, i \in \mathcal{S}_u \end{cases} \tag{3.55}$$

3.A.7 Interval Bound Propagation through CBF-QP

The dynamics of our NODE is parameterized by a neural network followed by a CBF-QP layer. Let $\hat{\boldsymbol{f}}(\boldsymbol{x})$ be the dynamics output by the neural network, and let $\boldsymbol{f}(\hat{\boldsymbol{f}})$ be the dynamics after CBF-QP layer. Given perturbed input in an interval bound $\underline{\boldsymbol{x}_i} \leq \boldsymbol{x}_i \leq \overline{\boldsymbol{x}_i}$, we first use a a popular linear relaxation based verifier named CROWN [57] to get an interval bound for $\hat{\boldsymbol{f}}$: $\underline{\hat{\boldsymbol{f}}_i} \leq \hat{\boldsymbol{f}}_i \leq \overline{\hat{\boldsymbol{f}}_i}$. However, CROWN does not support perturbation analysis on differentiable optimization layers such as our CBF-QP layer and deriving linear relaxation for CBF-QP can be hard. However, it is possible to derive interval bounds (a special case of linear bounds in CROWN) through CBF-QP. Consider a QP in the form of Equations (3.10a) to (3.10c), we bound each dimension of $\boldsymbol{f}(\hat{\boldsymbol{f}})$ in $O(n)$ by solving the QP with the corresponding element of the input set to the lower or upper bound (Proposition 1).

Proposition 1. *Consider a CBF-QP in the form of 3.49. Define function h_i to be $h_i : \boldsymbol{x}, \hat{\boldsymbol{f}} \mapsto \boldsymbol{f}(\hat{\boldsymbol{f}})_i$. Given perturbed input in an interval bound $\underline{\boldsymbol{x}_i} \leq \boldsymbol{x}_i \leq \overline{\boldsymbol{x}_i}$, and $\underline{\hat{\boldsymbol{f}}_i} \leq \hat{\boldsymbol{f}}_i \leq \overline{\hat{\boldsymbol{f}}_i}$, we have*

$$h_i(\boldsymbol{x}_{ub}^i, \hat{\boldsymbol{f}}_{lb}^i) \leq \boldsymbol{f}(\hat{\boldsymbol{f}})_i \leq h_i(\boldsymbol{x}_{lb}^i, \hat{\boldsymbol{f}}_{ub}^i) \tag{3.56}$$

where $\boldsymbol{x}^i_{ub}, \boldsymbol{x}^i_{lb}$ *and* $\hat{\boldsymbol{f}}^i_{ub}, \hat{\boldsymbol{f}}^i_{lb}$ *are defined as follows:*

$$\boldsymbol{x}^i_{ub} = \begin{cases} \overline{\boldsymbol{x}_j}, & j = i \\ \underline{\boldsymbol{x}_j}, & j \neq i \end{cases} \qquad \boldsymbol{x}^i_{lb} = \begin{cases} \underline{\boldsymbol{x}_j}, & j = i \\ \overline{\boldsymbol{x}_j}, & j \neq i \end{cases} \tag{3.57}$$

$$\hat{\boldsymbol{f}}^i_{ub} = \begin{cases} \overline{\hat{\boldsymbol{f}}_j}, & j = i \\ \underline{\hat{\boldsymbol{f}}_j}, & j \neq i \end{cases} \qquad \hat{\boldsymbol{f}}^i_{lb} = \begin{cases} \underline{\hat{\boldsymbol{f}}_j}, & j = i \\ \overline{\hat{\boldsymbol{f}}_j}, & j \neq i \end{cases} \tag{3.58}$$

Proof. We prove by contradiction. For the upper bound $\boldsymbol{f}(\hat{\boldsymbol{f}})_i \leq h_i(\boldsymbol{x}^i_{lb}, \hat{\boldsymbol{f}}^i_{ub})$, suppose $\boldsymbol{f}(\hat{\boldsymbol{f}})_i > h_i(\boldsymbol{x}^i_{lb}, \hat{\boldsymbol{f}}^i_{ub})$. Plug in Equation (3.50), we have

$$\left[\hat{\boldsymbol{f}}_i + \lambda\right]^{\overline{\mathtt{f}}(\boldsymbol{x}_i)}_{\underline{\mathtt{f}}(\boldsymbol{x}_i)} > \left[\overline{\hat{\boldsymbol{f}}_i} + \lambda'\right]^{\overline{\mathtt{f}}(\underline{\boldsymbol{x}_i})}_{\underline{\mathtt{f}}(\overline{\boldsymbol{x}_i})} \tag{3.59}$$

Since $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$ are non-increasing function of $\boldsymbol{x}$, we have $\underline{\mathtt{f}}(\overline{\boldsymbol{x}_i}) \geq \underline{\mathtt{f}}(\boldsymbol{x}_i)$ and $\overline{\mathtt{f}}(\underline{\boldsymbol{x}_i}) \geq \overline{\mathtt{f}}(\boldsymbol{x}_i)$. Then $\hat{\boldsymbol{f}}_i + \lambda > \overline{\hat{\boldsymbol{f}}_i} + \lambda'$. Since $\hat{\boldsymbol{f}}_i \leq \overline{\hat{\boldsymbol{f}}_i}$, we have $\lambda > \lambda'$. Then for all $j \neq i$, we have

$$\left[\hat{\boldsymbol{f}}_j + \lambda\right]^{\overline{\mathtt{f}}(\boldsymbol{x}_j)}_{\underline{\mathtt{f}}(\boldsymbol{x}_j)} \geq \left[\underline{\hat{\boldsymbol{f}}_j} + \lambda'\right]^{\overline{\mathtt{f}}(\underline{\boldsymbol{x}_i})}_{\underline{\mathtt{f}}(\overline{\boldsymbol{x}_i})} \tag{3.60}$$

Sum on both sides of 3.59 and 3.60, we have

$$\mathbb{1}^\top f(\boldsymbol{x}, \hat{\boldsymbol{f}}) > \mathbb{1}^\top f(\boldsymbol{x}_{lb}, \hat{\boldsymbol{f}}_{ub}) \tag{3.61}$$

which is contradictory to the equality constraint in 3.49. Therefore, we have $\boldsymbol{f}(\hat{\boldsymbol{f}})_i \leq h_i(\boldsymbol{x}^i_{lb}, \hat{\boldsymbol{f}}^i_{ub})$. The lower bound in 3.56 can be proved in the same way. □

3.A.8 Sampling Algorithms for Certification

We describe the process of generating samples on the decision boundary in Algorithm 5. The trick is to break down the n-class decision boundary sampling problem to 2 to $(n-1)$-class sampling problems. For instance, to generate samples with k non-zero elements on an n-class decision boundary with density T, one can sample points on an k-class decision boundary with density $T - k$ first, adding 1 to each dimension to make each element non-zero, and assign each element to an n dimensional vector. This operation is denoted by function G in Algorithm 5. G takes two list inputs a and c, increases each element in a by 1, rearranges the elements in a according to the indices given by c, and output a new list w of shape k. Equation 3.62 gives the form of the output of G. If the inputs are $y = 0, a = (3, 2, 3, 0), c = \{2, 3, 7\}$ and $k = 8$ (y is the label, a corresponds to a point on 4-class decision boundary, and c

Algorithm 5 Sample the points on the decision boundary by dynamic programming.

Require: Number of classes K, sample density T, solution set sol with dimension $T \times K$. ▷ Initialize sol.

Initialize each element of sol to be $\emptyset$.
sol$[0][k] = \{\mathbf{0}_k\}$, where $\mathbf{0}_k = [0, .., 0] \in \mathbb{R}^k$.
sol$[j][2] = \{[j/2, j/2]\}$. ▷ Append elements to sol$[j][k]$.
for j from 2 to T **do**
    for k from 3 to K **do**
        for l from 0 to $k-2$ **do**
            if $j - k + l \geq 0$ and $k - l \geq 0$ **then**
                Let C be the set that contains $k-l-1$ combinations of $\{1, 2, ..., k-1\}$.
                sol$[j][k] =$ sol$[j][k] \cup \{G(y, a, c, k) | a \in$ sol$[j - k + l][k - l], c \in C\}$.
            end if
        end for
    end for
end for
return $\tilde{S}_y =$ sol$[T][K]/T$

specifies the non-zero dimension except for the label dimension in an 8 dimensional vector), then the output is $w = (4, 0, 3, 4, 0, 0, 0, 1)$.

$$w_i = \begin{cases} a_0 + 1, & i = y \\ a_{|\{1,2,\cdots,i\} \cap c|} + 1, & i \in c \\ 0. & \text{o/w} \end{cases} \tag{3.62}$$

3.A.9 Experiment Details

Nonlinear control

Baselines. Although there are many baselines in robust control/RL, the settings and goals are not the same as our work (certified robust forward invariance) and thus not directly comparable (Table 3.3). We found the setting and goal in [3] is the most similar to us, and thus we compare with their method in Table 3.1.

Paper	Setting	Goal	Certified or not
Robust MPO [39]	Continuous MDP. model mis-specification	Maximize worst case RL performance.	No
CROP [52]	Discrete MDP. Input state perturbations	Certification of per-state actions and lower bound of cumulative rewards	Yes for action and cumulative reward
Robust MBP [18]	Norm-bounded linear differential inclusions. Disturbance is norm-bounded and added to the dynamics	Train a nonlinear controller that satisfies the exponential stability condition under perturbed dynamics	Yes for stability
Robust FI-ODE (ours)	Continuous nonlinear dynamics. Norm-bounded system parameter perturbations.	Train a nonlinear controller that satisfies the forward invariance (safety) condition under perturbed dynamics	Yes for forward invariance (safety)

Table 3.3: Settings of baseline methods.

Experiment details. The dynamics for the segway system is [24]:

$$\frac{d}{dt}\begin{bmatrix}\phi\\ v\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\dot{\phi}\\ \frac{\cos\phi(-1.8u+11.5v+9.8\sin\phi)-10.9u+68.4v-1.2\dot{\phi}^2\sin\phi}{\cos\phi-24.7}\\ \frac{(9.3u-58.8v)\cos\phi+38.6u-234.5v-\sin\phi(208.3+\dot{\phi}^2\cos\phi)}{\cos^2\phi-24.7}\end{bmatrix} \tag{3.63}$$

All the constants except the acceleration of gravity $g = 9.8$ are system parameters. We enforce robust forward invariance under ±2% perturbation on each of the parameters. We use a 3 layer MLP as the controller and use the Adam optimizer [35]. First, we train the controller to imitate a Linear Quadratic Regulator (LQR) controller. Then we jointly learn the Lyapunov function and the controller. We use adversarial training on $\boldsymbol{o}$ and $\boldsymbol{x}$ to encourage the smoothness of $\boldsymbol{f_\theta}(\boldsymbol{x}, \boldsymbol{o})$ with respect to $\boldsymbol{o}$. Specifically, we find $\boldsymbol{\epsilon}_o$ and $\boldsymbol{\epsilon}_x$ that maximizes $\frac{\partial V}{\partial \boldsymbol{x}}^\top \boldsymbol{f_\theta}(\boldsymbol{x}+\boldsymbol{\epsilon}_x, \boldsymbol{o}+\boldsymbol{\epsilon}_o)$ and train on $\boldsymbol{x}+\boldsymbol{\epsilon}_x$ and $\boldsymbol{o}+\boldsymbol{\epsilon}_o$. We set $\kappa(V(\boldsymbol{x}))$ in Equation (3.6) to be a constant: $\kappa(V(\boldsymbol{x})) = \kappa' \leq \bar{\epsilon} L_V L_f^o$ (κ' is smaller than the requisite lower bound since we train with adversarial inputs and the requisite lower bound is for nominal inputs). We use learning rate of 0.02 for the Lyapunov function and 0.01 for the controller. Next, we jointly learn the controller and finetune the lyapunov function from the previous stage via adversarial training on both the system states and system parameters. We use learning rate of 0.01 for the controller and 0.002 for the lyapunov function.

To certify forward invariance, we use rejection sampling on the state space to cover the boundary of the forward invariant set. The spacing of the ambient grid is set to 0.01 for all 3 dimensions, and the rejection criteria is Equation (3.7). For *robust* forward invariance, we certify in 2 phases. In phase 1, we set the spacing of the ambient grid to be 0.005, and we also sample a grid on the system parameter space to cover the ±2% perturbation range on the parameters with the spacing in Table 3.4.

In phase 2, we sample denser states and system parameters around the states that cannot be certified in phase 1. We set the spacing along each state dimension to be 0.0025, and the spacing of the parameters to be those in brackets in Table 3.4.

Table 3.4: Spacing of the sampled grid on the system parameter space in terms of percentage of each parameter value. Spacing for phase 2 is in the bracket.

Parameter	1.8	11.5	10.9	68.4	1.2	9.3	58.8	38.6	234.5	208.3	24.7
Spacing (%)	4 (4)	4 (4)	4 (4)	1 (1)	4 (4)	4 (4)	2 (1)	1 (1)	1 (0.25)	4 (4)	4 (4)

We run all the control experiments on Intel Core i9 CPU. The certification time for forward invariance is 3.1 seconds, while for robust forward invariance, it is 3285.3 seconds. We also report the training time and the standard deviation of forward invariant rate of each method in Table 3.5.

Table 3.5: Robustness of controllers trained with different training methods. The numbers are the percentage of trajectories that stay within the forward invariant set under the nominal and adversarial system parameters on 1000 adversarially selected initial states. We report the mean and standard deviation over 3 runs. The certificate column indicates whether or not we can certify the (robust) FI property.

Training Method	**Training Time (s)**	**Empirical FI rate (%)**		**Certificate**	
		Nominal Params	**Adv Params**	**FI**	**Robust FI**
Standard Backprop Training	191.3	58.0 ± 1.9	50.4 ± 1.1	✗	✗
Basic Lyapunov Training [28]	1.5	90.2 ± 0.3	52.6 ± 0.6	✗	✗
+ Adaptive Sampling	3.3	100 ± 0.0	68.9 ± 0.3	✓	✗
+ Adversarial Training	67.7	100 ± 0.0	97.8 ± 0.3	✓	✗
+ Both (Ours)	96.7	100 ± 0.0	100 ± 0.0	✓	✓

3.A.10 Image classification

Useful techniques for classification problems. Typically the decision boundary $\{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x}_y = \max_{i \neq y} \boldsymbol{x}_i\}$ is not compact on the logit space ($\boldsymbol{x} \in \mathbb{R}^n$). However, we need the Lyapunov level set to be compact for certification because we can only sample finite points and verify the conditions hold in their neighborhoods. Therefore, we restrict the states to evolve on a probability simplex for classification problems. To do so, we use a Control-Barrier Function based Quadratic Program (CBF-QP) [5], implemented as a differentiable optimization layer [2] in the dynamics (Appendix 3.A.2). However, the linear relaxation based verifier that we use (CROWN) [57] does not support perturbation analysis on differentiable optimization layers such as our CBF-QP layer. Since deriving linear relaxation for CBF-QP is hard, we derive

an interval bound (a special case of linear bounds in CROWN) through CBF-QP (Section 3.A.7).

Experiment settings. For image classification tasks, we use orthogonal layers [46] in the neural network so that the dynamics has 1 Lipschitz constant with respect to both the state and the input. Specifically, we have $\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{o}) = W_3\sigma(W_2\sigma(W_1\boldsymbol{x} + g(\boldsymbol{o})) + b_2) + b_3$, where g is a neural network with 4 orthogonal convolution layers and 3 orthogonal linear layers, and W_1, W_2, W_3 are orthogonal matrices, σ is the ReLU activation function. We set $\kappa(V(\boldsymbol{x})) = \bar{\epsilon} L_V L_f^o V(\boldsymbol{x})$ in the training loss (Equation (3.6)). Note that on the decision boundary, $V(\boldsymbol{x}) = 1$.

In the CBF-QP, we need to pick a class $\mathcal{K}_\infty^e$ function α for the inequality constraint $\mathtt{f} \geq -\alpha(\boldsymbol{x})$. Here we use $\alpha(\boldsymbol{x}) = c_1(e^{c_2\boldsymbol{x}} - 1)$, where $c_1 = 100$, and $c_2 = 0.02$. Comparing with a linear function, this $\alpha(\boldsymbol{x})$ leads to a higher margin over Lipschitz ratio, resulting in better certified accuracy.

During training, we train with batch size of 64. For each image, we sample 512 states. From epoch 1 to 10, all the states are uniformly sampled in the simplex. From epoch 11 to epoch 60, we linearly decay the proportion of uniform sampling in the simplex and increase the portion of uniform sampling within the correct classification set for each class. To sample uniformly in the simplex, we first sample n points from exponential distribution Exp(1) independently, then we normalize the n dimensional vector to have sum 1. To sample uniformly in the correct classification region for each class, we first uniformly sample from the simplex, then we swap the maximum element with the element corresponding to the correct label. We choose κ to be 2.0 in the loss function (Equation (3.6)). We use Adam optimizer with learning rate 0.01, and train for 300 epochs in total.

For certification, we choose $N = 40$ when sampling on the decision boundary. A larger N will lead to better certified accuracy but increases the computational cost dramatically. We ran the experiments on an NVIDIA RTX A6000 GPU.

For baseline methods [41, 12], the adversarial accuracy is evaluated with PGD attack. For our methods, we use AutoAttack [14] to evaluate the empirical adversarial robustness.

Computational cost. The main computational costs of our method comes from the number of samples that are needed to cover the boundary of the forward invariant set. Table 3.6 compares the computational costs and performance for different

Table 3.6: Computational costs for certification on CIFAR-10.

Certification Method	Sampling density (N)	# samples	Time (s)	Certified
Lipschitz	20	3.67×10^5	1.03	0
Lipschitz	30	5.50×10^6	1.37	27.40
Lipschitz	40	4.13×10^7	2.8	33.46
CROWN	40	4.13×10^7	240	42.27

certification methods on CIFAR-10. We first compare the results of certifying with Lipschitz bounds and CROWN [57]. Certifying with Lipschitz bounds is faster. Since we can pre-compute the Lipschitz bound of the dynamics, the certification time equals to the inference time on all the states. Certifying with CROWN provides a tighter bound and thus higher certified accuracy but is more computationally expensive than using the Lipschitz bound. We also compare the performance of different sampling density by choosing different N in Equation (3.8). With larger N, we can cover the region of interest with smaller neighborhood around each sampled point. We vary N using the Lipschitz certification method because it is faster to evaluate, but the pattern should remain the same for CROWN. As expected, we get better accuracy with larger sampling density, but the computational time is longer since we have more samples.

References

[1] Murad Abu-Khalaf, Frank L Lewis, and Jie Huang. "Policy Iterations on the Hamilton–Jacobi–Isaacs Equation for $\mathcal{H}_\infty$ State Feedback Control With Input Saturation". In: *IEEE Transactions on Automatic Control* 51.12 (2006), pp. 1989–1995.

[2] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. "Differentiable Convex Optimization Layers". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[3] Matthias Althoff. "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets". In: *Proceedings of the 16th international conference on hybrid systems: computation and control*. 2013, pp. 173–182.

[4] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[5] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876.

[6] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. "Control Barrier Functions: Theory and Applications". en. In: *2019 18th European Control Conference (ECC)*. Naples, Italy: IEEE, June 2019, pp. 3420–3431. ISBN: 978-3-907144-00-8. DOI: 10.23919/ECC.2019.8796030. URL: https://ieeexplore.ieee.org/document/8796030/ (visited on 04/14/2021).

[7] Anish Athalye, Nicholas Carlini, and David A. Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *International Conference on Machine Learning (ICML)*. 2018.

[8] Tamer Başar and Pierre Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[9] Lucas Böttcher and Thomas Asikis. "Near-optimal control of dynamical systems with neural ordinary differential equations". In: *Machine Learning: Science and Technology* 3.4 (2022), p. 045004.

[10] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, P Kohli, P Torr, and P Mudigonda. "Branch and bound for piecewise linear neural network verification". In: *Journal of Machine Learning Research (JMLR)* (2020).

[11] Ya-Chien Chang, Nima Roohi, and Sicun Gao. "Neural lyapunov control". In: *Advances in neural information processing systems* 32 (2019).

[12] Tong Chen, Jean B Lasserre, Victor Magron, and Edouard Pauwels. "Semialgebraic representation of monotone deep equilibrium models and applications to certification". In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), pp. 27146–27159.

[13] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1310–1320.

[14] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 2206–2216.

[15] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. "Lyapunov-stable neural-network control". In: *Robotics: Science and Systems (RSS)* (2021).

[16] Charles Dawson, Sicun Gao, and Chuchu Fan. "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control". In: *IEEE Transactions on Robotics* (2023).

[17] Alessandro De Palma, Harkirat Singh Behl, Rudy Bunel, Philip H. S. Torr, and M. Pawan Kumar. "Scaling the Convex Barrier with Active Sets". In: *International Conference on Learning Representations (ICLR)* (2021).

[18] Priya L Donti, Melrose Roderick, Mahyar Fazlyab, and J Zico Kolter. "Enforcing robust control guarantees within neural network policies". In: *International Conference on Learning Representations* (2020).

[19] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. "A dual approach to scalable verification of deep networks". In: *Conference on Uncertainty in Artificial Intelligence (UAI)* (2018).

[20] Ruediger Ehlers. "Formal verification of piece-wise linear feed-forward neural networks". In: *International Symposium on Automated Technology for Verification and Analysis (ATVA)*. 2017.

[21] Stefan R Friedrich and Martin Buss. "A robust stability approach to robot reinforcement learning based on a parameterization of stabilizing controllers". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 3365–3372.

[22] Sicun Gao, Soonho Kong, and Edmund M Clarke. "dReal: An SMT solver for nonlinear theories over the reals". In: *International conference on automated deduction*. Springer. 2013, pp. 208–214.

[23] Sophie Grunbacher, Ramin Hasani, Mathias Lechner, Jacek Cyranka, Scott A Smolka, and Radu Grosu. "On the verification of neural odes with stochastic guarantees". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021.

[24] Thomas Gurriet, Andrew Singletary, Jacob Reher, Laurent Ciarletta, Eric Feron, and Aaron Ames. "Towards a framework for realizable safety critical control through active set invariance". In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2018.

[25] Minghao Han, Yuan Tian, Lixian Zhang, Jun Wang, and Wei Pan. "$\mathcal{H}_\infty$ model-free reinforcement learning with robust stability guarantee". In: *CoRR* (2019).

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] Yifei Huang, Yaodong Yu, Hongyang Zhang, Yi Ma, and Yuan Yao. "Adversarial Robustness of Stabilized Neural ODE Might be from Obfuscated Gradients". In: *Mathematical and Scientific Machine Learning*. PMLR. 2022.

[28] Ivan Dario Jimenez Rodriguez, Aaron D. Ames, and Yisong Yue. "LyaNet: A Lyapunov Framework for Training Neural ODEs". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 18687–18703. URL: https://proceedings.mlr.press/v162/rodriguez22a.html.

[29] Ivan Dario Jimenez Rodriguez, Noel Csomay-Shanklin, Yisong Yue, and Aaron D. Ames. "Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies". In: *Proceedings of the 4th Annual Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 1060–1072. URL: https://proceedings.mlr.press/v168/rodriguez22a.html.

[30] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. "Neural certificates for safe control policies". In: *arXiv preprint arXiv:2006.08465* (2020).

[31] Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. "Stable Neural ODE with Lyapunov-Stable Equilibrium Points for Defending Against Adversarial Attacks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[32] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. "Reluplex: An efficient SMT solver for verifying deep neural networks". In: *International Conference on Computer Aided Verification (CAV)*. 2017.

[33] Hassan Khalil. "Nonlinear systems third edition". In: *Patience Hall* (2002).

[34] Islam Khalil, JC Doyle, and K Glover. *Robust and optimal control*. prentice hall, new jersey, 1996.

[35] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR*. 2015.

[36] HaoChih Lin, Baopu Li, Xin Zhou, Jiankun Wang, and Max Q-H Meng. "No need for interactions: Robust model-based imitation learning using neural ode". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 11088–11094.

[37] Diego Manzanas Lopez, Patrick Musau, Nathaniel Hamilton, and Taylor T Johnson. "Reachability Analysis of a General Class of Neural Ordinary Differential Equations". In: *arXiv preprint arXiv:2207.06531* (2022).

[38] Biao Luo, Huai-Ning Wu, and Tingwen Huang. "Off-policy reinforcement learning for $\mathcal{H}_\infty$ control design". In: *IEEE transactions on cybernetics* 45.1 (2014), pp. 65–76.

[39] Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. "Robust reinforcement learning for continuous control with model misspecification". In: *ICLR* (2020).

[40] Mitio Nagumo. "Über die lage der integralkurven gewöhnlicher differentialgleichungen". In: *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series* (1942).

[41] Chirag Pabbaraju, Ezra Winston, and J Zico Kolter. "Estimating Lipschitz constants of monotone deep equilibrium models". In: *International Conference on Learning Representations (ICLR)*. 2020.

[42] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified Defenses against Adversarial Examples". In: *International Conference on Learning Representations*. 2018.

[43] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. "A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[44] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. "An abstract domain for certifying neural networks". In: *Proceedings of the ACM on Programming Languages (POPL)* (2019).

[45] Vincent Tjeng, Kai Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *International Conference on Learning Representations (ICLR)* (2019).

[46] Asher Trockman and J Zico Kolter. "Orthogonalizing convolutional layers with the cayley transform". In: *International Conference on Learning Representations (ICLR)* (2021).

[47] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. "Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[48] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. "Efficient formal safety analysis of neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[49] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[50] Ezra Winston and J Zico Kolter. "Monotone operator equilibrium networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 10718–10728.

[51] Eric Wong and Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5286–5295.

[52] Fan Wu, Linyi Li, Zijian Huang, Yevgeniy Vorobeychik, Ding Zhao, and Bo Li. "Crop: Certifying robust policies for reinforcement learning through functional smoothing". In: *ICLR* (2022).

[53] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Mathias Lechner, and Daniela Rus. “On the Forward Invariance of Neural ODEs”. In: *arXiv preprint arXiv:2210.04763* (2022).

[54] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. “Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[55] Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. “Robustness of control barrier functions for safety critical control”. In: *IFAC-PapersOnLine* (2015).

[56] Hanshu Yan, Jiawei Du, Vincent YF Tan, and Jiashi Feng. “On robustness of neural ordinary differential equations”. In: *International Conference on Learning Representations (ICLR)* (2020).

[57] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. “Efficient neural network robustness certification with general activation functions”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

[58] Kaiqing Zhang, Bin Hu, and Tamer Basar. “Policy Optimization for $\mathcal{H}_2$ Linear Control with $\mathcal{H}_\infty$ Robustness Guarantee: Implicit Regularization and Global Convergence”. In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 179–190.

[59] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*. Vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

Part III

Perception and Learning Dynamics

Chapter 4

KALMAN-IMPLICIT KOOPMAN OPERATOR LEARNING

4.1 Introduction

The preceding chapters developed two approaches for integrating control structure with learning: training on pointwise conditions to guarantee global properties (LyaNet and FI-ODE), and learning inputs to structured controllers (stereo uncertainty for robust CBFs). This chapter pursues the third strategy identified in Chapter 1: enforcing structure architecturally. Rather than training a network to satisfy Lyapunov or barrier conditions, or estimating quantities that a safety-critical controller requires, we build the desired structure directly into the model. Specifically, we enforce that learned latent dynamics are globally linear, corresponding to a finite-dimensional approximation of the Koopman operator.

Linear latent dynamics provide several advantages for control. Model predictive control becomes computationally tractable when the prediction model is linear. The eigenstructure of the learned dynamics matrix reveals interpretable features of the underlying system, such as limit cycles, equilibria, and invariant quantities. These benefits persist regardless of the training outcome because the linearity is guaranteed by construction, not learned from data. This represents a different kind of guarantee than the stability certificates of earlier chapters: architectural enforcement provides structure that cannot be violated, though it does not by itself certify closed-loop behavior.

Koopman operator theory provides the mathematical foundation for this approach. The Koopman operator $\mathcal{K}$ governs the evolution of embedding functions[1] under the flow of a dynamical system, and crucially, $\mathcal{K}$ is linear even when the underlying dynamics are nonlinear [17, 5]. This observation suggests that nonlinear systems can be modeled with linear tools if one works in an appropriately lifted coordinate space. In practice, one seeks finite-dimensional approximations of $\mathcal{K}$ by parameterizing embedding functions via fixed dictionaries of basis functions [6], neural network encoders [23, 30], or hybrid approaches that learn dictionaries [15] (see Fig. 4.1A). These methods have shown promise for prediction and control across robotics and

[1] We use the term "embedding" rather than "observable" to avoid confusion with "measurements" and "observations" in Kalman filtering.

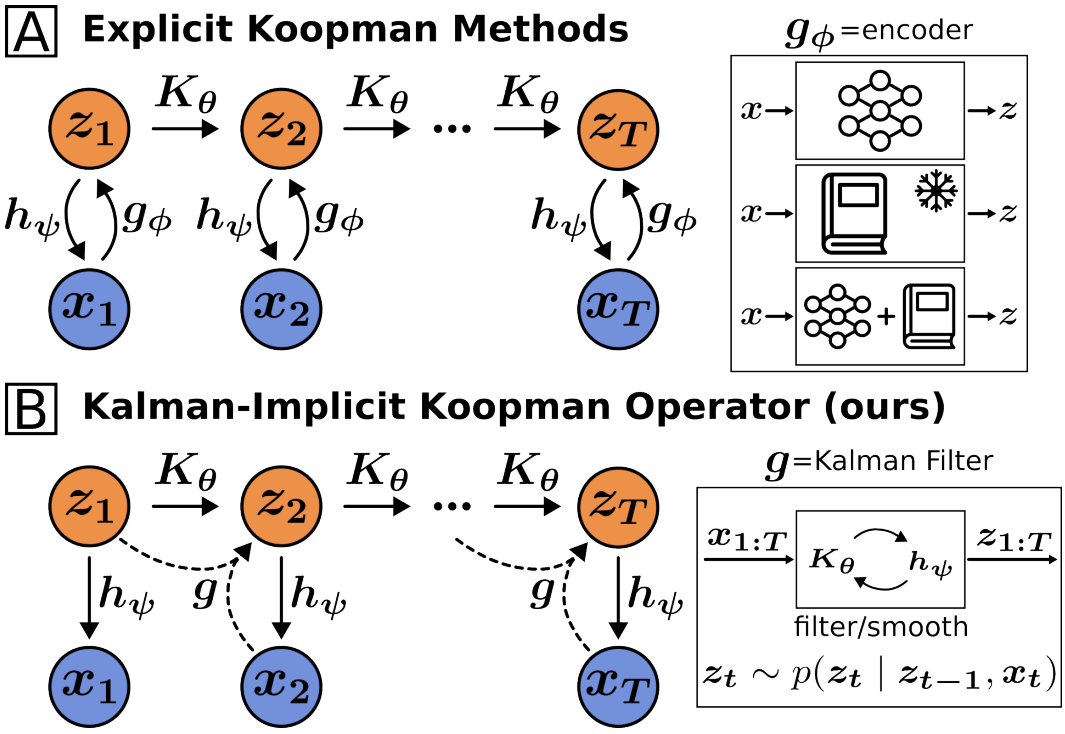


Figure 4.1: **(A)** Most methods for learning the linear Koopman dynamics $\boldsymbol{K}_\theta$ are *explicit*, parameterizing an encoder ϕ_ϕ that maps data $\boldsymbol{x}$ to high-dimensional Koopman embeddings ζ. Examples include neural networks, fixed dictionaries of basis functions, or learnable dictionaries. **(B)** KALIKO (ours) instead *implicitly* learns Koopman embeddings by letting ϕ be a Kalman filter and smoother, which is composed of only two models: the latent dynamics $\boldsymbol{K}_\theta$ and decoder D_ψ. We show that these implicitly-recovered representations are highly interpretable and yield strong open-loop prediction and closed-loop control performance.

related domains [3, 34].

However, selecting embeddings that capture an approximately invariant subspace of the Koopman operator is difficult. Poor choices produce spurious modes or overfitted models [30, 5]. Practitioners counter this with heavy regularization, complex multi-objective losses with carefully tuned weights, and sensitive hyperparameter selection [23, 42, 30]. The challenge stems from the need to simultaneously learn an encoder, a decoder, and a dynamics matrix such that the encoder maps to an invariant subspace. When these components are trained independently or with competing objectives, the resulting embeddings may fail to support accurate long-horizon prediction.

This chapter presents an alternative: ***Kal****man-****I****mplicit* ***K****oopman* ***O****perator* (KALIKO) learning. KALIKO reframes the encoding problem as Bayesian state estima-

tion. Instead of parameterizing an explicit encoder network, we let the Kalman filter and smoother serve as an implicit encoder that infers latent states from observed trajectories (see Fig. 4.1B). The only learned components are the latent dynamics matrix $\boldsymbol{K}_\theta$ and the decoder D_ψ. Because the Kalman filter interleaves prediction updates (which depend on $\boldsymbol{K}_\theta$) with measurement updates (which depend on D_ψ), training with a single reconstruction objective jointly optimizes for both accurate dynamics and faithful decoding. This coupling avoids the need for multi-objective losses and produces embeddings suitable for both reconstruction and prediction.

We validate KALIKO on canonical nonlinear systems and a challenging application domain: predicting the motion of a floating platform under ocean waves and using these predictions for closed-loop control of a crane-mounted payload. This application domain differs from the locomotion and manipulation settings of other thesis chapters, but it provides a rigorous testbed for predictive modeling of high-dimensional PDE-governed phenomena. The methodology itself is general and applies wherever interpretable linear latent dynamics are desirable for downstream control.

Related Work

Computing Koopman Embeddings. A standard data-driven method is *Dynamic Mode Decomposition* (DMD) [31, 5], which fits a least-squares linear map between state snapshots, projecting $\mathcal{K}$ onto the span of embedding coordinates of those snapshots. As DMD operates on raw measurements, it cannot in general realize a nonlinear change of coordinates; *Extended DMD* (EDMD) addresses this by regressing in a lifted space built from a *dictionary* of embedding functions, like polynomials, which must be carefully constructed [39, 5]. Time-delay coordinates provide another practical lifting technique: Delay-DMD constructs Hankel embeddings from past samples to capture approximately Koopman-invariant structure [4]. Yet another method is *Kernel DMD* (KDMD), which implements EDMD *implicitly* by only learning inner products between the embedded data, avoiding explicit enumeration of the dictionary [30, 24]. Lastly, data-driven methods attempt to learn these embeddings, either by learning dictionary elements or by training a deep autoencoder alongside a latent dynamics model [23, 36, 42, 30, 15].

These diverse approaches share canonical failure modes. Even for linear systems, a single spurious basis function can induce severe overfitting [30]. In the data-driven setting, practitioners counter this with heavy regularization [30, 42], complex multi-

term losses [23, 25], and/or highly-sensitive weight initializations and hyperparameters [25]. Conversely, convergence of learned dictionaries is often unclear [42] and poor choices of the above can just as easily trap solutions in bad local minima, leading to underfitting [25, 27].

Kalman Filters and Dynamical Learning. Learnable Kalman filters (KFs) jointly train a latent dynamics model and decoder that maximize the likelihood of data trajectories. The decoder $p(\boldsymbol{x}_t \mid \zeta_t)$ is trained together with the dynamics $p(\zeta_t \mid \zeta_{t-1})$ so that new data points $\boldsymbol{x}_t$ accurately update the posterior belief over the latent state ζ_t. Unlike autoencoder-based approaches that learn an encoder/decoder independently of the dynamics, KFs couple representation and prediction by *interleaving* prediction and measurement updates (Sec. 4.2), which encourages embeddings suitable for both long-horizon prediction and reconstruction.

Prior work has leveraged this insight to learn latent dynamics for image sequences [9, 20] by *compressing* high-dimensional observations into a low-dimensional latent state. In line with Koopman theory, KALIKO instead *lifts* the dimension to find an embedding space where the latent dynamics are approximately linear. Moreover, almost all prior works learn filters for state estimation [11, 18, 16, 7] or simple open-loop prediction tasks [9, 20]. In this work, we show that KALIKO's predictions can also be used in a closed-loop control task.

Kalman Filters and Koopman. Many works have combined KFs with Koopman methods. Most commonly, KFs are used for latent state estimation *after* learning some linear dynamics [35, 43, 14]. Others have used KFs for spectral analysis [22, 44]. Closest to our work are system identification or operator learning methods, but unlike KALIKO, assume the existence of embeddings *a priori* [41] or train an encoder rather than a decoder [10].

Time Series Prediction. The machine learning community has used methods like recurrent neural networks [33] to model sequential data and learn dynamical systems [26], especially those that are hard to model, like continuum robot dynamics [38], human motion [45], and video [12]. More recently, transformer-based models have been applied to many phenomena like weather, traffic patterns, electricity usage, etc. [29]. It has been shown that LLM backbones like GPT2 are effective for time series prediction [46], prompting research on time series tokenization [37].

The recent work "Koopman-Kalman Enhanced Variational Autoencoder" (K2VAE) at first appears similar to KALIKO, but several details distinguish them. First,

K2VAE learns an explicit encoder. Second, K2VAE uses its KF to drive learning error to 0 with a dynamics $\mathbf{A}$ distinct from $\mathcal{K}$, while KALIKO uses the KF directly on the learned operator. Lastly, KALIKO uses several techniques not present in K2VAE that are crucial for performance (see Sec. 4.3 and 4.5). We compare to K2VAE in Sec. 4.5.

Contributions

Our chief contribution is KALIKO, a method that uses the Kalman filter to *implicitly* learn a Koopman embedding space governed by globally linear latent dynamics. Second, we show that KALIKO's embeddings produce faithful reconstructions, are highly interpretable, and align with theoretical expectations. Lastly, on complex wave data generated by a high-dimensional PDE, KALIKO outperforms several baselines on both an open-loop prediction task and a challenging simulated closed-loop stabilization task where it must stabilize an underactuated manipulator's payload by predicting and compensating for wave disturbances.

4.2 Mathematical Preliminaries

Let $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ be the state of a discrete-time nonlinear system which generates the data of interest,

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t), \tag{4.1}$$

and $\zeta \in \mathcal{Z} \subseteq \mathbb{R}^m$ be the state of some corresponding desired linear latent dynamics

$$\zeta_{t+1} = \boldsymbol{K}_\theta \zeta_t, \quad \boldsymbol{x}_t = D_\psi(\zeta_t), \tag{4.2}$$

where θ, ψ are learnable parameters and $D : \mathcal{Z} \to \mathcal{X}$ decodes the latent state back to the data domain. We use "encoder" or "embedding function" to refer to any operation $\phi : \mathcal{X} \to \mathcal{Z}$ mapping data $\boldsymbol{x}$ to latents ζ.

Koopman Theory

We provide a brief overview of Koopman theory, deferring a detailed treatment to [5]. Let $\mathcal{G}(\mathcal{X})$ be a vector space of embedding functions on data $\boldsymbol{x} \in \mathcal{X}$. The *Koopman operator* $\mathcal{K} : \mathcal{G}(\mathcal{X}) \to \mathcal{G}(\mathcal{X})$ acts by composition,

$$(\mathcal{K}\phi)(\boldsymbol{x}) = \phi\big(\boldsymbol{f}(\boldsymbol{x})\big), \quad \phi \in \mathcal{G}(\mathcal{X}), \tag{4.3}$$

and is linear even if $\boldsymbol{f}$ is nonlinear. Since $\mathcal{K}$ is generally infinite-dimensional, in practice we seek finite approximations $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ of $\mathcal{K}$ as in (4.2), and we say ϕ is

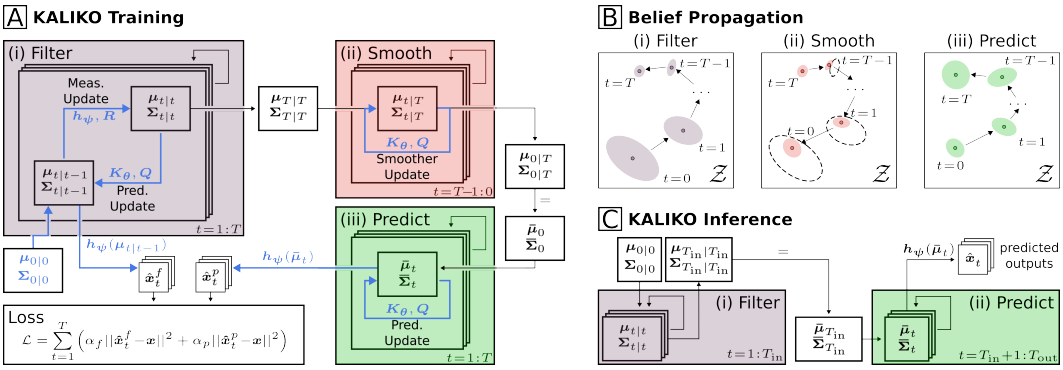


Figure 4.2: **(A) KALIKO Training.** (i) Filter for T steps, outputting a sequence of filtered distributions $\{(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})\}_{t=1}^{T}$ used for the loss. (ii) Run a backward smoother from the final belief $(\boldsymbol{\mu}_{T|T}, \boldsymbol{\Sigma}_{T|T})$, yielding the posterior initial belief $(\boldsymbol{\mu}_{0|T}, \boldsymbol{\Sigma}_{0|T})$. (iii) Roll this belief forward for T steps to get predicted beliefs $\{(\bar{\mu}_t, \bar{\Sigma}_t)\}_{t=1}^{T}$. Filtered and predicted means are decoded to observations and the model is trained end-to-end with a reconstruction loss. Learnable modules/parameters and their operations are highlighted in blue. **(B) Belief Propagation.** The filter's latent belief becomes more certain with more steps. The smoother reuses future evidence to calibrate the initial belief, seeding a strong prediction rollout during training. **(C) KALIKO Inference.** KALIKO filters on T_{in} data points then rolls out its latent dynamics for T_{out} steps. The T_{out} predicted latents are decoded into the final predicted trajectory.

$\mathcal{K}$*-invariant* if the following closure condition is satisfied:

$$\phi(\boldsymbol{f}(\boldsymbol{x})) = \boldsymbol{K}\phi(\boldsymbol{x}), \quad \forall \boldsymbol{x}. \tag{4.4}$$

That is, the latents ζ evolve linearly on the range of ϕ. When (4.4) fails, $\boldsymbol{K}$ introduces *spurious modes* that accumulate multi-step prediction error due to the closure residual [30, 6, 5]. Thus, computing and analyzing invariant embeddings is central to Koopman analysis, which we revisit in Sec. 4.4.

Kalman Filtering

We now review the mechanics of Kalman filters (KFs), deferring details to [32]. KFs recursively update a posterior belief $p(\zeta_t \mid \zeta_{t-1}, \boldsymbol{x}_t)$ over the state given new measurements and its previous state. We assume Gaussian noise with covariance $\boldsymbol{Q}$ corrupts the dynamics such that

$$p(\zeta_t \mid \zeta_{t-1}) = \mathcal{N}(\boldsymbol{K}\zeta_{t-1}, \boldsymbol{Q}), \tag{4.5}$$

and similarly, that Gaussian noise with covariance $\boldsymbol{R}$ corrupts the decoder such that

$$p(\boldsymbol{x}_t \mid \zeta_t) = \mathcal{N}(D(\zeta_t), \boldsymbol{R}). \tag{4.6}$$

Starting from a prior belief over $\zeta_0 \sim \mathcal{N}(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$, the KF iteratively updates its belief by interleaving two updates. First, the *prediction update* propagates the belief at time $t-1$ through the dynamics:

$$\boldsymbol{\mu}_{t|t-1} = \boldsymbol{K}\boldsymbol{\mu}_{t-1|t-1}, \tag{4.7}$$

$$\boldsymbol{\Sigma}_{t|t-1} = \boldsymbol{K}\boldsymbol{\Sigma}_{t-1|t-1}\boldsymbol{K}^\top + \boldsymbol{Q}. \tag{4.8}$$

Second, the *measurement update* adjusts the posterior by accounting for the likelihood of observing $\boldsymbol{x}_t$:

$$\boldsymbol{G}_t = \boldsymbol{\Sigma}_{t|t-1}\boldsymbol{C}_t^\top \left(\boldsymbol{C}_t\boldsymbol{\Sigma}_{t|t-1}\boldsymbol{C}_t^\top + \boldsymbol{R}\right)^{-1}, \tag{4.9}$$

$$\boldsymbol{\mu}_{t|t} = \boldsymbol{\mu}_{t|t-1} + \boldsymbol{G}_t \left(\boldsymbol{x}_t - D(\boldsymbol{\mu}_{t|t-1})\right), \tag{4.10}$$

$$\boldsymbol{\Sigma}_{t|t} = (I - \boldsymbol{G}_t\boldsymbol{C}_t)\boldsymbol{\Sigma}_{t|t-1}, \tag{4.11}$$

where $\boldsymbol{C}_t$ is the decoder's Jacobian, $\boldsymbol{C}_t := \frac{\partial}{\partial \zeta} D(\boldsymbol{\mu}_{t|t-1})$.

The Kalman filter causally updates its belief distribution by observing only past data. However, after observing T data points, it is often useful to update *past* beliefs conditioned on future observations using *Kalman smoothing*. The Rauch-Tung-Striebel smoother caches values from the filtering pass to recursively compute smoothed beliefs from time T to 0:

$$\boldsymbol{G}_t^s = \boldsymbol{\Sigma}_{t|t}\boldsymbol{K}^\top \left(\boldsymbol{\Sigma}_{t+1|t}^\top\right)^{-1}, \tag{4.12}$$

$$\boldsymbol{\mu}_{t|T} = \boldsymbol{\mu}_{t|t} + \boldsymbol{G}_t^s(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t}), \tag{4.13}$$

$$\boldsymbol{\Sigma}_{t|T} = \boldsymbol{\Sigma}_{t|t} + \boldsymbol{G}_t^s(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{t+1|t}) \left(\boldsymbol{G}_t^s\right)^\top. \tag{4.14}$$

In summary, filtering and smoothing require the dynamics $\boldsymbol{K}$, decoder D, prior belief $(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$, dynamics and measurement covariances $\boldsymbol{Q}$ and $\boldsymbol{R}$, and a sequence of data points $\boldsymbol{x}_{1:T}$. In return, they yield filtered and smoothed belief distributions over the corresponding latent states $\zeta_{1:T}$, so we can view the Kalman filter/smoother as an implicit encoder over trajectories, $\phi : \boldsymbol{x}_{1:T} \mapsto \zeta_{1:T}$.

4.3 The Kalman-Implicit Koopman Operator

Our method has two goals: (i) producing high-quality predictions under latent linear dynamics $\boldsymbol{K}$, and (ii) accurately decoding them with $\boldsymbol{x} = D_\psi(\zeta)$. Observe that inaccurate dynamics $\boldsymbol{K}$ degrade the prediction update (4.7)–(4.8), and an inaccurate decoder D_ψ degrades the measurement update (4.9)–(4.11). Thus, a performant KF requires both accurate $\boldsymbol{K}$ and D_ψ. This motivates our method, Kalman-implicit

Koopman operator (KALIKO) learning, which trains a KF end-to-end with a single reconstruction objective to jointly achieve both goals. In contrast, autoencoding Koopman methods typically separate prediction and reconstruction [23, 30, 24], which may overemphasize one at the expense of the other.

Training

KALIKO is trained with the replay overshooting method [20], which consists of three stages (see Fig. 4.2A). Given a data trajectory $\boldsymbol{x}_{1:T}$, the *filtering stage* interleaves the prediction and measurement updates in (4.7)-(4.8) and (4.9)-(4.11) respectively to recover the sequences of belief distributions $\{(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})\}_{t=1}^{T}$ and $\{(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})\}_{t=1}^{T}$. The *smoothing stage* then applies (4.12)-(4.14) to these beliefs in reverse to recover an initial belief at $t = 0$ conditioned on all measurements, $(\boldsymbol{\mu}_{0|T}, \boldsymbol{\Sigma}_{0|T})$. Lastly, the *prediction stage* rolls out T prediction updates (4.7)-(4.8) from the smoothed initial condition *without measurement updates* to recover a sequence of predicted beliefs $\{(\bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t)\}_{t=1}^{T}$.

KALIKO learns $\boldsymbol{K}_\theta$ and D_ψ along with the prior ($\boldsymbol{\mu}_{0|0}$, $\boldsymbol{\Sigma}_{0|0}$) and covariances ($\boldsymbol{Q}$, $\boldsymbol{R}$) end-to-end via the loss

$$\mathcal{L} = \sum_{t=1}^{T} \alpha_f \underbrace{\left\| D_\psi\left(\boldsymbol{\mu}_{t|t-1}\right) - \boldsymbol{x}_t \right\|_2^2}_{\text{filtering}} + \alpha_p \underbrace{\left\| D_\psi\left(\bar{\boldsymbol{\mu}}_t\right) - \boldsymbol{x}_t \right\|_2^2}_{\text{prediction}}, \tag{4.15}$$

where the first term corresponds to the filtered[2] reconstructions of the data and the second to the predicted reconstructions. The coefficients $\alpha_f, \alpha_p \in \mathbb{R}_{>0}$ control the relative weights for filtering and prediction respectively, which we set to 1 in all experiments.

The *filtering loss* is analogous to an autoencoding loss, as it fuses measurement consistency ("decoding") via (4.9)-(4.11) with latent inference via (4.7)-(4.8) ("encoding") [20]. The *prediction loss* ensures accurate open-loop long-horizon predictions, which mirrors inference-time execution. We remark that [20] maximizes the log-likelihood of the data while we found that a simple MSE loss was more computationally-efficient with no performance degradation.

Inference

At inference time, KALIKO uses T_{in} data points to predict the next T_{out} steps. First, the filter consumes $\boldsymbol{x}_{1:T_{\text{in}}}$ to obtain the posterior belief $(\boldsymbol{\mu}_{T_{\text{in}}|T_{\text{in}}}, \boldsymbol{\Sigma}_{T_{\text{in}}|T_{\text{in}}})$. T_{out}

[2] We use the pre-measurement update mean $\boldsymbol{\mu}_{t|t-1}$, as in [20].

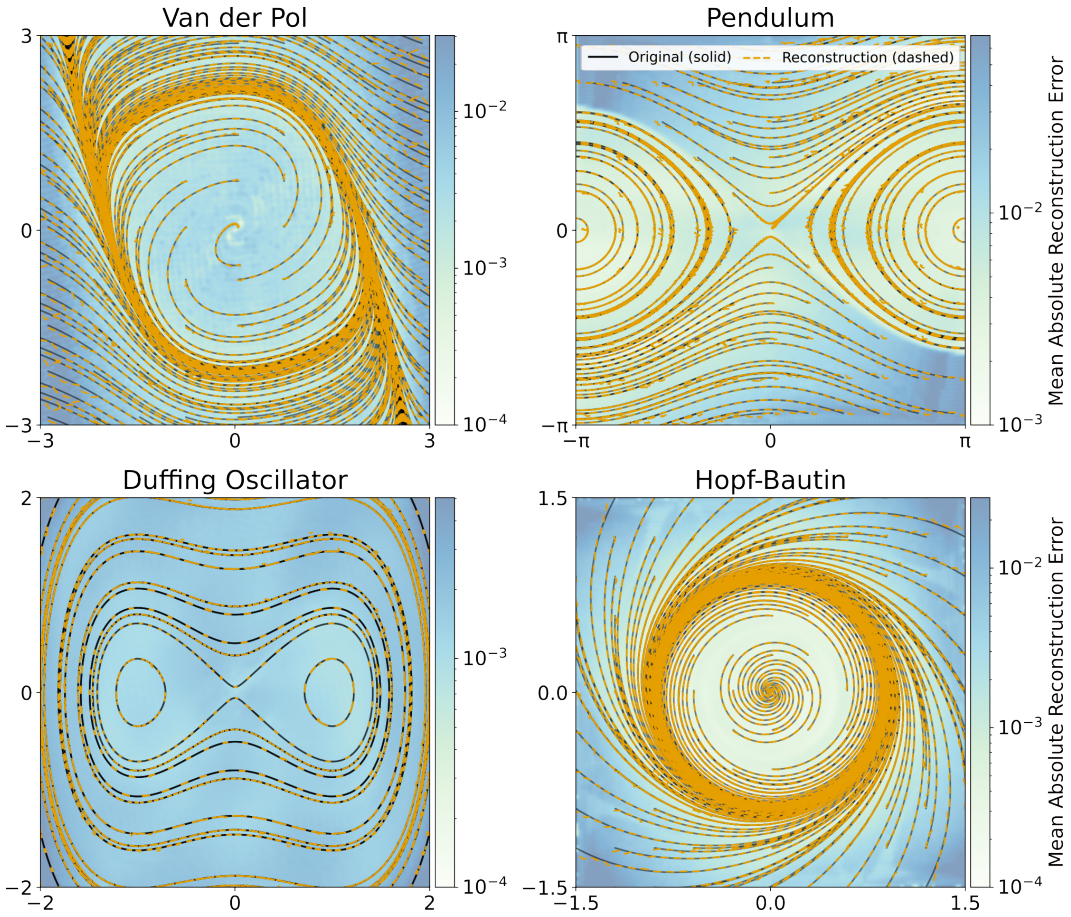


Figure 4.3: **KALIKO Reconstructions.** KALIKO closely reconstructs the vector field of nonlinear systems without an encoder using globally linear latent dynamics. Shown are some ground truth trajectories in black (solid) and their reconstructions in orange (dashed). A denser heatmap of the reconstruction error is shown in blue (darker denotes more error).

prediction steps are then executed from this filtered belief, yielding a sequence of predicted latent beliefs $\{(\bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t)\}_{t=T_{\text{in}}+1}^{T_{\text{in}}+T_{\text{out}}}$. The predicted means are each decoded to obtain a predicted trajectory: $\hat{\boldsymbol{x}}_t = D_\psi(\bar{\boldsymbol{\mu}}_t)$. See Fig. 4.2C for a visualization.

Additional Implementation Details

Delay Embedding. Inspired by the use of delay embeddings in Koopman methods [4], KALIKO parameterizes $\boldsymbol{K}_\theta$ as the dynamics on a delay-embedded latent state,

yielding a sparse block-companion matrix form

$$\underbrace{\begin{bmatrix} \boldsymbol{\zeta}_{t+1} \\ \vdots \\ \boldsymbol{\zeta}_{t+n_z-1} \\ \boldsymbol{\zeta}_{t+n_z} \end{bmatrix}}_{\zeta_{t+1}} = \underbrace{\begin{bmatrix} 0 & I & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & I \\ \Gamma_1 & \Gamma_2 & \dots & \dots & \Gamma_{n_z} \end{bmatrix}}_{\boldsymbol{K}} \underbrace{\begin{bmatrix} \boldsymbol{\zeta}_t \\ \vdots \\ \boldsymbol{\zeta}_{t+n_z-2} \\ \boldsymbol{\zeta}_{t+n_z-1} \end{bmatrix}}_{\zeta_t},$$

where the only learnable parameters are the blocks $\Gamma_1, \dots, \Gamma_{n_z}$, and each $\boldsymbol{\zeta}_t$ is a sub-latent state of dimension d_z, which is stacked into the full delay-embedded latent state $\zeta_t = (\boldsymbol{\zeta}_t, \dots, \boldsymbol{\zeta}_{t+n_z-1}) \in \mathbb{R}^{n_z \cdot d_z}$.

Temporal Chunking. KALIKO temporally chunks the full-resolution data into groups of n_x states, each corresponding to a sub-latent vector as follows:

$$\boldsymbol{\zeta}_t \leftrightarrow (\boldsymbol{\xi}_\tau, \dots, \boldsymbol{\xi}_{\tau+n_x-1}), \tag{4.16}$$

where τ is the original time index satisfying $t = \lfloor \tau / n_x \rfloor$ and $\boldsymbol{\xi}_\tau$ is a raw data point. Thus, D decodes n_z sub-latent vectors into a contiguous sequence of $n_z \cdot n_x$ raw data points:

$$D(\underbrace{\boldsymbol{\zeta}_{t:(t+n_z-1)}}_{\zeta_t}) = \underbrace{\boldsymbol{\xi}_{\tau:(\tau+n_x \cdot n_z-1)}}_{\boldsymbol{x}_t}. \tag{4.17}$$

Temporal chunking greatly speeds up training, as it reduces the number of steps in each training stage by a factor of n_x, until the cost of memory and linear system solves associated with a large data dimension n outweighs the speedup.

Decoder Design. The decoder D_ψ is a convolutional neural network that interleaves depthwise convolutions across the delay dimension (mixing $\boldsymbol{\zeta}$'s) with GELU-activated MLPs, which allows D to mix sub-latent vectors across many time steps to decode long sequences of raw observations. We found two such blocks sufficient for all experiments.

4.4 Analyzing KALIKO Embeddings

Reconstruction Performance

As discussed in Sec. 4.2, the Kalman filter/smoother can be viewed as an encoder on sequences of data $\boldsymbol{x}_{1:T}$. We study KALIKO's autoencoding capabilities on four systems for visualization purposes: the Van der Pol oscillator, pendulum, Duffing oscillator, and a two-limit cycle Hopf-Bautin system.

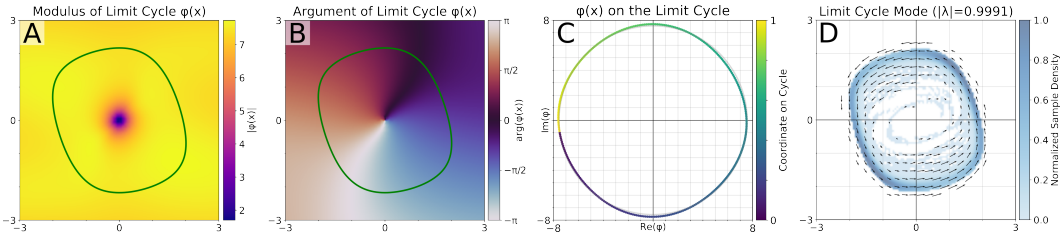


Figure 4.4: **Limit Cycle Eigenfunction for the VDP System. (A/B)** Without parameterizing an encoder, KALIKO implicitly recovers an eigenfunction associated with the Van der Pol system's limit cycle (overlaid in green) with $|\lambda| \approx 1$. **(C)** Evaluating $\varphi(x)$ along the limit cycle traces out a nearly perfect circle in the complex plane, showing that KALIKO entirely captures the limit cycle dynamics into this eigenfunction. **(D)** The Koopman mode associated with this eigenvalue corresponds to a vector field that drives trajectories onto the cycle.

Van der Pol	$\begin{cases} \dot{x}_1 = x_1 - x_1^3/3 - x_2 \\ \dot{x}_2 = x_1 \end{cases}$	(4.18)
Pendulum	$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \sin(x_1) \end{cases}$	(4.19)
Duffing	$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 - \delta\, x_2 - x_1^3 \end{cases}$	(4.20)
Hopf-Bautin (Polar)	$\begin{cases} \dot{r} = r e^{(1-2r^2)/2}\,(r^2 - r^4 - 3/16) \\ \dot{\theta} = 1 \end{cases}$	(4.21)

Table 4.1: Systems used for embedding analysis. We learn discrete-time dynamics despite presenting these systems in continuous time for clarity.

We plot reconstructed trajectories on a heatmap of reconstruction errors (Fig. 4.3). KALIKO accurately reconstructs the original data (MAE $< 10^{-2}$), verifying that the Kalman filter and smooth function as an effective encoder ϕ. Moreover, all models used identical hyperparameters, demonstrating KALIKO's robustness to features of nonlinear systems like fixed points, limit cycles, and multiple attractors.

Analyzing KALIKO's Eigenfunctions

Sec. 4.2 introduced a key property, $\mathcal{K}$-invariance, in (4.4). Observe that the *eigenfunctions* φ of $\mathcal{K}$ satisfy (4.4), since

$$\lambda\varphi(\boldsymbol{x}_t) = (\mathcal{K}\varphi)\,(\boldsymbol{x}_t), \quad \lambda \in \mathbb{C}, \tag{4.22}$$

thus giving coordinates on which $\mathcal{K}$ acts linearly, making them central objects of study in the literature [5]. A left eigenpair $(\lambda, \mathbf{w})$ of the approximation $\boldsymbol{K}$ of $\mathcal{K}$ (and

ϕ) implies an eigenfunction $\varphi(\boldsymbol{x}) := \mathbf{w}^\top \phi(\boldsymbol{x})$, because

$$\begin{aligned}(\mathcal{K}\varphi)(\boldsymbol{x}_t) &= \varphi(\boldsymbol{f}(\boldsymbol{x}_t)) = \varphi(\boldsymbol{x}_{t+1}) = \mathbf{w}^\top \phi(\boldsymbol{x}_{t+1}) \\ &= \mathbf{w}^\top \boldsymbol{K}\phi(\boldsymbol{x}_t) = \lambda \mathbf{w}^\top \phi(\boldsymbol{x}_t) = \lambda \varphi(\boldsymbol{x}_t).\end{aligned}$$

Thus, we examine the eigenfunctions of $\boldsymbol{K}_\theta$ via eigendecomposition, allowing us to interpret the dynamics captured by KALIKO's implicit encoder, the Kalman filter and smoother. We consider two case studies.

Van der Pol. The Van der Pol (VDP) system (4.18) has a globally-stable limit cycle (Fig. 4.3, top left). We identify a complex eigenpair $(\lambda, \mathbf{w})$ of $\boldsymbol{K}_\theta$ with $|\lambda| \approx 1$, implying a steady-state oscillatory latent mode. We approximate the eigenfunction $\varphi(\boldsymbol{x})$ by evaluating $\mathbf{w}^\top \phi(\boldsymbol{x})$ over latent beliefs in the plane, plotting VDP's limit cycle (green) against the modulus and argument of $\varphi(\boldsymbol{x})$ in Fig. 4.4A/B.

We can now check whether the limit cycle is described by $\varphi(\boldsymbol{x})$. First, we parameterize the (unit-modulus) eigenvalue as $\lambda = e^{i\omega}$ and let ζ_0 be an arbitrary fixed latent element of the corresponding eigenspace. Observe that the corresponding latent trajectory must have constant modulus:

$$|\zeta_t| = |\lambda^t| \cdot |\zeta_0| = |e^{it\omega}| \cdot |\zeta_0| = |\zeta_0|.$$

Moreover, the argument along the latent trajectory must evolve linearly (mod 2π):

$$\arg(\zeta_t) = \arg(\lambda^t \zeta_0) = t\omega + \arg(\zeta_0) \pmod{2\pi}.$$

Thus, if $\varphi(\boldsymbol{x})$ evaluated on the limit cycle traces out a single revolution of a circle in the complex plane, we can conclude that KALIKO *encodes the limit cycle dynamics entirely in this mode*, which is confirmed in Fig. 4.3C.

We also examine the time-invariant directions in $\mathcal{X}$ given by the associated *right eigenvector* of $\boldsymbol{K}_\theta$, or the *Koopman mode* [5]. To do so, we encode a set of trajectories, project them onto the limit cycle eigenspace, and decode them, resulting in Fig. 4.4D. The Koopman mode shows the attraction of the limit cycle in a neighborhood around it, verifying that KALIKO has learned VDP's dominant dynamical behavior.

Duffing Oscillators. The undamped Duffing oscillator (UDO, $\delta = 0$ in (4.20)) has a saddle at the origin and two neutrally stable wells at $(\pm 1, 0)$. Because the oscillation frequency depends on energy (i.e., continuous spectrum [23]), no finite embedding space can recover a global phase eigenfunction; instead, invariants (e.g., energy) correspond to $\lambda \approx 1$. Consistent with this, KALIKO recovers an eigenfunction

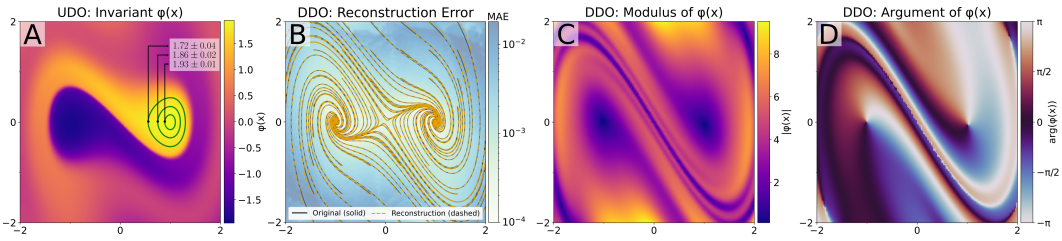


Figure 4.5: **Eigenfunctions for the Undamped and Damped Duffing Oscillator. (A)** KALIKO recovers an eigenfunction $\varphi(\boldsymbol{x})$ with $\lambda \approx +1$ capturing the invariance of systems with continuous spectra like the *undamped Duffing oscillator* (UDO). The value of $\varphi(x)$ is nearly constant along each of three distinct cycles (green) with little variation (shown are mean and stdev). Across cycles, the value clearly changes, corresponding to energy invariance. **(B)** Conversely, KALIKO also reconstructs the *damped Duffing oscillator* (DDO) with no adjustments. **(C/D)** KALIKO recovers an eigenfunction capturing the attractive "spiral" behavior about the wells at $(\pm 1, 0)$, reproducing the results from explicit methods in prior work [30, Fig. 5].

φ that is nearly constant along each closed orbit and varies across energy levels (Fig. 4.5A), capturing UDO's energy invariance. Adding damping (DDO, $\delta = 1$) eliminates the continuous spectrum, leaving the saddle at the origin and making $(\pm 1, 0)$ asymptotically stable. Here, KALIKO identifies a contractive mode with $|\lambda| < 1$ whose eigenfunction φ tends to 0 at the three equilibria and whose phase clearly depicts the attractive spirals (Fig. 4.5B–D), in line with Koopman analyses in prior work [30, Fig. 5].

Takeaway. Despite using an *implicit* encoder, KALIKO reproduces the expected Koopman structure on these canonical systems (unit-modulus phase on the VDP limit cycle, energy-like invariants for UDO, and contractive modes near DDO equilibria) using *identical hyperparameters* with no system-specific tuning. These results illustrate that KALIKO's implicit encoding does not preclude typical Koopman analysis; rather, it facilitates it by stably recovering expressive, interpretable representations without sensitive handcrafted features or model tuning.

4.5 Experiments

Simulated Ocean Data

We now study the predictive capacity of KALIKO and its application to a closed-loop stabilization control problem. To do so, we simulate the motion of a barge in various ocean conditions, yielding thousands of trajectories in $SE(3)$. The data are generated using `capytaine` [1], which solves a linear-flow potential PDE in the

frequency domain using the boundary element method. The irregular sea state is generated from the JONSWAP spectrum [13] by randomly sampling the significant wave height, peak period, and peak enhancement. For a set of discrete frequencies, we solve radiation and diffraction problems to assemble the hydrodynamic load on the moving body, which we then convert to a time series of motion by inverse Fourier transform. All parameter ranges were chosen such that the computed solutions were numerically stable, yielding realistic solutions. For more detail on hydrodynamics, we refer the reader to [28].

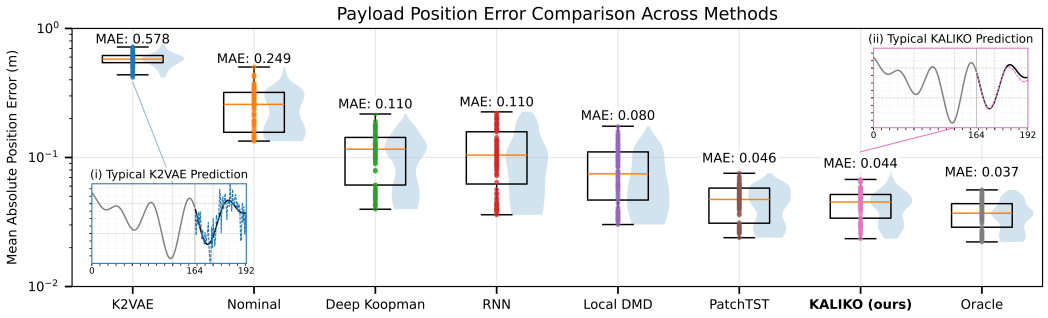


Figure 4.6: **Control Results.** KALIKO outperforms all baselines on the control task with the least variance and near-oracle performance. Insets show one dimension of a hold-out trajectory. Gray denotes input context, black denotes ground truth prediction data. (i) K2VAE's poor performance can be explained by its overly-noisy predictions (blue, dashed). (ii) KALIKO's smooth prediction trajectories enable much more stable closed-loop control (pink, dashed).

Baselines and Methodological Details

We compare KALIKO to both Koopman-specific and general time series prediction baselines.

Koopman Baselines. We have 3 Koopman baselines. The *Deep Koopman net* [23] is an autoencoding-based method that has the ability to parameterize continuous eigenvalue spectra. It is trained for long-horizon predictions with a multistep prediction consistency loss along with autoencoding losses, giving a canonical baseline for Koopman autoencoders. The *Koopman-Kalman Variational Autoencoder* (K2VAE) [40] claims state-of-the-art performance in probabilistic time series prediction. K2VAE predicts a "residual" global Koopman operator $\boldsymbol{K}_{\text{glo}}$ added to a locally-fitted operator derived from DMD, $\boldsymbol{K}_{\text{loc}}$, yielding the model $\boldsymbol{K} = \boldsymbol{K}_{\text{loc}} + \boldsymbol{K}_{\text{glo}}$. It uses an auxiliary Kalman filter to estimate and drive errors to 0. Since K2VAE assumes that $\boldsymbol{K}_{\text{loc}}$ is a good prior about which to learn a residual, we implement a

simple baseline where we fit a *local DMD* model at each time step on the context T_{in} using delay embeddings as the Koopman embeddings. Several works have explored variations of this idea in an MPC-style context for modeling effects like fluid flows [2, 21]. We used the open-source implementations and settings of the baselines for fair comparison.

Time Series Baselines. We have 2 general time series baselines. *Recurrent neural networks* (RNNs) [33] are classic autoregressive models well-suited for dynamical prediction. RNNs can be viewed as "generalized" filters, making them a natural baseline: they ingest data, update an internal state, and can generate prediction sequences. *PatchTST* [29] is a transformer that consumes data as temporal "patches", inspired by vision transformers [8]. PatchTST is a popular method for general time series modeling, as it requires little domain-specific knowledge to yield reasonable predictions.

Data Preprocessing. For all methods, we map all poses in $SE(3)$ to $\mathfrak{se}(3)$, the tangent space at the identity, using the Log map. This makes the data Euclidean and enables geometrically consistent pose predictions. Second, we normalize the data with a running estimate of the mean and standard deviation along each axis, which is updated during training. At inference time, we fix these statistics to prevent leakage.

Prediction Performance

In *open-loop prediction*, each model is supplied an input context of $T_{\text{in}} = 128$ data points and predicts $T_{\text{out}} = 64$ steps. If applicable, methods are trained for 164k steps with a batch size of 32, and we report prediction errors in $\mathfrak{se}(3)$ averaged over all times, dimensions, and batches.

Method	MSE (↓)	MAE (↓)
Deep Koopman	0.0019	0.025
K2VAE	0.0015	0.026
Local DMD	0.0005	0.010
RNN	0.0003	0.010
PatchTST	0.0004	0.011
KALIKO (ours)	**0.0002**	**0.008**

Table 4.2: Performance on open-loop wave prediction.

In Table 4.2, KALIKO outperforms all methods using a global linear latent prediction model. In contrast, the RNN and PatchTST strongly leverage nonlinear models,

and K2VAE and local DMD rely heavily on locally-fitted models rather than a global linear operator. Surprisingly, local DMD outperforms both Deep Koopman and K2VAE, which we attribute to the sensitivity of autoencoder-based Koopman methods to their learned embeddings, as noted in Sec. 4.1, and K2VAE's noisy predictions (see Fig. 4.6).

Ablations

Delay Length. KALIKO uses a delay length $n_z = 4$ for its latent dynamics. We perform two ablations: letting $n_z = 1$ (no delay) and increasing to $n_z = 6$. We find that the delay is crucial for performance, as the MAE nearly doubles without it. Conversely, using a higher delay does not improve performance, justifying the choice $n_z = 4$.

Decoder Architecture. We replaced KALIKO's convolutional decoder with a simple MLP that performs no depthwise mixing, instead mapping each sub-latent vector ζ_t to a distinct temporal chunk. We find this only slightly harms performance, showing KALIKO's robustness to architectural design decisions.

Learnable Priors. We evaluate KALIKO's performance when fixing the prior to $(0, I)$. This substantially degrades performance, which we hypothesize is because the scale and anisotropy of the prior are key for allowing KALIKO to learn the most expressive possible embedding space geometry.

Ablation	MSE (↓)	MAE (↓)
$n_z = 4 \rightarrow n_z = 1$	0.0007	0.015
$n_z = 4 \rightarrow n_z = 6$	0.0002	0.008
Conv → MLP Decoder	0.0003	0.009
Learnable → Fixed Latent Prior	0.0005	0.012
KALIKO (Original)	**0.0002**	**0.008**

Table 4.3: Ablations: effect on KALIKO's prediction performance.

Control Performance

We evaluate all methods on a simulated control task: stabilizing a cable-suspended payload with a barge-mounted crane (see Fig. 4.7). The task is challenging because the barge and crane are severely underactuated: the free-body motion is driven by the ocean waves, and the crane's payload height is controllable only via a damped winch. We need not model any control dynamics, since the crane is controlled with sampling-based MPC, which selects controls by choosing high-performance

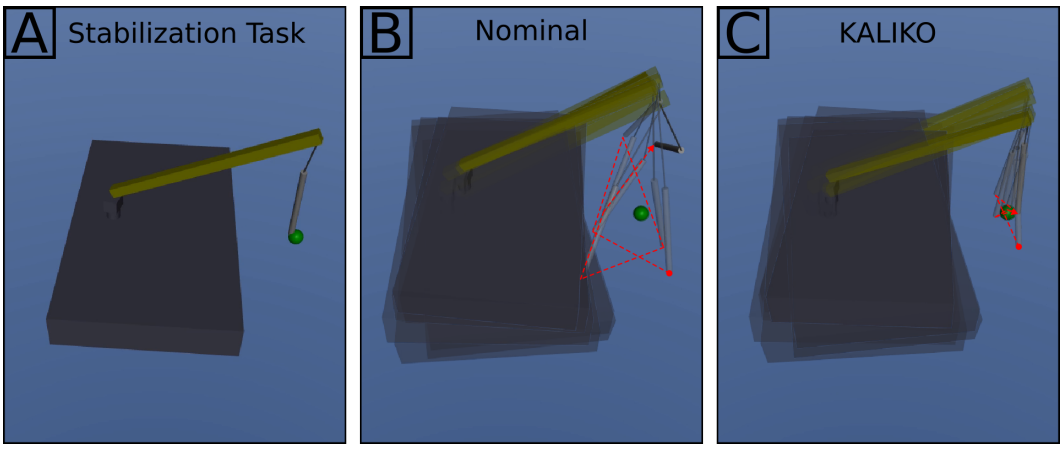


Figure 4.7: **(A)** We aim to stabilize the tip of the payload to a goal location (green) by compensating for motion induced by realistic waves. A crane controls the payload by rotating its base and boom and/or reeling its winch. **(B)** The nominal controller poorly regulates the payload, as seen in this 3 second snapshot (red dashed lines). **(C)** KALIKO stabilizes the payload at near-oracle levels in the presence of the same waves.

candidates under forward simulations of the system. Thus, the predictive models in Sec. 4.5 suffice.

We evaluate all methods on 100 random wave simulations, feeding their predictions to an MPPI controller implemented using the open-source toolbox `judo` [19]. We calibrate performance against two other methods: an **oracle** supplied with ground-truth future barge motion, and a **nominal** stabilizing controller that ignores the wave disturbances. Since the controller is stochastic, the oracle allows us to measure the error induced by its sampling noise. We report mean absolute error in the payload target position (see Fig. 4.6).

KALIKO outperforms all methods with near-oracle performance and the tightest variance, showing that its superior open-loop prediction translates to high-performance closed-loop control. However, prediction performance is not a perfect indicator of control performance, as evidenced by local DMD outperforming the RNN. We attribute this gap to the role of state estimation, which is as crucial as prediction in control. Trained explicitly for filtering, KALIKO maintains an accurate latent belief in closed loop, whereas autoencoding-based Koopman methods and the RNN are trained only for prediction, and their internal states drift. Conversely, local DMD and PatchTST, which are not latent-variable methods, perform well because they operate directly on (patches of) raw data; with reasonably accurate

short-horizon predictions and fast enough feedback, the controller can effectively stabilize the system.

4.6 Discussion

KALIKO uses Kalman filtering to implicitly learn Koopman embeddings without parameterizing an explicit encoder. The linear latent dynamics $\boldsymbol{K}_\theta$ are enforced by construction, and training with a single reconstruction objective jointly optimizes dynamics and decoding. On canonical nonlinear systems, KALIKO recovers eigenfunctions consistent with theoretical expectations. On wave prediction and crane stabilization, KALIKO outperforms all baselines with near-oracle control performance.

Within the thesis, KALIKO exemplifies the third strategy from Chapter 1: enforcing structure architecturally. Unlike the pointwise-condition training of LyaNet and FI-ODE, which must verify Lyapunov or barrier conditions after training, KALIKO's linearity holds by design. The tradeoff is that linearity in latent space does not by itself certify closed-loop stability or safety. The eigenstructure of $\boldsymbol{K}_\theta$ provides interpretability (limit cycles, decay rates, invariants), but translating these into guarantees requires decoding back to the original state space where the controller operates.

Several limitations constrain KALIKO's applicability to the agile robotic systems emphasized elsewhere in this thesis. KALIKO assumes smooth dynamics and cannot represent hybrid systems with impacts or mode switches. The zero dynamics policies of Part IV handle precisely such systems; KALIKO does not. Systems with continuous frequency spectra (Section 4.4) admit only approximate finite-dimensional Koopman representations, and prediction accuracy degrades on long horizons. KALIKO also assumes clean state observations, unlike the stereo chapter which explicitly models perception uncertainty.

The wave domain was chosen because quasi-periodic PDE-governed dynamics are well-suited to linear lifting, providing a rigorous testbed for the methodology. Extending these ideas to contact-rich locomotion and manipulation, where hybrid dynamics and perception uncertainty are unavoidable, remains open. Combining Koopman-based disturbance prediction with the safety-critical control structures developed in earlier chapters is a natural direction for future work.

References

[1] Matthieu Ancellin and Frédéric Dias. "Capytaine: a Python-based linear potential flow solver". In: *Journal of Open Source Software* 4.36 (Apr. 2019), p. 1341. DOI: 10.21105/joss.01341. URL: https://doi.org/10.21105%2Fjoss.01341.

[2] Anqi Bao, Eduardo Gildin, Abhinav Narasingam, and Joseph S. Kwon. "Data-Driven Model Reduction for Coupled Flow and Geomechanics Based on DMD Methods". In: *Fluids* 4.3 (2019). ISSN: 2311-5521. DOI: 10.3390/fluids4030138. URL: https://www.mdpi.com/2311-5521/4/3/138.

[3] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. "Machine learning for fluid mechanics". In: *Annual review of fluid mechanics* 52.1 (2020), pp. 477–508.

[4] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. "Chaos as an intermittently forced linear system". In: *Nature Communications* 8.1 (May 2017). ISSN: 2041-1723. DOI: 10.1038/s41467-017-00030-8. URL: http://dx.doi.org/10.1038/s41467-017-00030-8.

[5] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. "Modern Koopman Theory for Dynamical Systems". In: *SIAM Review* 64.2 (2022), pp. 229–340. DOI: 10.1137/21M1401243. eprint: https://doi.org/10.1137/21M1401243. URL: https://doi.org/10.1137/21M1401243.

[6] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1517384113. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1517384113.

[7] Adrien Corenflos, James Thornton, George Deligiannidis, and Arnaud Doucet. *Differentiable Particle Filtering via Entropy-Regularized Optimal Transport*. 2021. arXiv: 2102.07850 [stat.ML]. URL: https://arxiv.org/abs/2102.07850.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[9] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. *A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning*. 2017. arXiv: 1710.05741 [stat.ML]. URL: https://arxiv.org/abs/1710.05741.

[10] Zi Cong Guo, Vassili Korotkine, James R. Forbes, and Timothy D. Barfoot. *Koopman Linearization for Data-Driven Batch State Estimation of Control-Affine Systems*. 2021. arXiv: 2109.07000 [cs.RO]. URL: https://arxiv.org/abs/2109.07000.

[11] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. *Backprop KF: Learning Discriminative Deterministic State Estimators*. 2017. arXiv: 1605.07148 [cs.LG]. URL: https://arxiv.org/abs/1605.07148.

[12] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. *Learning Latent Dynamics for Planning from Pixels*. 2019. arXiv: 1811.04551 [cs.LG]. URL: https://arxiv.org/abs/1811.04551.

[13] Klaus F. Hasselmann, Tim P. Barnett, Evert Bouws, H. C. Carlson, David Edgar Cartwright, Kurt Enke, James Alfred Ewing, Hans Gienapp, Dieter E. Hasselmann, P. Kruseman, Arend Meerburg, Peter M. Müller, Dirk J. Olbers, Karl Richter, Wolfgang Sell, and Hans Walden. "Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP)". In: 1973. URL: https://api.semanticscholar.org/CorpusID:122820060.

[14] Peng Huang, Ketong Zheng, and Gerhard Fettweis. "Data-Driven Koopman Operator-Based Error-State Kalman Filter for Enhanced State Estimation of Quadrotors in Agile Flight". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 10983–10989. DOI: 10.1109/IROS58592.2024.10802457.

[15] Yuhong Jin, Lei Hou, and Shun Zhong. "Extended Dynamic Mode Decomposition with Invertible Dictionary Learning". In: *Neural Networks* 173 (2024), p. 106177. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2024.106177. URL: https://www.sciencedirect.com/science/article/pii/S0893608024001011.

[16] Alina Kloss, Georg Martius, and Jeannette Bohg. "How to train your differentiable filter". In: *Autonomous Robots* 45.4 (May 2021), pp. 561–578. ISSN: 1573-7527. DOI: 10.1007/s10514-021-09990-9. URL: http://dx.doi.org/10.1007/s10514-021-09990-9.

[17] Bernard O Koopman. "Hamiltonian systems and transformation in Hilbert space". In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318.

[18] Michelle A. Lee, Brent Yi, Roberto Martín-Martín, Silvio Savarese, and Jeannette Bohg. *Multimodal Sensor Fusion with Differentiable Filters*. 2020. arXiv: 2010.13021 [cs.RO]. URL: https://arxiv.org/abs/2010.13021.

[19] Albert H. Li, Brandon Hung, Aaron D. Ames, Jiuguang Wang, Simon Le Cleac'h, and Preston Culbertson. *Judo: A User-Friendly Open-Source Package for Sampling-Based Model Predictive Control*. 2025. arXiv: 2506.17184 [cs.RO]. URL: https://arxiv.org/abs/2506.17184.

[20] Albert H. Li, Philipp Wu, and Monroe Kennedy. "Replay Overshooting: Learning Stochastic Latent Dynamics with the Extended Kalman Filter". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 852–858. DOI: 10.1109/ICRA48506.2021.9560811.

[21] Qiuqi Li, Chang Liu, and Yifei Yang. *Localized Dynamic Mode Decomposition with Temporally Adaptive Partitioning*. 2025. arXiv: 2503.13093 [math.NA]. URL: https://arxiv.org/abs/2503.13093.

[22] Ningxin Liu, Shuigen Liu, Xin T. Tong, and Lijian Jiang. *Estimate of Koopman modes and eigenvalues with Kalman Filter*. 2024. arXiv: 2410.02815 [eess.SY]. URL: https://arxiv.org/abs/2410.02815.

[23] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. "Deep learning for universal linear embeddings of nonlinear dynamics". In: *Nature Communications* 9.1 (Nov. 2018). ISSN: 2041-1723. DOI: 10.1038/s41467-018-07210-0. URL: http://dx.doi.org/10.1038/s41467-018-07210-0.

[24] Yuhuang Meng, Jianguo Huang, and Yue Qiu. "Koopman operator learning using invertible neural networks". In: *Journal of Computational Physics* 501 (Mar. 2024), p. 112795. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2024.112795. URL: http://dx.doi.org/10.1016/j.jcp.2024.112795.

[25] Jack W. Miller, Charles O'Neill, Navid C. Constantinou, and Omri Azencot. *Eigenvalue initialisation and regularisation for Koopman autoencoders*. 2022. arXiv: 2212.12086 [cs.LG]. URL: https://arxiv.org/abs/2212.12086.

[26] Nima Mohajerin and Steven L. Waslander. *Multi-Step Prediction of Dynamic Systems with Recurrent Neural Networks*. 2018. arXiv: 1806.00526 [cs.NE]. URL: https://arxiv.org/abs/1806.00526.

[27] Indranil Nayak, Ananda Chakrabarti, Mrinal Kumar, Fernando L. Teixeira, and Debdipta Goswami. "Temporally-consistent koopman autoencoders for forecasting dynamical systems". In: *Scientific Reports* 15.1 (July 2025). ISSN: 2045-2322. DOI: 10.1038/s41598-025-05222-7. URL: http://dx.doi.org/10.1038/s41598-025-05222-7.

[28] John Nicholas Newman. *Marine Hydrodynamics*. Cambridge, MA: MIT Press, 1977.

[29] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023. arXiv: 2211.14730 [cs.LG]. URL: https://arxiv.org/abs/2211.14730.

[30] Samuel E. Otto and Clarence W. Rowley. *Linearly-Recurrent Autoencoder Networks for Learning Dynamics*. 2019. arXiv: 1712.01378 [math.DS]. URL: https://arxiv.org/abs/1712.01378.

[31] Clarence W. Rowley, Igor Mezic, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. "Spectral analysis of nonlinear flows". In: *Journal of Fluid Mechanics* 641 (2009), pp. 115–127. DOI: 10.1017/S0022112009992059.

[32] Simo Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013.

[33] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: http://dx.doi.org/10.1016/j.physd.2019.132306.

[34] Lu Shi, Masih Haseli, Giorgos Mamakoukas, Daniel Bruder, Ian Abraham, Todd Murphey, Jorge Cortes, and Konstantinos Karydis. *Koopman Operators in Robot Learning*. 2025. arXiv: 2408.04200 [cs.RO]. URL: https://arxiv.org/abs/2408.04200.

[35] Amit Surana and Andrzej Banaszuk. "Linear observer synthesis for nonlinear systems using Koopman Operator framework". In: *IFAC-PapersOnLine* 49.18 (2016). 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016, pp. 716–723. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.10.250. URL: https://www.sciencedirect.com/science/article/pii/S2405896316318304.

[36] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. *Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition*. 2018. arXiv: 1710.04340 [cs.LG]. URL: https://arxiv.org/abs/1710.04340.

[37] Sabera Talukder, Yisong Yue, and Georgia Gkioxari. *TOTEM: TOkenized Time Series EMbeddings for General Time Series Analysis*. 2025. arXiv: 2402.16412 [cs.LG]. URL: https://arxiv.org/abs/2402.16412.

[38] Abbas Tariverdi, Venkatasubramanian Kalpathy Venkiteswaran, Michiel Richter, Ole Elle, Jim Tørresen, Kim Mathiassen, Sarthak Misra, and Ørjan Martinsen. "A Recurrent Neural-Network-Based Real-Time Dynamic Model for Soft Continuum Manipulators". In: *Frontiers in Robotics and AI* 8 (Mar. 2021), p. 631303. DOI: 10.3389/frobt.2021.631303.

[39] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition". In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346.

[40] Xingjian Wu, Xiangfei Qiu, Hongfan Gao, Jilin Hu, Bin Yang, and Chenjuan Guo. *K^2VAE: A Koopman-Kalman Enhanced Variational AutoEncoder for Probabilistic Time Series Forecasting*. 2025. arXiv: 2505.23017 [cs.LG]. URL: https://arxiv.org/abs/2505.23017.

[41] Guan Xiaoqing, You Wang, Xiaomeng Kang, Wei Yao, Jin Zhang, and Guang Li. "An Online System Identification Algorithm for Spherical Robot Using the Koopman Theory". In: *IEEE Robotics and Automation Letters* PP (Jan. 2025), pp. 1–8. DOI: 10.1109/LRA.2025.3552997.

[42] Enoch Yeung, Soumya Kundu, and Nathan Hodas. "Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems". In: *2019 American Control Conference (ACC)*. 2019, pp. 4832–4839. DOI: 10.23919/ACC.2019.8815339.

[43] Lingyun Zeng, S.M.Hadi Sadati, and Christos Bergeles. "Koopman Operator-based Extended Kalman Filter for Cosserat Rod Wrench Estimation". In: *2023 International Symposium on Medical Robotics (ISMR)*. 2023, pp. 1–7. DOI: 10.1109/ISMR57123.2023.10130210.

[44] Zhexuan Zeng, Jun Zhou, Yasen Wang, and Zuowei Ping. *Koopman Spectral Analysis from Noisy Measurements based on Bayesian Learning and Kalman Smoothing*. 2025. arXiv: 2410.00703 [eess.SY]. URL: https://arxiv.org/abs/2410.00703.

[45] Jianjing Zhang, Hongyi Liu, Qing Chang, Lihui Wang, and Robert X. Gao. "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly". In: *CIRP Annals* 69.1 (2020), pp. 9–12. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2020.04.077. URL: https://www.sciencedirect.com/science/article/pii/S0007850620300998.

[46] Tian Zhou, PeiSong Niu, Xue Wang, Liang Sun, and Rong Jin. *One Fits All:Power General Time Series Analysis by Pretrained LM*. 2023. arXiv: 2302.11939 [cs.LG]. URL: https://arxiv.org/abs/2302.11939.

Chapter 5

SELF-SUPERVISED ONLINE LEARNING FOR SAFETY-CRITICAL CONTROL USING STEREO VISION

5.1 Introduction

The preceding chapters developed methods for training neural networks to satisfy pointwise conditions that imply global stability and safety guarantees. These approaches assume accurate knowledge of the system state. In practice, robotic systems must operate with imperfect perception, and the gap between measured and true state can violate the assumptions underlying control-theoretic guarantees. This chapter addresses a complementary problem: how to learn the perception uncertainty that safety-critical controllers require as input.

Control Barrier Functions (CBFs) [3, 12] provide a principled framework for guaranteeing safety through the satisfaction of a Lyapunov-like condition. CBFs have enabled safety-critical behaviors including obstacle avoidance [23], multi-agent navigation [16], and safe walking [9]. Robust extensions of CBF theory can accommodate bounded state uncertainty [11, 8], but these methods require knowledge of the uncertainty bounds. The CBF framework thus exemplifies the second strategy for structure-aware learning identified in this thesis: the control structure provides guarantees, but learning must supply the quantities those guarantees depend upon.

Vision-based perception presents a particular challenge for this approach. Stereo vision is widely used in robotics for depth estimation and is an integral component of systems such as simultaneous localization and mapping (SLAM) [6]. The errors generated by stereo algorithms depend on scene texture, lighting, and geometry in ways that are difficult to characterize analytically. Supervised methods can learn mappings from images to uncertainty estimates [10, 27, 9, 21, 7], but these approaches require ground-truth training data that may be unavailable in deployment environments. A model trained on indoor scenes with accessible ground truth often fails to generalize to outdoor environments where the safety-critical application actually operates.

This chapter develops a self-supervised method for learning stereo vision uncertainty that can adapt online to novel environments without ground-truth depth measurements. The key insight is that a multibaseline stereo system provides geometric

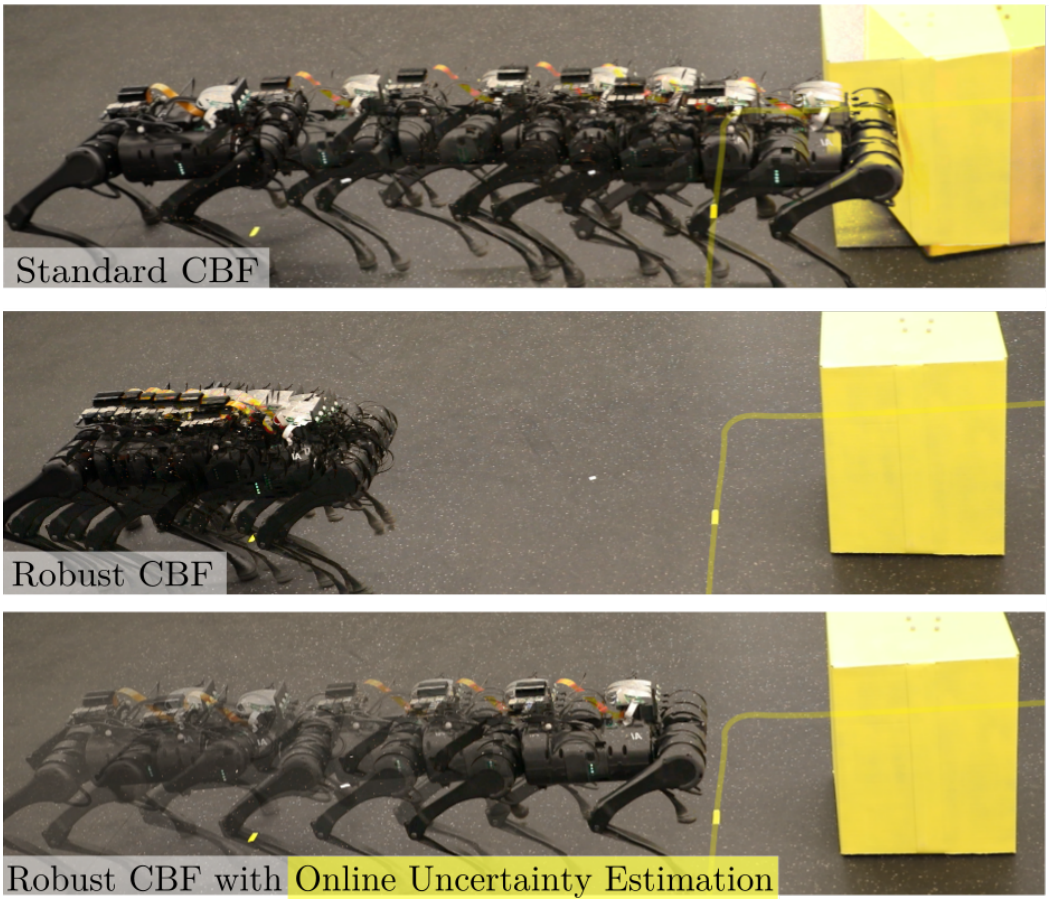


Figure 5.1: These space-time images display our quadrupedal robot throughout the course of an experiment. The robot is considered safe if it remains left of the yellow line. The standard control barrier function (CBF) condition fails to keep the robot safe due to errors in stereo vision; the robust CBF condition keeps the robot safe, but is conservative; and our proposed method, Robust CBFs with Online Uncertainty Estimation, keeps the robot safe without remaining overly conservative.

redundancy: disparities computed from different camera pairs must satisfy consistency constraints, and deviations from these constraints reveal the stereo algorithm's errors. By learning a mapping from image appearance to error distributions, we obtain uncertainty estimates that can serve as inputs to robust CBF controllers. Online learning frameworks have shown success in a variety of robotic applications [17, 14, 24, 18], and we frame our uncertainty adaptation in this setting.

We demonstrate this approach on obstacle avoidance for a quadrupedal robot (Fig. 5.1). The quadruped platform provides a challenging testbed: the robot moves through varied environments where stereo accuracy changes substantially, and the safety constraint (avoiding collision) admits no tolerance for systematic

underestimation of uncertainty. Our experiments show that online adaptation of the uncertainty model enables safety without excessive conservatism, whereas fixed uncertainty estimates either fail to prevent collisions or stop the robot far from obstacles. A visualization of the complete method can be found in Fig. 5.2.

The contributions of this chapter are three-fold. First, we present an online, self-supervised method for characterizing the uncertainty of disparity errors in novel environments (Section 5.2). Second, we develop a robustified CBF formulation that incorporates learned uncertainty estimates for vision-based obstacle avoidance (Section 5.3). Third, we validate the complete system on hardware, demonstrating that learned uncertainty can serve as the missing input to formal safety frameworks (Section 5.4).

5.2 Stereo Vision Uncertainty Quantification

We begin by revisiting stereo vision-based depth estimation. We then propose an approach for learning the uncertainty of a black box stereo-matching algorithm. The proposed self-supervised learning approach can be trained online and takes advantage of geometric structure in stereo disparity maps so as not to require ground truth data.

Background in Stereo Vision

Stereo vision is a popular tool for determining depth from images. These methods compute a *disparity*: the shift observed in an object's projection onto two camera planes. Using a geometric understanding of the camera setup, pixel-based disparity maps can be converted to depth maps. Errors in the final depth-map result from a combination of pixel-mismatch in disparity estimation and error in the camera parameters used to convert from disparity to depth. The errors in the intrinsic and extrinsic parameters of the camera are usually small and their effect on the resulting depth distribution is easy to compute. On the other hand, pixel matching errors are much larger and are the result of a much more complicated stereo matching procedure whose effect on the resulting disparity is difficult to quantify and environment-dependent.

For standard stereo vision we adopt the model of [22] for two cameras (left and right) and assume that they are perfectly rectified, vertically aligned and evenly spaced with known distance $b \in \mathbb{R}_{>0}$ between each camera. Pixel coordinates within an image are given by the tuple $p \triangleq (u, v) \in K$, where $K \triangleq \{0, \dots, W\} \times \{0, \dots, H\}$ for image width $W \in \mathbb{N}_{>0}$ and image height $H \in \mathbb{N}_{>0}$.

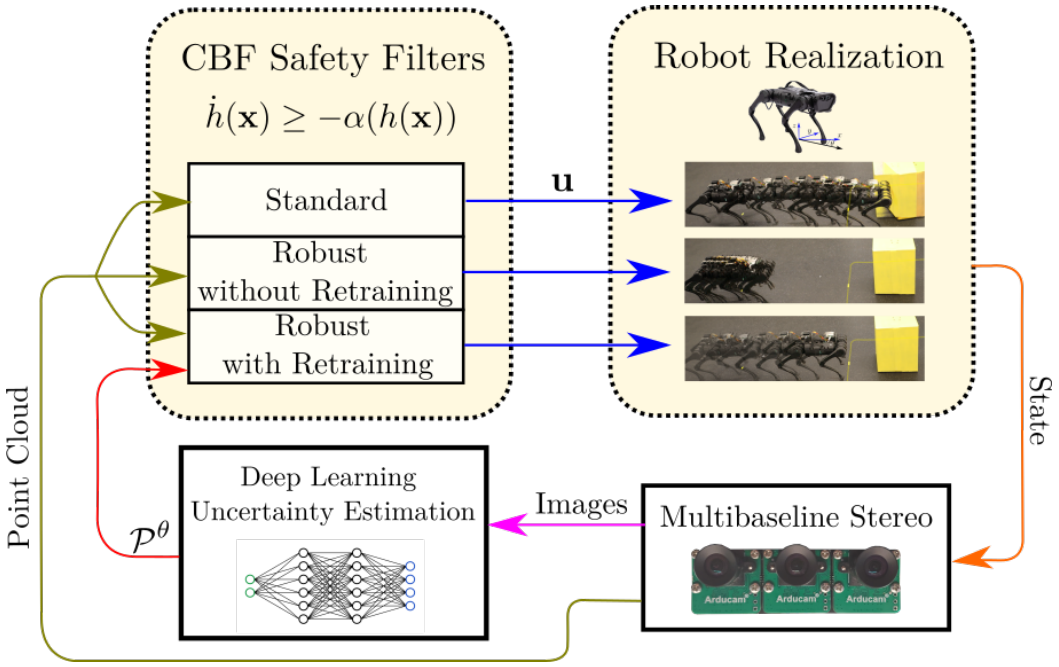


Figure 5.2: The overarching structure of our approach. It begins on the bottom right by capturing three time-synchronized images that are then fed into an uncertainty estimation pipeline and also used to generate a 3D point cloud. There are three possible CBF filters that result in the three robot realizations shown. From top to bottom, the standard filter only takes into account the noisy point-cloud in avoiding obstacles. The robust CBF safety filters use the estimate of the uncertainty $\mathcal{P}$ to compensate for noise in the point cloud. Finally, the "Robust with Retraining" filter refines the model of uncertainty to the current environment in real-time.

Stereo algorithms such as Block Matching, Semi-Global Block Matching, and Efficient Large-Scale Stereo [13] compute disparities by determining the discrete pixel distance between matching regions of two images. Since the disparity represents a shift between pixels of two images, the measured disparity $\widehat{d}$ must be a finite integer value. Assuming that the true disparity d is a finite integer implies that the error $e \triangleq \widehat{d} - d$ must also be a finite integer. Prior work has been done to interpolate disparities for non-integer subpixel accuracy [25]; however, we restrict our attention to integer disparity values to highlight the error in pixel-matching.

Self-supervised Error Estimation

To learn the error in disparity, we introduce a three-camera multibaseline stereo system which produces multiple disparity maps that are related through simple functions; deviations from the ideal relationship indicate error in the estimated disparities. By analyzing the correlation of image appearance with these errors,

a function that estimates disparity error from appearance is learned and used to specify state error-bounds in real-time for use in a robustified CBF.

We introduce a three-element camera system, whose central camera is assumed to be perfectly rectified and vertically aligned with the other two cameras as shown in Fig. 5.2. This third camera is placed between the left and right cameras such that it has a baseline of $b/2$ with both. The three cameras produce a time-synchronized grayscale image triple (I_1, I_2, I_3) where $I_i \in \mathbb{N}^{W\times H}$ for $i \in 1, 2, 3$ and $1, 2, 3$ correspond with left, center, and right, respectively. The disparity between any image pair (I_i, I_j) for $i < j$ is obtained using the stereo-vision algorithm $\mathcal{D} : \mathbb{N}^{W\times H} \times \mathbb{N}^{W\times H} \to \Gamma^{W\times H}$, so that $\widehat{d}_{i,j} = \mathcal{D}(I_i, I_j)$. Here, $\Gamma \subset \mathbb{N}_{\geq 0}$ is the set of possible disparity values.

Given the measurement $\widehat{d}_{i,j}$, the error appears as $\widehat{d}_{i,j} = d_{i,j} + e_{i,j}$ with error distribution $e_{i,j} \sim \mathcal{P}(I_i, I_j)$ and ground truth disparity $d_{i,j} \in \Gamma^{W\times H}$. We model this error as a discrete random variable with probability $\mathcal{P}(I_i, I_j)$ on $\Gamma^{W\times H}$. This model of disparity errors contrasts sharply with other common error models, such as punctual observation, uniform observation, and Gaussian observation [22], in that it accounts for the discrete nature of stereo-pixel matching algorithms. If ground-truth knowledge of $d_{i,j}$ is obtainable, then supervised learning methods can be implemented to directly estimate this error term. However, it is often the case that ground-truth knowledge is unavailable; particularly when a domain transfer must occur during operation. Thus we seek a general method to estimate $e_{i,j}$ for any black-box disparity algorithm without the need for ground-truth data.

We leverage the known geometric relationships between the three cameras to learn a mapping between image appearance and disparity error distribution that can adapt during operation in new environments. Given a multibaseline stereo system, if one ignores occlusions, it is possible to completely reconstruct each disparity map from the other two maps. The relationship to reconstruct $\widehat{d}_{1,3}$ from $\widehat{d}_{1,2}$ and $\widehat{d}_{2,3}$ is shown in Algorithm 6; we denote this reconstruction as $\overline{d}_{1,3} \triangleq \widehat{d}_{1,2} \oplus \widehat{d}_{2,3}$.

We use the reconstructed disparity $\overline{d}_{1,3}$ to learn the parameters θ of a function $\mathcal{P}^\theta$ that approximates error distribution $\mathcal{P}$ (refer to Algorithm 7). Since this method does not require ground truth information, Algorithm 7 can be run online during operation to adapt $\mathcal{P}^\theta$ to new visual environments. Recall that the disparity error, $e_{1,3}$ is discrete in nature. Therefore, the pixel-wise reconstruction error $re(p) \triangleq \|\widehat{d}^p_{1,3} - \overline{d}^p_{1,3}\|_1$ will also be discrete. For this reason, optimizing the loss L reduces to a pixel-wise classification problem similar to image segmentation. Thus, as is done in image

Algorithm 6 Disparity Reconstruction: $\overline{d}_{1,3} = \widehat{d}_{1,2} \oplus \widehat{d}_{2,3}$

1: $\overline{d}_{1,3} \leftarrow \mathbf{0}_{H \times W}$
2: **for** $v \in [1, ..., H]$ **do**
3: **for** $u \in [1, ..., W]$ **do**
4: $\widehat{u} \leftarrow n + \widehat{d}_{1,2}(u, v)$
5: $\overline{d}_{1,3}(u, v) \leftarrow \widehat{d}_{1,2}(u, v) + \widehat{d}_{2,3}(u, \widehat{v})$
6: **end for**
7: **end for**

segmentation, we use pixel-wise cross entropy as the loss function L. This method is shown in Algorithm 7. In Algorithm 7, for each pixel p of the disparity $\widehat{d}_{1,3}$ the

Algorithm 7 Self-Supervised Stereo Error Estimation Adaptation

1: $L \leftarrow 0$
2: **while** robot is running **do**
3: $(I_1, I_2, I_3) \leftarrow$ Capture Current Frame
4: $\widehat{d}_{1,2} \leftarrow \mathcal{D}(I_1, I_2)$
5: $\widehat{d}_{2,3} \leftarrow \mathcal{D}(I_2, I_3)$
6: $\widehat{d}_{1,3} \leftarrow \mathcal{D}(I_1, I_3)$
7: $\overline{d}_{1,3} \leftarrow \widehat{d}_{1,2} \oplus \widehat{d}_{2,3}$
8: $re(p) \leftarrow \left| \widehat{d}^{p}_{1,3} - \overline{d}^{p}_{1,3} \right|$
9: $L \leftarrow -\frac{1}{H \times W} \sum_p \mathbb{E}_{\mathbb{1}(re(p))}[\log \mathcal{P}^{\theta}(I_i, I_k)]$
10: $\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial L}{\partial \theta}$
11: **end while**

corresponding reconstruction error is computed. The loss function in Algorithm 7 is then equivalent to the expected negative log likelihood of each pixel under the proposed model $\mathcal{P}^{\theta}$. An example visualization of lines 3 – 8 can be found in Fig. 5.3. Although this algorithm focuses on the reconstructed disparity $\overline{d}_{1,3}$, it can be easily extended to similar reconstructions of $d_{1,2}$ and $d_{2,3}$.

Supervised methods have been used in the past to estimate uncertainty in robotic applications by computing the covariance of state estimates [19]. Our approach differs in that we do not require ground truth and we take advantage of the discrete structure of images to learn a discrete, rather than a Gaussian, distribution, which is better suited to the disparity measurements of stereo vision.

5.3 Safe Vision-Based Control

In this section we review Control Barrier Functions (CBFs) [2] as a tool for guaranteeing the safety of dynamical systems. We then propose CBFs that rely on the

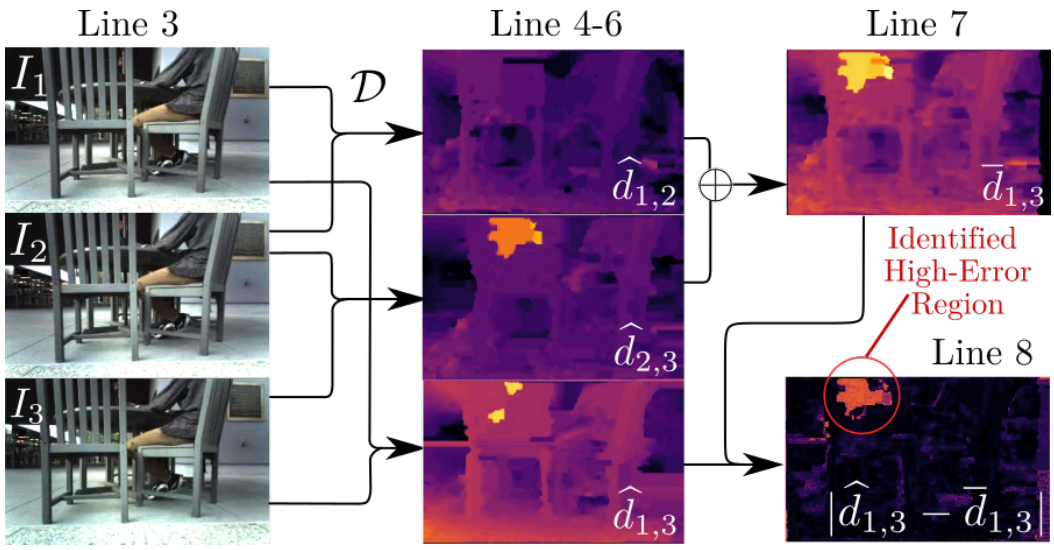


Figure 5.3: Lines 3-8 of Algorithm 7 illustrated from left to right. Starting from three time-synchronized images three pairwise disparities are computed as shown in the middle column. Two of these disparities are used to build a reconstruction of the third disparity shown in the top right which can then be used to estimate the pixel-wise error of the stereo algorithm shown in the bottom right image. These steps of the algorithm correctly identify that the back of the closest chair is a high-error region without using ground truth information. This information is used to learn a correspondence between visual features and error distributions.

position of pixels provided by stereoscopic sensing. Finally, we incorporate the proposed self-supervised error estimates of Section 5.2 to enforce robust safety.

Control Barrier Functions

First we give a brief introduction to CBFs which follows our description in [8], where additional technical details can be found. In this work we consider the safety of robotic systems with control affine dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}, \quad \mathbf{x} \in \mathbb{R}^n, \ \mathbf{u} \in \mathbb{R}^m, \tag{5.1}$$

where $\mathbf{x}$ is the state of the system, $\mathbf{u}$ is the input, $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is the drift dynamics, and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the input matrix. We assume that $\mathbf{f}$ and $\mathbf{g}$ are locally Lipschitz continuous. Given a locally Lipschitz continuous state-feedback controller $\mathbf{k} : \mathbb{R}^n \to \mathbb{R}^m$, the closed-loop dynamics are governed by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{k}(\mathbf{x}). \tag{5.2}$$

For any initial condition $\mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^n$ there exists a unique solution $\mathbf{x}(t)$ to (5.2), which we assume to exist $\forall t \in [0, \infty)$.

The notion of safety is formalized by defining a *safe set* $C \subset \mathbb{R}^n$ in the state space that the system must remain within. In particular, consider the set C as the 0-superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$:

$$C \triangleq \{\mathbf{x} \in \mathbb{R}^n \mid h(\mathbf{x}) \geq 0\}, \tag{5.3}$$

where $h(\mathbf{x}) = 0 \implies \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}) \neq 0$ and C is non-empty and has no isolated points. Safety is defined as the *forward invariance* of C, i.e., if $\mathbf{x}_0 \in C$, then $\mathbf{x}(t) \in C$ for all $t \geq 0$.

To synthesize controllers that ensure safety, we use Control Barrier Functions (CBFs) [3] defined as follows:

Definition 12 (*Control Barrier Function (CBF)*). *Let $\mathsf{C} \subset \mathbb{R}^n$ be a safe set given by (5.3). The function h is a Control Barrier Function (CBF) for* (5.1) *on C if there exists $\gamma \in \mathcal{K}_{\infty,e}$[1] such that for all $\mathbf{x} \in \mathsf{C}$:*

$$\sup_{\mathbf{u}\in\mathbb{R}^m} \dot{h}(\mathbf{x}, \mathbf{u}) \triangleq \underbrace{\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x})\mathbf{f}(\mathbf{x})}_{L_\mathbf{f}h(\mathbf{x})} + \underbrace{\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x})\mathbf{g}(\mathbf{x})}_{L_\mathbf{g}h(\mathbf{x})} \mathbf{u} \geq -\gamma(h(\mathbf{x})), \tag{5.4}$$

where $L_\mathbf{f}h : \mathbb{R}^n \to \mathbb{R}$ and $L_\mathbf{g}h : \mathbb{R}^n \to \mathbb{R}^m$ are the Lie derivatives of h with respect to $\mathbf{f}$ and $\mathbf{g}$ respectively.

A main result in [2, 28] relates CBFs to the safety of the closed-loop system (5.2) with respect to C:

Theorem 8. *Given a safe set $\mathsf{C} \subset \mathbb{R}^n$, if h is a CBF for* (5.1) *on C, then any locally Lipschitz continuous controller $\mathbf{k} : \mathbb{R}^n \to \mathbb{R}^m$ satisfying*

$$L_\mathbf{f}h(\mathbf{x}) + L_\mathbf{g}h(\mathbf{x})\mathbf{k}(\mathbf{x}) \geq -\gamma(h(\mathbf{x})) \tag{5.5}$$

for all $\mathbf{x} \in \mathsf{C}$, renders the system (5.2) *safe w.r.t. C.*

Given a nominal (but not necessarily safe) locally Lipschitz continuous controller $\mathbf{k}_d : \mathbb{R}^n \to \mathbb{R}^m$ and a CBF h, the CBF-Quadratic Program (CBF-QP) [3] is a controller that guarantees the system's safety:

$$\begin{aligned} \mathbf{k}(\mathbf{x}) = \ & \underset{\mathbf{u}\in\mathbb{R}^m}{\operatorname{argmin}} \quad \frac{1}{2}\|\mathbf{u} - \mathbf{k}_d(\mathbf{x})\|_2^2 \\ & \text{s.t.} \qquad L_\mathbf{f}h(\mathbf{x}) + L_\mathbf{g}h(\mathbf{x})\mathbf{u} \geq -\gamma(h(\mathbf{x})). \end{aligned} \tag{CBF-QP}$$

[1] $\mathcal{K}_{\infty,e}$ denotes the set of extended class-$\mathcal{K}$ infinity functions, wherein $\gamma \in \mathcal{K}_{\infty,e}$ satisfies $\gamma : \mathbb{R} \to \mathbb{R}$ is strictly increasing, $\gamma(0) = 0$, and $\lim_{r\to\infty} \gamma(r) = \infty$, $\lim_{r\to-\infty} \gamma(r) = -\infty$.

Control Barrier Functions for Safe Vision-Based Control

Next we apply CBFs to achieve safe obstacle avoidance for robotic systems based on stereo vision. First we construct CBFs for safe vision-based control. Let $\rho_p \in \mathbb{R}^3$ represent the true three-dimensional position of the portion of the scene which generated pixel p. Using this, we can define a CBF $h : \mathbb{R}^n \times \mathbb{R}^3 \to \mathbb{R}$ that relies on both the state $\mathbf{x}$ and three dimensional pixel position ρ_p. The pixel position is a geometric function of the true disparity, $\rho_p = T(\mathbf{x}, r(p, d^p_{1,3}))$ where $r : \mathbb{N}^2 \times \mathbb{N}$ is the stereo reprojection function and $T : \mathbb{R}^n \times \mathbb{R}^3 \to \mathbb{R}^3$ is the transformation mapping from the robot's state and relative pixel position to pixel position.

In order to relate the output of the stereoscopic sensor with safety, we make the following assumptions:

Assumption 1. The environment is static, so the time derivative of the pixelized environment is zero: $\frac{d\rho_p}{dt} = 0$ for all $p \in K$.

Assumption 2. Ensuring safety with respect to the true three dimensional pixel locations is sufficient to ensure safety with respect to the environment. That is, the safe set for the system is given by:

$$C_K = \{\mathbf{x} \in \mathbb{R}^n \mid h(\mathbf{x}, \rho_p) \geq 0, \ \forall p \in K\} \tag{5.6}$$

where $h : \mathbb{R}^n \times \mathbb{R}^3 \to \mathbb{R}$ is the CBF for pixel p.

Although it is not outlined in this work, Assumption 1 can be relaxed to include moving environments by calculating $\dot{h}$ accordingly and estimating the motion of obstacles in the environment. Assumption 2 simplifies the surrounding environment from infinite- to finite-dimensional by assuming that the environment is smooth between a sufficiently dense coverage of pixels. It also implies that the system only has to stay safe with respect to objects that can be seen in the cameras' field of view.[2]

Given Assumptions 1 and 2 and Theorem 8, synthesizing the control input $\mathbf{u}$ such that

$$L_{\mathbf{f}} h(\mathbf{x}, \rho_p) + L_{\mathbf{g}} h(\mathbf{x}, \rho_p)\mathbf{u} \geq -\gamma(h(\mathbf{x}, \rho_p)), \tag{5.7}$$

$\forall p \in K$, is sufficient to guarantee safety. Considering each pixel $p \in K$, however, may be computationally intractible, therefore we seek a condition with fewer required constraints.

[2]The field of view aspect of Assumption 2 can be overcome by tracking features that leave the frame as done in Simultaneous Localization and Mapping (SLAM) algorithms [6].

To combine the constraints, we apply Boolean composition to each CBF h to produce a single nonsmooth CBF h_{ns},

$$h_{\text{ns}}(\mathbf{x}) \triangleq \min_{p \in K} h(\mathbf{x}, \rho_p), \tag{5.8}$$

and simply enforce the CBF constraint associated with the pixels whose CBFs have the smallest value [15]. In particular, to achieve safety it is sufficient to enforce only the constraints whose indices appear in the locally-encapsulating index set:

$$\Lambda = \{p \in K : h(\mathbf{x}, \rho_p) \leq h_{\text{ns}}(\mathbf{x}) + \delta\}, \tag{5.9}$$

for some $\delta > 0$, as stated formally below.

Theorem 9 ([15], Prop III.6). *Let $h : \mathbb{R}^n \times \mathbb{R}^3 \to \mathbb{R}^3$ be a locally Lipschitz function and h_{ns} be as in* (5.8)*. If there exists a locally Lipschitz extended class $\mathcal{K}$ function γ and a measurable and locally bounded controller $\mathbf{k} : \mathbb{R}^n \to \mathbb{R}^m$ that satisfies:*

$$\min_{p \in \Lambda} \left\{ L_{\mathbf{f}} h(\mathbf{x}, \rho_p) + L_{\mathbf{g}} h(\mathbf{x}, \rho_p) \mathbf{k}(\mathbf{x}) \right\} \geq -\gamma(h_{\text{ns}}(\mathbf{x})). \tag{5.10}$$

Then h_{ns} is a valid nonsmooth CBF and the closed loop dynamics (5.2) *with controller $\mathbf{k}$ are safe with respect to C_K.*

This theorem indicates that enforcing the CBF condition only for the “least safe" pixel is sufficient to guarantee the safety of the system.

Robustness to Uncertainty

Error in the disparity propagates to the controller in the form of the measured 3D pixel position $\widehat{\rho}_p$. The measured value $\widehat{\rho}_p$ lies in a neighborhood $\mathcal{E}_p$ of the true value ρ_p, which is characterized by the error distribution $\mathcal{P}(I_i, I_j)$. We assume that the distribution $\mathcal{P}(I_i, I_j)$ is symmetric about the measured value and define the pixel-wise uncertainty set:

$$\mathcal{E}_p \triangleq \left\{ \rho \in \mathbb{R}^3 \;\middle|\; \begin{array}{c} \rho = T(\mathbf{x}, r(p, \xi)), \quad \xi \in \Gamma \\ \mathcal{P}^{\theta}(e_{1,3}(p) < |\xi - \widehat{d}(p)|; I_1, I_3) \geq \sigma \end{array} \right\} \tag{5.11}$$

where $\sigma > 0$ is a parameter defining the desired uncertainty robustness.

To achieve safety, one must determine which pixels are safety-critical given $\mathcal{E}_p$ and then enforce robust safety with respect to those pixels. The safety-critical pixels can be determined by expanding the index set Λ using the uncertainty:

$$\Lambda \subseteq \left\{ p \in K \;\middle|\; h(\mathbf{x}, \rho_p) \leq \max_{\rho_p \in \mathcal{E}_p} \min_{p \in K} h(\mathbf{x}, \rho_p) + \delta \right\}. \tag{5.12}$$

This can further be expanded to an easily calculable index set $\widehat{\Lambda} \supseteq \Lambda$ by minimizing the left-hand-side of the inequality condition and using the max-min inequality [4]:

$$\widehat{\Lambda} = \left\{ p \in K \middle| \min_{\rho_p \in \mathcal{E}_p} h(\mathbf{x}, \rho_p) \leq \min_{p \in K} \max_{\rho_p \in \mathcal{E}_p} h(\mathbf{x}, \rho_p) + \delta \right\}. \tag{5.13}$$

This expanded index set $\widehat{\Lambda}$ accounts for uncertainty and indicates which pixels are safety-critical and which constraints must be enforced to achieve safety given the pixel-wise uncertainty sets $\mathcal{E}_p$.

Measurement-Robust Control Barrier Functions (MR-CBFs) as outlined in [11] are a general method for accounting for state uncertainty in CBFs. We can use this method for each pixel $p \in \widehat{\Lambda}$ to ensure that the safety constraint is satisfied despite the uncertainty. The resulting constraint is:

$$\begin{aligned} L_{\mathbf{f}} h(\mathbf{x}, \widehat{\rho}_p) + L_{\mathbf{g}} h(\mathbf{x}, \widehat{\rho}_p)\mathbf{u} & \\ - \left(\mathfrak{L}_{L_f h} + \mathfrak{L}_{\gamma \circ h_{\text{ns}}} + \mathfrak{L}_{L_g h} \|\mathbf{u}\|_2 \right) \epsilon_p & \\ \geq -\gamma(h_{\text{ns}}(\mathbf{x})), \quad \forall p \in \widehat{\Lambda} & \end{aligned} \tag{5.14}$$

where $\mathfrak{L}$ is the Lipschitz constant of the subscript and

$$\epsilon_p \geq \max_{\rho_p \in \mathcal{E}_p} \|\rho_p - \widehat{\rho}_p\|_2 \tag{5.15}$$

is a bound on the uncertainty. Since $\Lambda \subseteq \widehat{\Lambda}$ and the MR-CBF condition implies the CBF condition (5.5), satisfying (5.14) also satisfies (5.10) providing safety of the system if $\sigma = 1$ and $\mathcal{P}^\theta = \mathcal{P}$.

5.4 Application: Obstacle Avoidance on a Quadrupedal Robot

In this section, we evaluate our approach on a quadrupedal robotic platform. With these experiments we aim to demonstrate: 1) Our method is capable of keeping the system safe in a simple do-not-collide task, and 2) Our method can adapt online to measurement uncertainty in different environments without ground-truth data.

Hardware System

For the hardware experiments we designed a custom camera array with three equally spaced inexpensive CMOS, global shutter, time-synchronized Arducam cameras. An Nvidia Jetson Nano is used to capture, downsize, and greyscale the stereo images. The images are then sent to an external computer that receives the images and outputs the filtered control input at a frequency of at least 10 Hz. The robot used

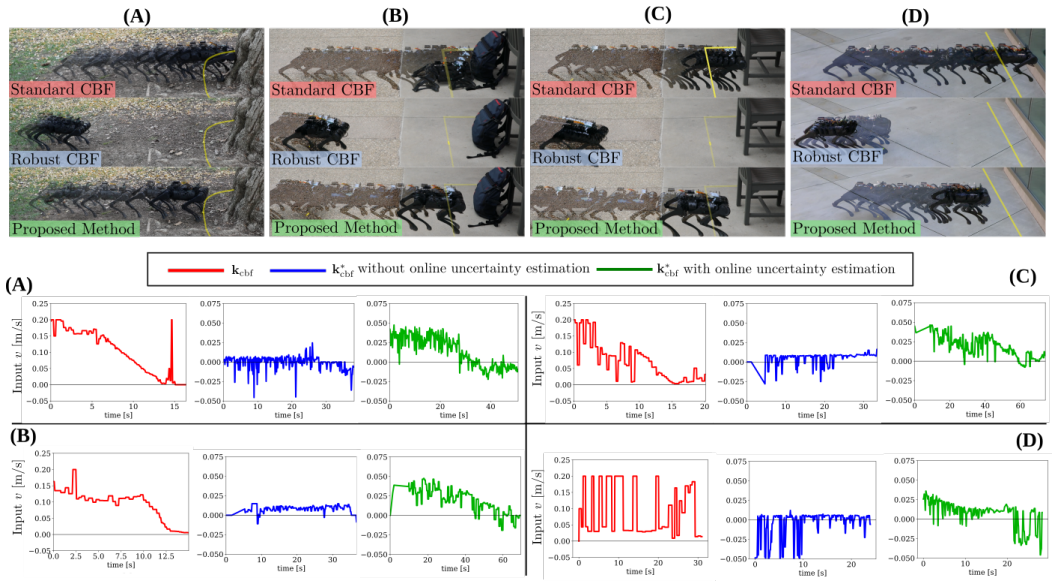


Figure 5.4: Demonstration of our method in a variety of environments. From left to right the goal is to maintain a safe distance from (**A**) a tree, (**B**) a backpack, (**C**) a chair, (**D**) and a glass window. The distance to the barrier is measured and marked on the floor with a yellow tape for visualization purposes – we emphasize this tape is not used for depth estimation. Notice that the barrier is assumed to be a sphere around an obstacle but in the case on the glass, this sphere degenerates into a plane. The quadrupedal robot is given a desired control input of 0.2 m/s. In all cases, a naive barrier implementation that simply takes the noisy measurements from a stereo vision system fails to keep the system safe. The robustified controller (5.19) with a pretrained model consistently shows overly conservative behavior. Finally, with online learning, the robot converges to the barrier without exhibiting conservative behavior, except for the glass environment where the robot is overly conservative and walks away from the barrier due to the perceived uncertainty. The (**A-D**) corresponding plots below show the control input filtered by the barrier in each of the three robustification cases.

in this experiment is a Unitree A1 quadrupedal robot that receives inputs of velocity and angle rate, $\mathbf{u} = \begin{bmatrix} v & \omega \end{bmatrix}^\intercal$. A 1 kHz Inverse Dynamics Quadratic Program (ID-QP) walking controller designed using the concepts in [5], is used to track these inputs. Stereo pixel-matching calculations were performed using Efficient LArge-scale Stereo (ELAS) [13].

Learning Method and Model

The architecture of the model used to estimate $\mathcal{P}^\theta$ is a modified version of the Hierarchical Multi-Scale Attention for Semantic Segmentation introduced in [26]; this model is relatively lightweight, consisting of only 196 thousand parameters

(e.g., network weights). The robustness threshold used was $\sigma = 0.99$ and the online learning rate was 0.001. We pretrain the model until convergence on a dataset of 6000 stereo image triples collected by manually moving the camera array through a variety of environments.

Dynamics Model and Control

In order to control the system we consider a reduced order model of the system dynamics given by the standard unicycle model. The specific form of (5.2) for this system is:

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}0\\ 0\\ 0\end{bmatrix}}_{\mathbf{f}(\mathbf{x})} + \underbrace{\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}}_{\mathbf{g}(\mathbf{x})} \underbrace{\begin{bmatrix}v\\ \omega\end{bmatrix}}_{\mathbf{k}(\mathbf{x})} \tag{5.16}$$

A formal analysis of CBFs which utilize reduced-order velocity input models is described in [20].

For this system we consider the pixel-wise CBFs,

$$h(\mathbf{x}, \rho_p) = \frac{1}{2}\left(\left\|\begin{bmatrix}x\\ y\end{bmatrix} - \begin{bmatrix}\rho_{p,x}\\ \rho_{p,y}\end{bmatrix}\right\|_2^2 - c^2\right) \tag{5.17}$$

where $\rho_{p,x}$ and $\rho_{p,y}$ indicate the global real-world x and y positions of pixel p. This function characterizes safety as remaining a planar distance $c > 0$ from ρ_p. This can be thought of as buffering surfaces in the environment by a radius c.

Robustness to Uncertainty

To illustrate the efficacy of our method we use two controllers in our experiments. A standard, unrobustified controller:

$$\begin{aligned}\mathbf{k}_{\text{cbf}} = \underset{\mathbf{u}\in\mathbb{R}^2}{\text{argmin}} \quad & \frac{1}{2}\|\mathbf{k}_{\text{des}}(\mathbf{x}) - \mathbf{u}\|_2^2 \\ \text{s.t.} \quad & \underbrace{-\begin{bmatrix}1 & 0 & 0\end{bmatrix}^\intercal r(p, \widehat{d_p})v}_{\dot{h}} \geq -\gamma(\min_{p\in K} h(\mathbf{x}, \widehat{\rho_p})), \\ & \qquad\qquad\qquad\qquad\qquad\qquad \forall p \in \Lambda\end{aligned} \tag{5.18}$$

and a robustified controller:

$$\mathbf{k}^*_{\text{cbf}} = \underset{\mathbf{u}\in\mathbb{R}^2}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{k}_{\text{des}}(\mathbf{x}) - \mathbf{u}\|_2^2 \tag{5.19}$$
$$\text{s.t.} \quad -v \geq \frac{-\gamma(\min_{p\in K} h(\mathbf{x}, \rho^*_p))}{\begin{bmatrix}1 & 0 & 0\end{bmatrix}^\intercal r(p, d^*_p)}, \quad \forall p \in \widehat{\Lambda}.$$

where $\mathbf{k}_{\text{des}} : \mathbb{R}^m \to \mathbb{R}^n$ is a desired controller, d^* is the maximum disparity for any $\rho_p \in \mathcal{E}_p$, and ρ^*_p is pixel location associated with d^*_p.

Controller (5.19) is obtained by first replacing the index set Λ with the $\widehat{\Lambda}$. Next we note that $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^\intercal r(p, \widehat{d}_p)$ is strictly positive. After dividing by this quantity, the constraint in (5.18) is robustified to account for the worst-case error as is done with MR-CBFs. Experimentally, this controller was implemented with $\delta = 0$ and a maximum of 4000 constraints.

Experimental Results

The system was run in 4 different environments (see Fig. 5.4). The CBF (5.17) was used with a safe radius of $c = 0.33$ m. The intended obstacle in the 4 different environments were (**A**) a tree, (**B**) a backpack, (**C**) a chair, (**D**) and a glass window. A desired constant forward velocity $v = 0.2$ m/s was used in each experiment and the robot was started approximately 1.3 m away from the obstacle. Since ground-truth measurements were unavailable, we use a yellow line on the ground to indicate the true location of the barrier.

For each environment three different tests were performed. First, controller (5.18) was used. Since this did not consider measurement uncertainty it failed to achieve safety in every environment; in all experiments the stereo vision overestimated the distance to objects at some point during the run and the quadruped ran directly into the obstacles. Second, the controller (5.19) was used with an error estimate computed through a pretrained function $\mathcal{P}^\theta$; this succeeded in providing safety, but was found to be overly conservative and did not allow the quadruped to approach the obstacle as desired. Third, the controller (5.19) was used with a $\mathcal{P}^\theta$ that adapted to the environment according to Algorithm 7. In this case, safety of the system was generally maintained and over time the system was able to approach the boundary of the safe set. Even when small safety violations occurred, the system eventually corrected and came to rest at a safe steady-state. These results can be seen in Figure 5.4. A video of the experiments can be found at [1].

5.5 Discussion

This chapter demonstrated that self-supervised learning can provide the perception uncertainty estimates required by robust safety controllers. The CBF framework supplies the mathematical structure for safety; learning fills the gap by estimating environment-dependent quantities that cannot be specified analytically. This division exemplifies the second strategy for structure-aware learning identified in this thesis: rather than replacing controllers with learned policies, we learn the inputs that structured controllers need to provide their guarantees. The approach parallels Chapter 4, where we learn dynamics models for MPC rather than replacing MPC with a learned policy.

The safety guarantee provided by this method rests on the assumption that the learned distribution $\mathcal{P}^{\theta}$ matches the true error distribution $\mathcal{P}$. In practice, online learning provides only an approximation, and the experiments demonstrate empirical success rather than formal certification. This gap between empirical performance and provable guarantees is characteristic of learning-based perception and motivates continued work on verifiable uncertainty quantification.

Future directions include extending the method to dynamic environments, which would require estimating obstacle motion alongside position uncertainty. The self-supervised uncertainty estimation outlined in Algorithm 7 is a general method that can be coupled with other control or state estimation frameworks beyond CBFs.

References

[1] Supplementary video, https://vimeo.com/605281037.

[2] Aaron D. Ames, Jessy W. Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". en. In: *53rd IEEE Conference on Decision and Control*. Los Angeles, CA, USA: IEEE, Dec. 2014, pp. 6271–6278. ISBN: 978-1-4673-6090-6 978-1-4799-7746-8 978-1-4799-7745-1. DOI: 10.1109/CDC.2014.7040372. URL: http://ieeexplore.ieee.org/document/7040372/ (visited on 04/07/2021).

[3] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". en. In: *IEEE Transactions on Automatic Control* 62.8 (Aug. 2017), pp. 3861–3876. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2016.2638961. URL: http://ieeexplore.ieee.org/document/7782377/ (visited on 04/07/2021).

[4] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Jonas Buchli, Mrinal Kalakrishnan, Michael Mistry, Peter Pastor, and Stefan Schaal. "Compliant quadruped locomotion over rough terrain". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 814–820. DOI: 10.1109/IROS.2009.5354681.

[6] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (Dec. 2016), pp. 1309–1332. ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2624754.

[7] Jason Choi, Fernando Castaneda, Claire J Tomlin, and Koushil Sreenath. "Reinforcement learning for safety-critical control under model uncertainty, using control Lyapunov functions and control barrier functions". In: *arXiv preprint arXiv:2004.07584* (2020).

[8] Ryan K. Cosner, Andrew W. Singletary, Andrew J. Taylor, Tamas G. Molnar, Katherine L. Bouman, and Aaron D. Ames. "Measurement-Robust Control Barrier Functions: Certainty in Safety with Uncertainty in State". en. In: *arXiv:2104.14030 [cs, eess]* (Apr. 2021). arXiv: 2104.14030. URL: http://arxiv.org/abs/2104.14030 (visited on 05/08/2021).

[9] Noel Csomay-Shanklin, Ryan K. Cosner, Min Dai, Andrew J. Taylor, and Aaron D. Ames. "Episodic Learning for Safe Bipedal Locomotion with Control Barrier Functions and Projection-to-State Safety". en. In: *arXiv:2105.01697 [cs]* (May 2021). arXiv: 2105.01697. URL: http://arxiv.org/abs/2105.01697 (visited on 05/08/2021).

[10] Sarah Dean, Nikolai Matni, Benjamin Recht, and Vickie Ye. "Robust Guarantees for Perception-Based Control". en. In: *arXiv:1907.03680 [cs, math, stat]* (Dec. 2019). arXiv: 1907.03680. URL: http://arxiv.org/abs/1907.03680 (visited on 04/24/2021).

[11] Sarah Dean, Andrew J. Taylor, Ryan K. Cosner, Benjamin Recht, and Aaron D. Ames. "Guaranteeing Safety of Learned Perception Modules via Measurement-Robust Control Barrier Functions". en. In: *arXiv:2010.16001 [cs, eess, math, stat]* (Oct. 2020). arXiv: 2010.16001. URL: http://arxiv.org/abs/2010.16001 (visited on 04/23/2021).

[12] Yousef Emam, Paul Glotfelter, and Magnus Egerstedt. "Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 3984–3990.

[13] Andreas Geiger, Martin Roser, and Raquel Urtasun. "Efficient large-scale stereo matching". In: *Asian conference on computer vision*. Springer. 2010, pp. 25–38.

[14] Jeremy H. Gillula and Claire J. Tomlin. "Guaranteed Safe Online Learning via Reachability: tracking a ground target using a quadrotor". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2723–2730. DOI: 10.1109/ICRA.2012.6225136.

[15] Paul Glotfelter, Jorge Cortes, and Magnus Egerstedt. "A Nonsmooth Approach to Controller Synthesis for Boolean Specifications". In: *IEEE Transactions on Automatic Control* (2020), pp. 1–1. ISSN: 1558-2523. DOI: 10.1109/TAC.2020.3035467.

[16] Paul Glotfelter, Jorge Cortes, and Magnus Egerstedt. "Nonsmooth Barrier Functions With Applications to Multi-Robot Systems". en. In: *IEEE Control Systems Letters* 1.2 (Oct. 2017), pp. 310–315. ISSN: 2475-1456. DOI: 10.1109/LCSYS.2017.2710943. URL: http://ieeexplore.ieee.org/document/7937882/ (visited on 06/07/2021).

[17] Elad Hazan. *Introduction to Online Convex Optimization*. 2019. arXiv: 1909.05207 [cs.LG].

[18] Alec Koppel, Jonathan Fink, Garrett Warnell, Ethan Stump, and Alejandro Ribeiro. "Online learning for characterizing unknown environments in ground robotic vehicle models". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 626–633. DOI: 10.1109/IROS.2016.7759118.

[19] Katherine Liu, Kyel Ok, William Vega-Brown, and Nicholas Roy. "Deep Inference for Covariance Estimation: Learning Gaussian Noise Models for State Estimation". en. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 1436–1443. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8461047. URL: https://ieeexplore.ieee.org/document/8461047/ (visited on 04/09/2021).

[20] Tamas G. Molnar, Ryan K. Cosner, Andrew W. Singletary, Wyatt Ubellacker, and Aaron D. Ames. "Model-Free Safety-Critical Control for Robotic Systems". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 944–951. DOI: 10.1109/LRA.2021.3135569.

[21] Motoya Ohnishi, Li Wang, Gennaro Notomista, and Magnus Egerstedt. "Barrier-Certified Adaptive Reinforcement Learning With Applications to Brushbot Navigation". In: *IEEE Transactions on Robotics* 35.5 (2019), pp. 1186–1205. DOI: 10.1109/TRO.2019.2920206.

[22] Mathias Perrollaz, Anne Spalanzani, and Didier Aubert. "Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection". In: *2010 IEEE Intelligent Vehicles Symposium*. IEEE. 2010, pp. 313–318.

[23] Andrew Singletary, Karl Klingebiel, Joseph Bourne, Andrew Browning, Phil Tokumaru, and Aaron Ames. "Comparative Analysis of Control Barrier Functions and Artificial Potential Fields for Obstacle Avoidance". en. In:

arXiv:2010.09819 [cs, eess] (Oct. 2020). arXiv: 2010.09819. URL: http://arxiv.org/abs/2010.09819 (visited on 04/23/2021).

[24] Boris Sofman, Ellie Lin, J Andrew Bagnell, John Cole, Nicolas Vandapel, and Anthony Stentz. "Improving robot navigation through self-supervised online learning". In: *Journal of Field Robotics* 23.11-12 (2006), pp. 1059–1075.

[25] Richard Szeliski and Daniel Scharstein. "Symmetric sub-pixel stereo matching". In: *European Conference on Computer Vision*. Springer. 2002, pp. 525–540.

[26] Andrew Tao, Karan Sapra, and Bryan Catanzaro. *Hierarchical Multi-Scale Attention for Semantic Segmentation*. 2020. arXiv: 2005.10821 [cs.CV].

[27] Andrew J Taylor, Andrew Singletary, Yisong Yue, and Aaron D Ames. "Learning for Safety-Critical Control with Control Barrier Functions". en. In: (), p. 10.

[28] Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. "Robustness of control barrier functions for safety critical control". In: *IFAC-PapersOnLine* (2015).

Part IV

The Zero Dynamics Policy (ZDP) Trilogy

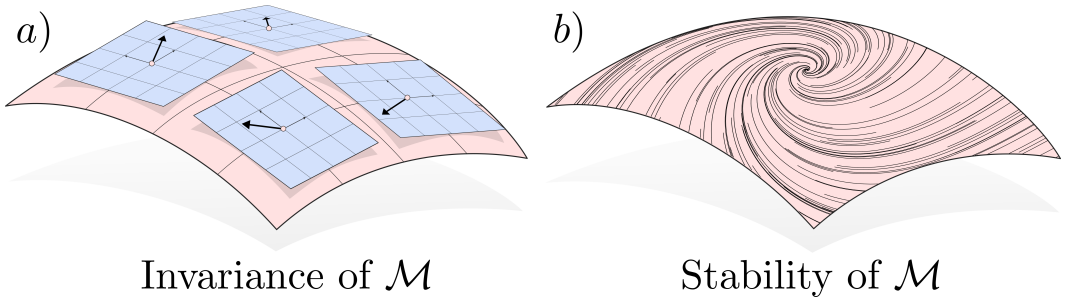


Figure 6.1: The two conditions required of the zeroing manifold: a) controlled invariance, and b) stable zero dynamics.

Chapter 6

CONSTRUCTIVE NONLINEAR CONTROL OF UNDERACTUATED SYSTEMS VIA ZERO DYNAMICS POLICIES

6.1 Introduction

This chapter establishes the theoretical foundations for controlling underactuated systems through learned manifolds with guaranteed stability. *Underactuated* systems pose fundamental challenges to control theory because one cannot directly command all degrees of freedom. Legged robots, dexterous manipulators, and systems with strict actuator limits all exhibit passive dynamics that cannot be arbitrarily shaped through feedback. When these passive dynamics happen to be stable, constructive feedback controllers can be synthesized using input-output linearization [16] or control Lyapunov functions [19]. Stabilizing underactuated systems without stability assumptions on the passive dynamics remains a challenging open problem.

The framework developed here builds on a key observation: the existence of passive dynamics does not necessarily imply loss of stabilizability. Achieving stable behaviors requires careful coordination of the actuated degrees of freedom with the unactuated ones. The central tool for this coordination is the *zero dynamics*: the residual dynamics that remain when all outputs have been driven to zero [10]. Traditional approaches to zero dynamics require the designer to craft output coordinates and then algebraically verify that the resulting zero dynamics are stable. This guess-and-check procedure arises from the difficulty of finding feedback linearizable output coordinates that span the complete state space [3].

This chapter takes a different approach: we treat the outputs as learnable design variables, enabling stable zero dynamics by construction. We prove that stabilizing to the zero dynamics surface defined by these learned outputs results in stability of the overall system, as illustrated in Figure 6.1. The resulting *Zero Dynamics Policies (ZDPs)* take the form of a mapping from underactuated states to actuated states, defining a controlled invariant and stable manifold. ZDPs exemplify how control-theoretic structure can serve as an inductive bias for learning: rather than learning a controller directly, we learn the manifold on which a structured controller achieves guaranteed stability. Pointwise Lyapunov and barrier conditions certify stability and safety by requiring inequalities to hold at each state; ZDPs provide a complementary approach where the learned object is a coordinate transformation that renders stability tractable.

The perspective developed here has origins in the stabilization of non-minimum phase systems [6, 21, 9] and extends work on Hybrid Zero Dynamics (HZD) for bipedal walking [22]. These prior methods are often domain-specific and challenging to synthesize, leading practitioners to turn to optimal control instead. Both approximate value function feedback [13, 17] and receding horizon Model Predictive Control [5, 8] are common in practice. These methods are useful but provide limited insight into why stabilizing underactuated systems is fundamentally difficult. We take inspiration from legged locomotion [11, 7, 14], where mappings between underactuated coordinates (center of mass position) and actuated coordinates (foot placement) are central to controller synthesis. The authors exploited this connection in [15], where planar biped walking was generated by enforcing barrier function certificates on the zero dynamics manifold through learned output parameters. The goal of this chapter is to formalize and unify these approaches into a general framework.

Contributions. We make three contributions. First, we prove that for locally controllable nonlinear systems, a stabilizing ZDP exists and can be constructed analytically in a neighborhood of the origin. Second, we show that optimal control provides a constructive method for extending the region of validity of ZDPs beyond this local neighborhood. Third, we demonstrate on the cartpole system that ZDPs yield a larger region of attraction than linear methods.

Broader Context. These results establish the theoretical foundations for the hardware experiments that follow. Chapter 7 extends ZDPs to hybrid dynamics with impacts and demonstrates bipedal walking on the AMBER-3M platform. Chapter 8 develops discrete-time ZDPs for hopping robots and validates the approach with

over three thousand hops on the ARCHER platform across stairs, ramps, and narrow bridges.

6.2 Preliminaries

Consider a control-affine nonlinear system:

$$\dot{\mathbf{x}} = \mathbf{f}_{\mathbf{x}}(\mathbf{x}) + \mathbf{g}_{\mathbf{x}}(\mathbf{x})v \tag{6.1}$$

with state $\mathbf{x} \in \mathbb{R}^n$, input $v \in \mathbb{R}$, and functions $\mathbf{f}_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{g}_{\mathbf{x}} : \mathbb{R}^n \to \mathbb{R}^n$ assumed to be continuously differentiable on $\mathbb{R}^n$. When analyzing underactuated systems, it will be useful to examine how actuation enters the system dynamics; to this end, consider an output $y : \mathbb{R}^n \to \mathbb{R}$. In order for the evolution of this output to be impacted by a controller, the input v must appear in a derivative of y in a meaningful way. Now, compute a time derivative of the output:

$$\dot{y}(\mathbf{x}) = \underbrace{\frac{\partial y}{\partial \mathbf{x}}\mathbf{f}_{\mathbf{x}}(\mathbf{x})}_{L_{\mathbf{f}_{\mathbf{x}}}y(\mathbf{x})} + \underbrace{\frac{\partial y}{\partial \mathbf{x}}\mathbf{g}_{\mathbf{x}}(\mathbf{x})}_{L_{\mathbf{g}_{\mathbf{x}}}y(\mathbf{x})} v$$

where $L_{\mathbf{f}_{\mathbf{x}}}y : \mathbb{R}^n \to \mathbb{R}$ and $L_{\mathbf{g}_{\mathbf{x}}}y : \mathbb{R}^n \to \mathbb{R}$ are the *Lie derivatives* of the output y with respect to the vector fields $\mathbf{f}_{\mathbf{x}}$ and $\mathbf{g}_{\mathbf{x}}$, respectively. If $L_{\mathbf{g}_{\mathbf{x}}}y(\mathbf{x}) \equiv 0$, we can attempt to continue differentiating until a higher derivative is nonzero:

$$y^{(\gamma)}(\mathbf{x}) = L^{\gamma}_{\mathbf{f}_{\mathbf{x}}}y(\mathbf{x}) + L_{\mathbf{g}_{\mathbf{x}}}L^{\gamma-1}_{\mathbf{f}_{\mathbf{x}}}(\mathbf{x})v.$$

Differentiating the output until the input appears is captured in the following notion of strict relative degree:

Definition 13. *[16] An output $y : \mathbb{R}^n \to \mathbb{R}$ for the system (6.1) is said to have relative degree $\gamma \in \mathbb{N}$ at $\mathbf{x}_0$ if:*

$$L_{\mathbf{g}_{\mathbf{x}}}L^{k}_{\mathbf{f}_{\mathbf{x}}}(\mathbf{x}) \equiv 0, \quad 0 \leq k \leq \gamma - 2$$

and $L_{\mathbf{g}_{\mathbf{x}}}L^{\gamma-1}_{\mathbf{f}_{\mathbf{x}}}(\mathbf{x}) \neq 0$.

Given an output of relative degree $\gamma \in \mathbb{N}$, consider the mapping $\boldsymbol{\Phi}_{\boldsymbol{\eta}} : \mathbb{R}^n \to \mathcal{N} \triangleq \mathbb{R}^{\gamma}$, defined as:

$$\boldsymbol{\Phi}_{\boldsymbol{\eta}}(\mathbf{x}) \triangleq \begin{bmatrix} y(\mathbf{x}) & \dot{y}(\mathbf{x}) & \cdots & y^{(\gamma-1)}(\mathbf{x}) \end{bmatrix}^{\top}. \tag{6.2}$$

We will subsequently take $\boldsymbol{\eta} = \boldsymbol{\Phi}_{\boldsymbol{\eta}}(\mathbf{x}) \in \mathcal{N}$ to represent coordinates of the output space. Valid relative degree allows the constructive synthesis of controllers which exponentially stabilizes the outputs [16], defined as:

Definition 14. *The signal $\boldsymbol{\eta}(t)$ is exponentially stable on domain $D \subset \mathcal{N}$ if there exists $M, \lambda > 0$ such that:*

$$\boldsymbol{\eta}_0 \in D \quad \Longrightarrow \quad \|\boldsymbol{\eta}(t)\| \leq M e^{-\lambda t} \|\boldsymbol{\eta}_0\|.$$

Lyapunov theory [18] states that exponential stability is one-to-one with the existence of a control Lyapunov function (CLF) $V : \mathcal{N} \to \mathbb{R}$ satisfying:

$$\begin{aligned} k_1 \|\boldsymbol{\eta}\|^2 \leq V(\boldsymbol{\eta}) \leq k_2 \|\boldsymbol{\eta}\|^2 \\ \inf_v \dot{V}(\mathbf{x}, v) \leq -k_3 V(\boldsymbol{\eta}). \end{aligned} \tag{6.3}$$

for $k_i > 0$. We define $\mathcal{K} = \{k(\mathbf{x}) \mid \dot{V}(\mathbf{x}, k(\mathbf{x})) \leq -k_3 V(\boldsymbol{\eta})\}$ to be the set of all output exponentially stabilizing feedback controllers, which is nonempty under valid relative degree [1]. A common technique to stabilize outputs with valid relative degree is via feedback linearization:

$$k_{\text{fbl}}(\mathbf{x}, u) = \left(L_{\mathbf{g_x}} L_{\mathbf{f_x}}^{\gamma-1}(\mathbf{x})\right)^{-1} \left(-L_{\mathbf{f_x}}^{\gamma}(\mathbf{x}) + u\right)$$

for $k_{\text{fbl}} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, with $u \in \mathbb{R}$ the auxiliary input. Under this controller, the $\boldsymbol{\eta}$ dynamics become:

$$\dot{\boldsymbol{\eta}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ 0 & \mathbf{0} \end{bmatrix}}_{\triangleq \mathbf{F}} \boldsymbol{\eta} + \underbrace{\begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}}_{\triangleq \mathbf{G}} u,$$

for $\mathbf{F} \in \mathbb{R}^{\gamma \times \gamma}, \mathbf{G} \in \mathbb{R}^{\gamma}$. Once a system's available outputs are zeroed, the remaining states evolve on a manifold [10].

In this context, consider a differentiable function $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^p$ with $\mathbf{0}$ a regular value, i.e. $h(\mathbf{x}) = \mathbf{0}$ implies $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ is full rank. Then, we have that $\mathcal{M} \triangleq \{\mathbf{x} \mid \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$ defines a $n - p$-dimensional embedded submanifold of $\mathbb{R}^n$ [20]. Associated with such a manifold is the notion of a tangent space. A vector $\mathbf{v} \in \mathbb{R}^n$ is a *tangent vector* to a manifold $\mathcal{M}$ at the point $\mathbf{x} \in \mathcal{M}$, denoted as $\mathbf{v} \in \mathsf{T}_{\mathbf{x}}\mathcal{M}$, if:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top} \mathbf{v} = \mathbf{0}.$$

This aligns with the classical notions of tangent vectors, as the gradient field of the function $\mathbf{h}$ forms a basis for the annihilator of the *tangent space* at a point, $\mathsf{T}_{\mathbf{x}}\mathcal{M}$.

A key property will be the notion of controlled invariance for such a manifold:

Definition 15. *A manifold $\mathcal{M}$ is controlled invariant under the dynamics* (6.1) *if for all* $\mathbf{x} \in \mathbb{R}^n$ *there exists an input* $v \in \mathbb{R}$ *such that:*

$$\mathbf{f}_{\mathbf{x}}(\mathbf{x}) + \mathbf{g}_{\mathbf{x}}(\mathbf{x})v \in \mathsf{T}_{\mathbf{x}}\mathcal{M}.$$

That is, there must exist an input such that the vector field associated with the dynamics lies in the tangent space of the manifold. Our proposed method aims to find controlled invariant manifolds with exponentially stable dynamics.

Finally, we introduce the *actuation decomposition*, which highlights the structure of actuated and unactuated states. When y is valid relative degree, each $y^{(i)}$, for $i = 0, \ldots, \gamma - 1$ are linearly independent, and $\boldsymbol{\eta}$ forms a basis for γ dimensions of $\mathbb{R}^n$ [10]. We can construct a set of normal coordinates $\mathbf{z} \in \mathcal{Z} \subset \mathbb{R}^{n_z}$, where $n_z = n - \gamma$, via $\boldsymbol{\Phi}_{\mathbf{z}} : \mathbb{R}^n \to \mathcal{Z}$. This transform is defined such that $\boldsymbol{\Phi} : \mathbb{R}^n \to \mathcal{N} \times \mathcal{Z}$ given by:

$$\begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{\eta}}(\mathbf{x}) \\ \boldsymbol{\Phi}_{\mathbf{z}}(\mathbf{x}) \end{bmatrix} \triangleq \boldsymbol{\Phi}(\mathbf{x})$$

is a diffeomorphism, and $\frac{\partial \boldsymbol{\Phi}_{\mathbf{z}}}{\partial \mathbf{x}} \mathbf{g}_{\mathbf{x}} \equiv 0$. For the system (6.1), such coordinates are guaranteed to exist by Frobenius Theorem [10]. This implies that the $\mathbf{z}$ dynamics are independent of the control input:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\mathbf{z}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{F}\boldsymbol{\eta} \\ \boldsymbol{\omega}(\boldsymbol{\eta}, \mathbf{z}) \end{bmatrix}}_{\mathbf{f}(\boldsymbol{\zeta})} + \underbrace{\begin{bmatrix} \mathbf{G} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{g}(\boldsymbol{\zeta})} u \tag{6.4}$$

where $\boldsymbol{\zeta} \triangleq (\boldsymbol{\eta}, \mathbf{z}) \in \mathcal{X} \triangleq \mathcal{N} \times \mathcal{Z}$. We deem this transformation an *actuation decomposition*; it separates the system states into a set of directly actuated coordinates ($\boldsymbol{\eta}$), and a set of unactuated coordinates ($\mathbf{z}$). As this decomposition takes a central role in our approach to underactuated control, (6.4) will be the starting point for dynamics in this paper.

6.3 Zero Dynamics Policies

We propose a differentiable mapping $\boldsymbol{\psi} : \mathcal{Z} \to \mathcal{N}$, which maps from the underactuated states $\mathbf{z}$ to desired locations for the actuated states $\boldsymbol{\eta}$. This is motivated by several results in robotics, such as the Raibert Heuristic, which maps a walking robot's center of mass (unactuated) to desired foot positions (actuated). The mapping $\boldsymbol{\psi}$ induces an n_z dimensional submanifold of $\mathcal{X}$ via the zero level set of the function $\mathbf{h}(\boldsymbol{\eta}, \mathbf{z}) \triangleq \boldsymbol{\eta} - \boldsymbol{\psi}(\mathbf{z})$:

$$\mathcal{M}_{\boldsymbol{\psi}} \triangleq \{(\boldsymbol{\eta}, \mathbf{z}) \in \mathcal{N} \times \mathcal{Z} \mid \mathbf{h}(\boldsymbol{\eta}, \mathbf{z}) = \mathbf{0}\}, \tag{6.5}$$

as $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\zeta}} = [\mathbf{I}\ \frac{\partial \boldsymbol{\psi}}{\partial \mathbf{z}}]$ is full row rank. With this, we can now introduce the notion of *zero dynamics*:

Definition 16. *The zero dynamics associated with a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$ are given by:*

$$\dot{\mathbf{z}} = \boldsymbol{\omega}(\boldsymbol{\psi}(\mathbf{z}), \mathbf{z}).$$

Required Properties of Zero Dynamics Policies

If the zero dynamics are exponentially stable, we will show that stabilizing to $\mathcal{M}_{\boldsymbol{\psi}}$ stabilizes the whole system. To this end, we propose the following output:

$$y = \eta_1 - \psi_1(\mathbf{z}), \tag{6.6}$$

where $(\cdot)_i$ denotes the i^{th} index. The following assumption is required to ensure y can maintain relative degree γ:

Assumption 3. We have that $\frac{\partial \boldsymbol{\omega}}{\partial \eta_i} = \mathbf{0}$ for all $i = 3, \dots, \gamma$.

This assumption is trivially satisfied for $\gamma \leq 2$, a case typical for robotic systems. We now give a condition for the relative degree of (6.6):

Lemma 3. *The output $y(\boldsymbol{\zeta}) = \eta_1 - \psi_1(\mathbf{z})$ has valid relative degree γ if and only if $\frac{\partial \psi_1}{\partial z}\frac{\partial \boldsymbol{\omega}}{\partial \eta_2} \neq 1$.*

Proof. Taking derivatives until the input appears yields:

$$\begin{aligned}
\dot{y} &= \eta_2 - \frac{\partial \psi_1}{\partial \mathbf{z}}\boldsymbol{\omega}(\boldsymbol{\eta}, \mathbf{z}) \\
y^{(i)} &= \left(1 - \frac{\partial \psi_1}{\partial \mathbf{z}}\frac{\partial \boldsymbol{\omega}}{\partial \eta_2}\right)\eta_{i+1} + W_i(\boldsymbol{\eta}_{1:i}, \mathbf{z}), \\
y^{(\gamma)} &= \left(1 - \frac{\partial \psi_1}{\partial \mathbf{z}}\frac{\partial \boldsymbol{\omega}}{\partial \eta_2}\right)u + W_\gamma(\boldsymbol{\eta}, \mathbf{z})
\end{aligned} \tag{6.7}$$

for $i = 2, \dots, \gamma - 1$ and where $W_i : \mathcal{N} \times \mathcal{Z} \to \mathbb{R}$ is introduced to hold additional terms for each derivative , and $\boldsymbol{\eta}_{i:j} = [\eta_i\ \dots\ \eta_j]^\top$. As u does not appear until $y^{(\gamma)}$, we have that $L_{\mathbf{g}}L_{\mathbf{f}}^i y \equiv 0$ for $i = 0, \dots, \gamma - 2$, and $L_{\mathbf{g}}L_{\mathbf{f}}^i y = 1 - \frac{\partial \psi_1}{\partial \mathbf{z}}\frac{\partial \boldsymbol{\omega}}{\partial \eta_2}$. Therefore, the output is relative degree γ if and only if this term is nonzero. □

In the case $\gamma = 1$, the output (6.6) has valid relative degree one. Importantly, if y is valid relative degree, each $y^{(i)}$, for $i = 0, \dots, \gamma - 1$ are linearly independent, and

form a basis for γ dimensions of $\mathcal{X}$ [10]. Defining the error coordinates:

$$\mathbf{e} = \begin{bmatrix} y & \dot{y} & \cdots & y^{(\gamma-1)} \end{bmatrix}^\top \in \mathcal{E} \tag{6.8}$$

as y and its first $\gamma - 1$ derivatives, we can construct the associated zeroing manifold of the output $y = \eta_1 - \psi_1(\mathbf{z})$.

Lemma 4. *Consider a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$ and its associated output $y = \eta_1 - \psi_1(\mathbf{z})$. If y has relative degree, then $\mathcal{M}_{\boldsymbol{\psi}}$ is the zeroing manifold associated with* (6.6)*.*

Proof. Valid relative degree implies the existence of a unique zeroing manifold $\mathcal{M}$, an n_z dimensional surface on which $\mathbf{e} \equiv 0$ [10]. Because $\mathcal{M}_{\boldsymbol{\psi}}$ is controlled invariant with $y \equiv 0$ (implying derivatives of y are zero), it is also an n_z dimensional zeroing surface; by uniqueness, $\mathcal{M} = \mathcal{M}_{\boldsymbol{\psi}}$. □

We have shown that a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$ is the zeroing manifold associated with (6.6) when this output is valid relative degree. If a function $\boldsymbol{\psi}$ can be found such that the zero dynamics on $\mathcal{M}_{\boldsymbol{\psi}}$ are also stable, then the system can be constructively stabilized as follows:

Theorem 10. *Consider a relative degree γ output $y = \eta_1 - \psi_1(\mathbf{z})$ with zeroing manifold $\mathcal{M}_{\boldsymbol{\psi}}$. If the zero dynamics of $\mathcal{M}_{\boldsymbol{\psi}}$ are exponentially stable, then any output stabilizing controller $\mathbf{k} \in \mathcal{K}$ renders the full state $\boldsymbol{\zeta}$ exponentially stable.*

Proof. By Lemma 4, the relative degree of y implies that holding $\mathbf{e} = 0$ renders $\mathcal{M}_{\boldsymbol{\psi}}$ invariant. Since $\mathcal{M}_{\boldsymbol{\psi}}$ has exponentially stable dynamics when rendered invariant, converse Lyapunov guarantees the existence of $V_{\mathbf{z}}(\mathbf{z})$ satisfying:

$$\begin{aligned} k_{1,z}\|\mathbf{z}\|^2 \leq V_{\mathbf{z}}(\mathbf{z}) &\leq k_{2,z}\|\mathbf{z}\|^2 \\ \dot{V}_{\mathbf{z}}(\mathbf{z}) = \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}}\boldsymbol{\omega}(\boldsymbol{\psi}(\mathbf{z}), \mathbf{z}) &\leq -k_{3,z}\|\mathbf{z}\|^2 \\ \left\| \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \right\| &\leq k_{4,z}\|\mathbf{z}\| \end{aligned}$$

for $k_{i,z} > 0$. Applying any controller $k(\boldsymbol{\zeta}) \in \mathcal{K}$ (nonempty by virtue of valid relative degree) implies by converse Lyapunov the existence of a Lyapunov function on the error. Define $V_{\mathbf{e}}(\mathbf{e})$ satisfying:

$$\begin{aligned} k_{1,e}\|\mathbf{e}\|^2 \leq V_{\mathbf{e}}(\mathbf{e}) &\leq k_{2,e}\|\mathbf{e}\|^2 \\ \dot{V}_{\mathbf{e}}(\mathbf{e}) &\leq -k_{3,e}\|\mathbf{e}\|^2 \end{aligned}$$

for $k_{i,e} > 0$. We then use the implicit function theorem to establish that $\boldsymbol{\eta}$ can be written as a function of $\mathbf{e}, \mathbf{z}$:

$$\boldsymbol{\xi}(\mathbf{e}, \boldsymbol{\eta}, \mathbf{z}) \triangleq \mathbf{e} - \begin{bmatrix} y & \dot{y} & \dots & y^{(\gamma-1)} \end{bmatrix}^\top = \mathbf{0}$$

Observe from (6.7) that $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}}$ is lower triangular. Furthermore, $(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}})_{1,1} = 1$ and $(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}})_{i,i} = (1 - \frac{\partial \psi_1}{\partial \mathbf{z}} \frac{\partial \boldsymbol{\omega}}{\partial \eta_2})$ for $i = 2, \dots, \gamma$. By assumption of relative degree, the diagonal elements are nonzero and therefore $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}}$ is invertible. Therefore, there exists a function $\boldsymbol{\Gamma} : \mathcal{E} \times \mathcal{Z} \to \mathcal{N}$ such that $\boldsymbol{\xi}(\mathbf{e}, \boldsymbol{\Gamma}(\mathbf{e}, \mathbf{z}), \mathbf{z}) = 0$. We can then redefine the $\mathbf{z}$ dynamics in terms of $\mathbf{e}$ via:

$$\tilde{\boldsymbol{\omega}}(\mathbf{e}, \mathbf{z}) \triangleq \boldsymbol{\omega}\left(\boldsymbol{\Gamma}(\mathbf{e}, \mathbf{z}), \mathbf{z}\right) = \boldsymbol{\omega}(\boldsymbol{\eta}, \mathbf{z}).$$

Finally, consider the positive definite function $V(\mathbf{e}, \mathbf{z}) = \sigma V_{\mathbf{e}}(\mathbf{e}) + V_{\mathbf{z}}(\mathbf{z})$, for $\sigma > 0$, whose time derivative is:

$$\begin{aligned}
\dot{V}(\mathbf{e}, \mathbf{z}) &= \sigma \dot{V}_{\mathbf{e}}(\mathbf{e}) + \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \tilde{\boldsymbol{\omega}}(\mathbf{e}, \mathbf{z}) \\
&= \dot{V}_{\mathbf{e}}(\mathbf{e}) + \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \tilde{\boldsymbol{\omega}}(\mathbf{0}, \mathbf{z}) + \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \left(\tilde{\boldsymbol{\omega}}(\mathbf{e}, \mathbf{z}) - \tilde{\boldsymbol{\omega}}(\mathbf{0}, \mathbf{z})\right) \\
&\leq -\sigma k_{3,e} \|\mathbf{e}\|^2 - k_{3,z} \|\mathbf{z}\|^2 + k_{4,z} L_\omega \|\mathbf{e}\| \|\mathbf{z}\| \\
&= -\begin{bmatrix} \|\mathbf{e}\| \\ \|\mathbf{z}\| \end{bmatrix}^\top \begin{bmatrix} \sigma k_{3,e} & -\frac{k_{4,z} L_\omega}{2} \\ -\frac{k_{4,z} L_\omega}{2} & k_{3,z} \end{bmatrix} \begin{bmatrix} \|\mathbf{e}\| \\ \|\mathbf{z}\| \end{bmatrix}
\end{aligned} \tag{6.9}$$

where L_ω is a Lipshitz constant of $\tilde{\boldsymbol{\omega}}$. Choosing $\sigma > \frac{k_{4,z}^2 L_\omega^2}{4 k_{3,e} k_{3,z}}$ renders the matrix positive definite, and the quadratic form (6.9) can be bounded, for $\lambda > 0$:

$$\dot{V}(\mathbf{e}, \mathbf{z}) \leq -\lambda V(\mathbf{e}, \mathbf{z})$$

since V can be bounded by quadratic functions, certifying that the V is a Lyapunov function by (6.3). Thus, the composite system is exponentially stable under any controller which exponentially stabilizes the outputs. □

Local Existence of Stabilizing Zero Dynamics Policies

Around the origin, we demonstrate via construction that $\boldsymbol{\psi}$ exists for any locally controllable nonlinear system. To achieve this, we identify a stable zeroing manifold for the linear system and leverage the relationship between a nonlinear system and its linearization to generate an output with stable zero dynamics. To this end, consider

the linearization of (6.4) about the origin:

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\boldsymbol{\eta}}_{2:\gamma-1} \\ \dot{\eta}_\gamma \\ \dot{\mathbf{z}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ 0 & 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{a}_{\eta_1} & \mathbf{a}_{\eta_2} & \mathbf{0} & \mathbf{A}_\mathbf{z} \end{bmatrix}}_{\triangleq \mathbf{A}} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \boldsymbol{\eta}_{3:\gamma} \\ \mathbf{z} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \mathbf{0} \\ 1 \\ \mathbf{0} \end{bmatrix}}_{\triangleq \mathbf{B}} u \tag{6.10}$$

with $\mathbf{a}_{\eta_i} = \frac{\partial \omega}{\partial \eta_i} \in \mathbb{R}^{n_z}$ and $\mathbf{A}_\mathbf{z} = \frac{\partial \omega}{\partial \mathbf{z}} \in \mathbb{R}^{n_z \times n_z}$. First, we identify a manifold invariant under a stabilizing controller:

Lemma 5. *There exists a nonempty set of controllers which stabilize* (6.10) *and induce an n_z dimensional invariant subspace $\mathcal{S}$ such that for each $\mathbf{z}$ there exists a unique $\boldsymbol{\eta}$ such that $(\boldsymbol{\eta}, \mathbf{z}) \in \mathcal{S}$.*

Proof. By assumption the nonlinear system is locally controllable; therefore (6.10) is controllable [10]. As such, the system can be stabilized by pole placement. Let $u = -\mathbf{K}\boldsymbol{\zeta}$ be any controller which places the poles at unique locations on the negative real axis. Then the closed loop system, $\mathbf{A} - \mathbf{B}\mathbf{K}$, will have n unique, linearly independent eigenspaces [2]. It is therefore possible to pick n_z distinct eigenvectors such that the projection onto $\mathcal{Z}$ spans the $\mathcal{Z}$ subspace. Let $\mathbf{v}_1, \ldots, \mathbf{v}_{n_z} \in \mathbb{R}^n$ be these eigenvectors. Define

$$\mathbf{S} = \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_{n_z} \end{bmatrix}$$

Then $\mathcal{S} = \text{span}(\mathbf{S})$ is an invariant subspace of the closed loop dynamics, since it is the span of eigenspaces. Given a point $\mathbf{z}$, the corresponding point on $\mathcal{S}$ is given by

$$\begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{z} \end{bmatrix} = \mathbf{S} \left(\begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \mathbf{S} \right)^{-1} \mathbf{z} = \begin{bmatrix} \mathbf{S}_{\boldsymbol{\eta}}^\top \\ \mathbf{I} \end{bmatrix} \mathbf{z}$$

where $\mathbf{S}_{\boldsymbol{\eta}} = \begin{bmatrix} \mathbf{s}_{\eta_1} & \ldots & \mathbf{s}_{\eta_\gamma} \end{bmatrix}$, with $\mathbf{s}_{\eta_i} \in \mathbb{R}^{n_z}$. The matrix inverse is well defined since $\mathbf{S}$ is selected such that its projection onto $\mathbf{z}$ coordinates spans $\mathcal{Z}$. □

Remark 4. While Lemma 5 is proven for a specific form of controller (pole placement), nearly all stabilizing controllers will have n_z dimensional invariant subspaces which are parameterizable by $\mathbf{z}$. Any such subspace can be chosen, and the resulting $\mathbf{S}$ can be used in the following analysis.

Lemma 5 defines an invariant manifold for the controlled linear system. In order to appeal to composite stability, there must exist a controller rendering $\mathcal{S}$ attractive.

This can be achieved by constructing an associated output with valid relative degree. However, from [10], we know that there may exist zeroing manifolds that do not have valid relative degree. We now show that under the assumption of controllability, $\mathcal{S}$ is the zeroing manifold of a suitable output:

Lemma 6. *Consider a n_z dimensional subspace $\mathcal{S}$ satisfying Lemma 5. The output $y = \eta_1 - \mathbf{s}_{\eta_1}^\top \mathbf{z}$ has valid relative degree γ and $\mathcal{S}$ is the zeroing manifold for this output.*

Proof. Take $\mathbf{C} \triangleq \begin{bmatrix} 1 & \mathbf{0} & -\mathbf{s}_{\eta_1}^\top \end{bmatrix}$ such that $y = \mathbf{C}\zeta$. Consider the closed loop matrix $\mathbf{A}_{cl} = \mathbf{A} - \mathbf{B}\mathbf{K}$. For compactness, define:

$$\begin{aligned} m_k &\triangleq -\mathbf{s}_{\eta_1}^\top \mathbf{A}_{\mathbf{z}}^k \mathbf{a}_{\eta_1} \\ \mathbf{O}_k &\triangleq -\mathbf{s}_{\eta_1}^\top \mathbf{A}_{\mathbf{z}}^k \\ q_k &\triangleq \mathbf{O}_k \left(\mathbf{a}_{\eta_1} + \mathbf{A}_{\mathbf{z}} \mathbf{a}_{\eta_2} \right) \\ p &\triangleq 1 - \mathbf{s}_{\eta_1}^\top \mathbf{a}_{\eta_2} \end{aligned}$$

Using these we construct the matrix $\mathbf{E} \in \mathbb{R}^{\gamma \times n}$:

$$\mathbf{E} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A}_{cl} \\ \mathbf{C}\mathbf{A}_{cl}^2 \\ \vdots \\ \mathbf{C}\mathbf{A}_{cl}^{\gamma-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & \mathbf{O}_0 \\ m_0 & p & 0 & \cdots & 0 & \mathbf{O}_1 \\ m_1 & q_0 & p & \cdots & 0 & \mathbf{O}_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{\gamma-2} & q_{\gamma-3} & q_{\gamma-4} & \cdots & p & \mathbf{O}_{\gamma-1} \end{bmatrix} \tag{6.11}$$

Observe that for $i < \gamma - 1$, $\mathbf{C}\mathbf{A}_{cl}^i \mathbf{B} = 0$, as the $E_{j,\gamma} = 0$ for $j = 1, \ldots, \gamma - 1$. Thus, the output y cannot be of relative degree less than γ. Assume for contradiction that the output is not relative degree γ. This implies:

$$L_{\mathbf{B}} L_{\mathbf{A}_{cl}\zeta}^{\gamma-1} y = \mathbf{C}\mathbf{A}_{cl}^{\gamma-1}\mathbf{B} = p = 1 - \mathbf{s}_{\eta_1}^\top \mathbf{a}_{\eta_2} = 0. \tag{6.12}$$

By Lemma 5, $\mathcal{S}$ is $\mathbf{A}_{cl}$ invariant - this implies the existence of $\mathbf{J} \in \mathbb{R}^{n_z \times n_z}$ such that $\mathbf{A}_{cl}\mathbf{S} = \mathbf{S}\mathbf{J}$, i.e. the image of $\mathbf{S}$ under $\mathbf{A}_{cl}$ is contained in $\mathbf{S}$. As $\mathbf{C}\mathbf{S} = \mathbf{0}$, and $\mathbf{C}\mathbf{A}_{cl}^k\mathbf{S} = \mathbf{C}\mathbf{S}\mathbf{J}^k = 0$ we have that $\mathbf{S} \in \ker(\mathbf{E})$. Furthermore, by (6.12) we have $\mathbf{C}\mathbf{A}_{cl}^{\gamma-1}\mathbf{B} = 0$, and $\mathbf{B} \in \ker(\mathbf{E})$. Finally, note that $\mathbf{B} \notin \mathbf{S}$, as $\mathbf{S}$ contains an identity block in the lower n_z rows, while $\mathbf{B}$ has a corresponding $\mathbf{0}$ block. This implies $\dim(\ker(\mathbf{E})) \geq \dim(\text{span}(\mathbf{S}, \mathbf{B})) = n_z + 1$. By Rank Theorem, $\text{rank}(\mathbf{E}) + \dim(\ker(\mathbf{E})) = n$, and therefore,

$$\text{rank}(\mathbf{E}) \leq n - (n_z + 1) = \gamma - 1.$$

Therefore, $\mathbf{E}$ is rank deficient, since it has γ rows. There is a nontrivial left nullspace; there exists $\mathbf{v} = [\alpha_0 \ \dots \ \alpha_{\gamma-1}]^\top$, such that $\mathbf{v}^\top \mathbf{E} = \mathbf{0}$. Without loss of generality[1], we take $\alpha_{\gamma-1} = 1$. As the rightmost block of $\mathbf{v}^\top \mathbf{E} = \mathbf{0}$,

$$\mathbf{O}_{\gamma-1} = -\sum_{i=0}^{\gamma-2} \alpha_i \mathbf{O}_i \tag{6.13}$$

Next, because $p = 0$ by (6.12), $\mathbf{v}^\top \mathbf{E}_{\gamma-2} = \alpha_{\gamma-1} q_0 = 0$, where $\mathbf{E}_i$ is the i'th column of $\mathbf{E}$. Therefore, $q_0 = 0$. Applying this logic on rows of $\mathbf{E}$ backward to $\mathbf{E}_2$ gives:

$$q_i = 0 \quad \forall i \in \{0, \dots, \gamma - 3\}.$$

Finally, we demonstrate via induction that $q_j = 0$, for any $j \in \mathbb{N}$. Assuming that $q_j = 0$, we aim to show that $q_{j+1} = -\mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^{j+1} \left(\mathbf{a}_{\eta_1} + \mathbf{A}_\mathbf{z} \mathbf{a}_{\eta_2}\right) = 0$. Introduce q_{j+1} by right-multiplying (6.13) by $\mathbf{A}_\mathbf{z}^k (\mathbf{a}_{\eta_1} + \mathbf{A}_\mathbf{z} \mathbf{a}_{\eta_2})$, for $k = j + 1 - (\gamma - 1)$:

$$\begin{aligned} q_{j+1} &= \mathbf{O}_{\gamma-1} \mathbf{A}_\mathbf{z}^k (\mathbf{a}_{\eta_1} + \mathbf{A}_\mathbf{z} \mathbf{a}_{\eta_2}) \\ &= -\sum_{i=0}^{\gamma-2} \alpha_i \mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^{i+k} \mathbf{a}_{\eta_1} - \sum_{i=0}^{\gamma-2} \alpha_i \mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^{i+k+1} \mathbf{a}_{\eta_2}. \end{aligned}$$

By the induction hypothesis $q_i = 0$, for $i = 0, \dots, j$ which can be rearranged:

$$-\mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^i \mathbf{a}_{\eta_1} = \mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^{i+1} \mathbf{a}_{\eta_2}.$$

Substituting this for terms $q_{j-(\gamma-1)}$ to q_j into the right sum:

$$q_{j+1} = -\sum_{i=0}^{\gamma-2} \alpha_i \mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^{i+k} \mathbf{a}_{\eta_1} + \sum_{i=0}^{\gamma-2} \alpha_i \mathbf{s}_{\eta_1}^\top \mathbf{A}_\mathbf{z}^{i+k} \mathbf{a}_{\eta_1} = 0.$$

And the inductive step has been shown. The base case holds trivially for $\gamma \geq 3$ by the structure of $\mathbf{E}$, and $\mathbf{CB} = 1$ for relative degree one systems (trivially valid relative degree). It remains to show the base case holds for $\gamma = 2$.

First, examine the fact that $\mathcal{S}$ is invariant under $\mathbf{A}_{cl}$. Expanding $\mathbf{A}_\mathrm{cl}\mathbf{S} = \mathbf{S}\mathbf{J}$, given that $\mathbf{K} = [k_{\eta_1} \ k_{\eta_2} \ \mathbf{k}_\mathbf{z}]$:

$$\begin{bmatrix} \mathbf{s}_{\eta_2}^\top \\ -k_{\eta_1} \mathbf{s}_{\eta_1}^\top - k_{\eta_2} \mathbf{s}_{\eta_2}^\top - \mathbf{k}_\mathbf{z} \\ \mathbf{a}_{\eta_1} \mathbf{s}_{\eta_1}^\top + \mathbf{a}_{\eta_2} \mathbf{s}_{\eta_2}^\top + \mathbf{A}_\mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{s}_{\eta_1}^\top \mathbf{J} \\ \mathbf{s}_{\eta_2}^\top \mathbf{J} \\ \mathbf{J} \end{bmatrix}$$

[1] $\mathbf{v}$ must have at least one nonzero term since the null space is nontrivial. If $\alpha_{\gamma-1} = 0$, this proof can be continued by redefining $\mathbf{E}$ to omit the last row (or as many as necessary such that the last element of $\mathbf{v}$ is nonzero). Then $\mathbf{v}$ can be scaled such that its last entry has magnitude 1.

Right multiply the second equation by $\mathbf{a}_{\eta_2}$, and using (6.12):

$$\mathbf{a}_{\eta_1} + \mathbf{A}_{\mathbf{z}}\mathbf{a}_{\eta_2} = (\mathbf{I} - \mathbf{a}_{\eta_2}\mathbf{s}_{\eta_1})\mathbf{J}\mathbf{a}_{\eta_2}$$

And note that left multiplying by $\mathbf{s}_{\eta_1}$ gives:

$$q_0 = \mathbf{s}_{\eta_1}^\top(\mathbf{a}_{\eta_1} + \mathbf{A}_{\mathbf{z}}\mathbf{a}_{\eta_2}) = \mathbf{s}_{\eta_1}^\top(\mathbf{I} - \mathbf{a}_{\eta_2}\mathbf{s}_{\eta_1}^\top)\mathbf{J}\mathbf{a}_{\eta_2} = 0$$

again leveraging $\mathbf{s}_{\eta_1}^\top\mathbf{a}_{\eta_2} = 1$. We have established the based case for induction, for relative degree $\gamma = 2$.

We now aim to demonstrate the contradiction, by showing that (6.12) leads to the system losing controllability. Consider the controllability matrix of the system. Controllability is preserved under feedback, so we examine $\mathscr{C}(\mathbf{A}, \mathbf{B})$:

$$\mathscr{C}(\mathbf{A}, \mathbf{B}) = \begin{bmatrix} \mathbf{0} & 1 & 0 & \dots & 0 \\ \mathbf{I} & 0 & 0 & \dots & 0 \\ \mathbf{0} & \mathbf{a}_{\eta_2} & \mathbf{Q}_0 & \dots & \mathbf{Q}_{n-(\gamma+1)} \end{bmatrix}$$
$$\mathbf{Q}_i = \mathbf{A}_{\mathbf{z}}^i(\mathbf{a}_{\eta_1} + \mathbf{A}_{\mathbf{z}}\mathbf{a}_{\eta_2})$$

We finish by showing the $\mathbf{C}$ is in the left null space of $\mathscr{C}$:

$$\mathbf{C}\mathscr{C} = \begin{bmatrix} 0 & p & q_0 & \dots & q_{n-(\gamma+1)} \end{bmatrix} = \mathbf{0}$$

And therefore $\mathscr{C}$ is not full rank. We have reached contradiction, and y must be relative degree γ. Finally, note that $\mathcal{S}$ is the zeroing surface associated with y by Lemma 4. □

Lemma 6 constructs an output and its associated zeroing manifold such that 1) the output has valid relative degree, and 2), the zeroing manifold is exponentially stable. Now, we show that the output locally retains both of these properties under the nonlinear dynamics:

Theorem 11. *Given a nonlinear system* (6.4)*, the output* $y = \eta_1 - \mathbf{s}_{\eta_1}^\top\mathbf{z}$ *obtained via linearization in Lemma 6 has valid relative degree and exponentially stable zero dynamics for the nonlinear system. As such, stabilizing* $\mathbf{e} \to 0$ *results in stability of the entire system in a neighborhood of the origin.*

Proof. First, we establish that the output is relative degree γ for the nonlinear system. The output has the form $y = \mathbf{C}\boldsymbol{\zeta} = \eta_1 - \mathbf{s}_{\eta_1}^\top \mathbf{z}$. By Lemma 3, we have that $L_{\mathbf{g}} L_{\mathbf{f}}^j y(\boldsymbol{\zeta}) \equiv 0$ for $j = 0, \dots, \gamma - 2$ and:

$$L_{\mathbf{g}} L_{\mathbf{f}}^{\gamma-1} y = 1 - \mathbf{s}_{\eta_1}^\top \frac{\partial \boldsymbol{\omega}}{\partial \eta_2}.$$

For relative degree, we need $L_{\mathbf{g}} L_{\mathbf{f}}^{\gamma-1} y \neq 0$.

$$\begin{aligned} L_{\mathbf{g}} L_{\mathbf{f}}^{\gamma-1} y &= 1 - \mathbf{s}_{\eta_1}^\top \frac{\partial \boldsymbol{\omega}}{\partial \eta_2} \\ &= \underbrace{1 - \mathbf{s}_{\eta_1}^\top \mathbf{a}_{\eta_2}}_{\mathbf{C}\mathbf{A}^{\gamma-1}\mathbf{B}} + \underbrace{\mathbf{s}_{\eta_1}^\top \left(\mathbf{a}_{\eta_2} - \frac{\partial \boldsymbol{\omega}}{\partial \eta_2} \right)}_{\triangleq \Delta(\boldsymbol{\zeta})} \end{aligned}$$

where $\Delta : \mathcal{X} \to \mathbb{R}$. Lemma 6 assures that $|\mathbf{C}\mathbf{A}^{\gamma-1}\mathbf{B}| \triangleq \delta > 0$. To guarantee $L_{\mathbf{g}} L_{\mathbf{f}}^{\gamma-1} y \neq 0$, we bound $|\Delta(\boldsymbol{\zeta})| < \delta$.

$$|\Delta(\boldsymbol{\zeta})| \leq \|\mathbf{s}_{\eta_1}\| \left\| \mathbf{a}_{\eta_2} - \frac{\partial \boldsymbol{\omega}}{\partial \eta_2} \right\|$$

Note that the function $\left\| \mathbf{a}_{\eta_2} - \frac{\partial \boldsymbol{\omega}}{\partial \eta_2} \right\|$ is continuous and zero at the origin. Therefore, there exists an $\varepsilon > 0$ such that

$$\left\| \mathbf{a}_{\eta_2} - \frac{\partial \boldsymbol{\omega}}{\partial \eta_2} \right\| \leq \frac{\delta}{2\|\mathbf{s}_{\eta_1}\|} \quad \forall\ (\boldsymbol{\eta}, \mathbf{z}) \in B_\varepsilon(\mathbf{0}, \mathbf{0})$$

Inside this epsilon ball, we have $|\Delta(\boldsymbol{\zeta})| \leq \frac{1}{2}\delta$, and therefore $|L_{\mathbf{g}} L_{\mathbf{f}}^{\gamma-1} y| > \frac{1}{2}\delta$. Locally, the output has valid relative degree for the nonlinear system.

It remains to show that the zeroing manifold, $\mathcal{M}_{\boldsymbol{\psi}}$ of this output is stable for the nonlinear system. First, given the error coordinates (6.8), consider $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\zeta}} = \begin{bmatrix} \frac{\partial \mathbf{e}}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{e}}{\partial \mathbf{z}} \end{bmatrix}$ evaluated at the origin. For a row of this matrix,

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\zeta}} \mathbf{e}_i \bigg|_{\zeta=\mathbf{0}} &= \frac{\partial}{\partial \boldsymbol{\zeta}} L_{\mathbf{f}}^i y \bigg|_{\zeta=\mathbf{0}} \\ &= \frac{\partial^2 L_{\mathbf{f}}^{(i-1)} y}{\partial \boldsymbol{\zeta}^2} \mathbf{f} \bigg|_{\zeta=\mathbf{0}} + \frac{\partial}{\partial \boldsymbol{\zeta}} L_{\mathbf{f}}^{(i-1)} y \frac{\partial \mathbf{f}}{\partial \boldsymbol{\zeta}} \bigg|_{\zeta=\mathbf{0}} \\ &= \frac{\partial}{\partial \boldsymbol{\zeta}} L_{\mathbf{f}}^{(i-1)} y \frac{\partial \mathbf{f}}{\partial \boldsymbol{\zeta}} \bigg|_{\zeta=\mathbf{0}} \\ &= \mathbf{C}\mathbf{A}^i \end{aligned} \tag{6.14}$$

where the third lines holds by $\mathbf{f}(\mathbf{0}) = \mathbf{0}$ and the last equality can be shown by induction. The base case holds as $\frac{\partial y}{\partial \zeta} = \mathbf{C}$, and the induction step is given by (6.14). We have:

$$\left.\frac{\partial \mathbf{e}}{\partial \boldsymbol{\zeta}}\right|_{\boldsymbol{\zeta}=\mathbf{0}} = \begin{bmatrix} \mathbf{C} \\ \vdots \\ \mathbf{C}\mathbf{A}^{\gamma-1} \end{bmatrix} \tag{6.15}$$

which is precisely the same as (6.11). Therefore, $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\eta}}$ is lower triangular with nonzero elements on the diagonal, and is thus invertible. By the implicit function theorem, there exists $\boldsymbol{\psi} : \mathcal{Z} \to \mathcal{N}$ such that $\mathbf{e}(\boldsymbol{\psi}(\mathbf{z}), \mathbf{z}) = \mathbf{0}$. We aim to show that $\mathcal{M}_{\boldsymbol{\psi}}$ is stable. Consider that by the implicit function theorem:

$$\frac{\partial \boldsymbol{\psi}}{\partial \mathbf{z}} = -\frac{\partial \mathbf{e}}{\partial \boldsymbol{\eta}}^{-1} \frac{\partial \mathbf{e}}{\partial \mathbf{z}}. \tag{6.16}$$

Furthermore, note that since $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\zeta}}|_{\boldsymbol{\zeta}=\mathbf{0}} = \mathbf{E}$, (6.16) also holds for the linearization. Therefore, at the origin, the tangent space of $\mathcal{M}_{\boldsymbol{\psi}}$ for the nonlinear system is equal to the invariant subspace used by the linearization to design the output, i.e. $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\eta}}|_{\boldsymbol{\zeta}=\mathbf{0}} = \mathbf{S}_{\boldsymbol{\eta}}$. On $\mathcal{M}_{\boldsymbol{\psi}}$ we have $\boldsymbol{\eta} = \boldsymbol{\psi}(\mathbf{z})$, or equivalently:

$$\begin{aligned} \boldsymbol{\eta} &= \left.\frac{\partial \boldsymbol{\psi}}{\partial \mathbf{z}}\right|_{\boldsymbol{\zeta}=\mathbf{0}} \mathbf{z} + \left(\boldsymbol{\psi}(\mathbf{z}) - \left.\frac{\partial \boldsymbol{\psi}}{\partial \mathbf{z}}\right|_{\boldsymbol{\zeta}=\mathbf{0}} \mathbf{z}\right) \\ &= \mathbf{S}_{\boldsymbol{\eta}}\mathbf{z} + \underbrace{\left(\boldsymbol{\psi}(\mathbf{z}) - \mathbf{S}_{\boldsymbol{\eta}}\mathbf{z}\right)}_{\boldsymbol{\Gamma}(\mathbf{z})} \end{aligned}$$

By mean value theorem, there exists $\varepsilon > 0$ such that inside a ball $B_\varepsilon(\mathbf{0}, \mathbf{0})$ we have

$$\|\Gamma(\mathbf{z})\| \leq N_\Gamma(\varepsilon)\|\mathbf{z}\|$$

With $N_\Gamma(\varepsilon) \to 0$ as $\varepsilon \to 0$.

Finally, given that the surface $\mathcal{S}$ is stable under the linearized dynamics from Lemma 5, converse Lyapunov guarantees the existence of a function $V_{\mathbf{z}}(\mathbf{z})$ satisfying

$$\begin{aligned} k_1\|\mathbf{z}\|^2 \leq V_{\mathbf{z}}(\mathbf{z}) &\leq k_2\|\mathbf{z}\|^2 \\ \dot{V}_{\mathbf{z}}(\mathbf{z}) &= \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \left(\mathbf{a}_{\eta_1}\mathbf{s}_{\eta_1} + \mathbf{a}_{\eta_2}\mathbf{s}_{\eta_2} + \mathbf{A}_{\mathbf{z}}\right)\mathbf{z} \\ &\leq -k_3\|\mathbf{z}\|^2 \\ \left\|\frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}}\right\| &\leq k_4\|\mathbf{z}\| \end{aligned}$$

for $k_i > 0$. We aim to show that this function is also a Lyapunov function for the dynamics on the nonlinear zeroing manifold, where $(\boldsymbol{\eta}, \mathbf{z}) = (\boldsymbol{\psi}(\mathbf{z}), \mathbf{z})$. To this end, define $\tilde{\mathbf{A}} \in \mathbb{R}^{n_z \times n_z}$ and $\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2 : \mathcal{Z} \to \mathbb{R}^{n_z}$:

$$\begin{aligned}
\tilde{\mathbf{A}} &= \mathbf{a}_{\eta_1} \mathbf{s}_{\eta_1} + \mathbf{a}_{\eta_2} \mathbf{s}_{\eta_2} + \mathbf{A}_{\mathbf{z}} \\
\boldsymbol{\Delta}_1(\mathbf{z}) &= \boldsymbol{\omega}\left(\mathbf{S}_{\boldsymbol{\eta}} \mathbf{z}, \mathbf{z}\right) - \tilde{\mathbf{A}} \mathbf{z} \\
\boldsymbol{\Delta}_2(\mathbf{z}) &= \boldsymbol{\omega}\left(\boldsymbol{\psi}(\mathbf{z}), \mathbf{z}\right) - \boldsymbol{\omega}\left(\mathbf{S}_{\boldsymbol{\eta}}, \mathbf{z}\right)
\end{aligned}$$

Differentiate $V_{\mathbf{z}}$ under the nonlinear dynamics:

$$\begin{aligned}
\dot{V}_{\mathbf{z}} &= \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \boldsymbol{\omega}\left(\boldsymbol{\psi}(\mathbf{z}), \mathbf{z}\right) \\
&= \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \tilde{\mathbf{A}} \mathbf{z} + \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \left(\boldsymbol{\Delta}_1(\mathbf{z}) + \boldsymbol{\Delta}_2(\mathbf{z})\right)
\end{aligned}$$

Noting that $\boldsymbol{\Delta}_1(\mathbf{z})$ is the error in the linearization of $\boldsymbol{\omega}$ on the linear surface $\mathcal{S}$, it can be bounded using mean value theorem as $\|\boldsymbol{\Delta}_1(\mathbf{z})\| \leq N_1(\varepsilon)\|\mathbf{z}\|$. Similarly, $\boldsymbol{\Delta}_2$ can be bounded using Lipschitz continuity of $\boldsymbol{\omega}$, and the fact that $\boldsymbol{\psi}(\mathbf{z}) - \mathbf{S}_{\boldsymbol{\eta}} \mathbf{z} = \boldsymbol{\Gamma}(\mathbf{z})$ to obtain $\|\boldsymbol{\Delta}_2(\mathbf{z})\| \leq L_{\boldsymbol{\omega}} N_{\Gamma}(\varepsilon)\|\mathbf{z}\| = N_2(\varepsilon)\|\mathbf{z}\|$. Letting $N(\varepsilon) = N_1(\varepsilon) + N_2(\varepsilon)$, we have

$$\begin{aligned}
\left\| \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \left(\boldsymbol{\Delta}_1(\mathbf{z}) + \boldsymbol{\Delta}_2(\mathbf{z})\right) \right\| &\leq \left\| \frac{\partial V_{\mathbf{z}}}{\partial \mathbf{z}} \right\| \left(\|\boldsymbol{\Delta}_1(\mathbf{z})\| + \|\boldsymbol{\Delta}_2(\mathbf{z})\|\right) \\
&\leq k_4 N(\varepsilon)\|\mathbf{z}\|^2
\end{aligned}$$

Choosing $\varepsilon > 0$ such that $N(\varepsilon) < \frac{k_3}{2k_4}$, we have

$$\begin{aligned}
\dot{V}_{\mathbf{z}}(\mathbf{z}) &\leq -k_3\|\mathbf{z}\|^2 + k_4 N(\varepsilon)\|\mathbf{z}\|^2 \\
&\leq -\frac{k_3}{2}\|\mathbf{z}\|^2
\end{aligned}$$

Therefore, we see that $V_{\mathbf{z}}$ is a Lyapunov function on the nonlinear zeroing manifold, with a sufficiently small ball around the origin. We have successfully demonstrated that $\boldsymbol{\psi}$ has the required properties to apply Theorem 10 and conclude local exponential stability of the composite system. □

This proof uses linearization to design an output which has desirable properties. Locally, the zeroing manifold for this output $\mathcal{M}_{\boldsymbol{\psi}}$ is close to the linear systems zeroing manifold (captured by $\boldsymbol{\Delta}_2$), and the zero dynamics are close to the dynamics of the linear system (captured by $\boldsymbol{\Delta}_1$). Sufficiently close to the origin, relative degree and stability of $\mathcal{M}_{\boldsymbol{\psi}}$ are retained for the nonlinear system.

It is important to emphasize that Theorem 11 gives a completely *constructive method for stabilizing a broad class of underactuated systems using output stabilization*. In the results section, we demonstrate this constructive method on a canonical example of underactuation, the cartpole.

6.4 Optimal Control for Learning Zero Dynamics Policies

While the linearization of a system about equilibrium can be used to construct locally stabilizing controllers, it is desirable to obtain larger regions of attraction, and leverage the nonlinear dynamics of the system. Therefore, we aim to construct $\mathcal{M}_{\psi}$ satisfying controlled invariance and stability.

Optimal Control for Stabilization

As asymptotic stability is a necessary condition for optimality [12], we leverage optimal control to find $\mathcal{M}_{\psi}$. Consider the infinite-time optimal control problem:

$$V(\zeta_0) \triangleq \min_{\zeta,u} \quad \int_0^{\infty} c(\zeta(t), u(t))dt \qquad (6.17)$$
$$\text{s.t.} \quad \dot{\zeta} = \mathbf{f}(\zeta) + \mathbf{g}(\zeta)u$$

where $V : \mathcal{X} \to \mathbb{R}$ is the value function and $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a positive definite cost function. In order to apply Theorem 10, we will require exponential stability on the manifold. Therefore, we begin by stating conditions under which the optimal controller is exponentially stabilizing:

Theorem 12. *Let $V(\zeta)$ be the value function for the optimal control problem defined* (6.17)*, with quadratic cost $c(\zeta, \mathbf{u}) = \zeta^{\top}\mathbf{Q}\zeta + ru^2$, with $\mathbf{Q} \in \mathbb{R}^{n\times n}$ positive definite, $r > 0$ and compact state space $\mathcal{X}$. The nonlinear system is exponentially stable under the optimal controller.*

Proof. In a sufficiently small ball around the origin, the LQR approximation of the optimal controller, obtained by linearizing the dynamics about equilibrium, will be exponentially stabilizing for the nonlinear system [16], as it locally satisfies input bounds. This implies constants $M_{LQR},\ \lambda_{LQR},\ \delta > 0$ such that:

$$\|\zeta_0\| \leq \delta \quad \implies \quad \|\zeta(t)\| \leq M_{LQR}e^{-\lambda_{LQR}t}\|\zeta_0\|$$

We aim to show that the trajectory emanating from an arbitrary initial condition $\zeta_0 \in B_{\delta}(\mathbf{0})$ is exponentially stable. For any $M, \lambda > 0$, consider the set:

$$T = \{t \geq 0 \mid \|\zeta(t)\| > Me^{-\lambda t}\|\zeta_0\|\}$$

We condition on whether there is an upper bound to the elements of T:

Case 1: There exists an upper bound $\overline{T}$ such that $t < \overline{T}$ for all $t \in T$. Then consider the maximum violation ratio

$$\overline{r} = \sup_{t \in T} \frac{\|\boldsymbol{\zeta}(t)\|}{Me^{-\lambda t}\|\boldsymbol{\zeta}_0\|} \leq \frac{B}{Me^{-\lambda \overline{T}}\|\boldsymbol{\zeta}_0\|}$$

Take $\overline{r} = 1$ if T is empty. Then:

$$\|\boldsymbol{\zeta}(t)\| \leq \overline{r} M e^{-\lambda t}\|\boldsymbol{\zeta}_0\|$$

implying the trajectory is exponentially stable.

Case 2: There is no upper bound on the elements in T. We will establish $V(\boldsymbol{\zeta})$ is a Lyapunov function certifying exponential stability of the trajectory. Bound the decrease:

$$\begin{aligned} \dot{V}(\boldsymbol{\zeta}) &= -\left(\boldsymbol{\zeta}^\top \mathbf{Q}\boldsymbol{\zeta} + ru^2\right) \\ &\leq -\underline{\lambda}(\mathbf{Q})\|\boldsymbol{\zeta}\|^2 \end{aligned} \tag{6.18}$$

Next, bound V above by a quadratic function. Because LQR is suboptimal for the nonlinear system, applying it can only increase the cost relative to $V(\boldsymbol{\zeta})$:

$$\begin{aligned} V(\boldsymbol{\zeta}_0) &\leq \int_0^\infty \boldsymbol{\zeta}^\top \mathbf{Q}\boldsymbol{\zeta} + r(\mathbf{K}\boldsymbol{\zeta})^2 dt \\ &\leq \int_0^\infty (\overline{\lambda}(\mathbf{Q}) + r\overline{\lambda}(\mathbf{K}^\top \mathbf{K}))\|\boldsymbol{\zeta}\|^2 dt \\ &\leq \int_0^\infty (\overline{\lambda}(\mathbf{Q}) + r\overline{\lambda}(\mathbf{K}^\top \mathbf{K}))M_{LQR}^2 e^{-2\lambda t}\|\boldsymbol{\zeta}_0\|^2 dt \\ &= \frac{(\overline{\lambda}(\mathbf{Q}) + r\overline{\lambda}(\mathbf{K}^\top \mathbf{K}))M_{LQR}^2}{2\lambda_{LQR}}\|\boldsymbol{\zeta}_0\|^2 \end{aligned}$$

with $\overline{\lambda}, \underline{\lambda}$ the maximum and minimum eigenvalues respectively. Finally, lower bound

$V(\boldsymbol{\zeta})$ by a quadratic.

$$\begin{aligned} V(\boldsymbol{\zeta}_0) &= \int_0^\infty \boldsymbol{\zeta}^\top \mathbf{Q}\boldsymbol{\zeta} + ru^2 dt \\ &\geq \int_T \underline{\lambda}(\mathbf{Q})\|\boldsymbol{\zeta}\|^2 dt \\ &\geq \underline{\lambda}(\mathbf{Q})M^2\|\boldsymbol{\zeta}_0\|^2 \left(\int_T e^{-2\lambda t} dt\right) \\ &= \underline{\lambda}(\mathbf{Q})M^2\|\boldsymbol{\zeta}_0\|^2 \left(\int_0^\infty e^{-2\lambda t} dt - \int_{\mathbb{R}_{\geq 0}\setminus T} e^{-2\lambda t} dt\right) \\ &= \underline{\lambda}(\mathbf{Q})M^2 \left(\frac{1}{2\lambda} - c\right) \|\boldsymbol{\zeta}_0\|^2 \end{aligned}$$

where $\mathbb{R}_{\geq 0}\setminus T$ is the set difference between the nonnegative reals and T, and $\frac{1}{2\lambda} - c > 0$ as both integrals integrate over the same strictly positive function, but the right integral does so over a smaller domain. The bounds hold at each point on the trajectory, and V is a Lyapunov function certifying exponential stability of the trajectory.

We now extend this claim over the compact state space $\mathcal{X}$. At $V \succ 0$ and $\dot{V} \prec 0$, we have that the optimal controller is asymptotically stabilizing [12]. By compactness of $\mathcal{X}$ and (6.18), the time for a trajectory to enter $B_\delta(\mathbf{0})$ is bounded by:

$$T_{\max} = \frac{\sup_{\boldsymbol{\zeta}_0 \in \mathcal{X}} V(\boldsymbol{\zeta}_0)}{\inf_{\boldsymbol{\zeta}_0 \in \mathcal{X}\setminus B_\delta(\mathbf{0})} \dot{V}(\boldsymbol{\zeta}_0)} \leq \frac{\sup_{\boldsymbol{\zeta}_0 \in \mathcal{X}} V(\boldsymbol{\zeta}_0)}{\underline{\lambda}(\mathbf{Q})\delta^2}$$

Because trajectories in $B_\delta(\mathbf{0})$ converge exponentially:

$$\|\boldsymbol{\zeta}(t)\| \leq Me^{-\lambda(t-T_{\max})}\|\boldsymbol{\zeta}(T_{\max})\| \quad \forall t > T_{\max}$$

By compactness of $\mathcal{X}$, trajectories are bounded by $\|\boldsymbol{\zeta}\| \leq B$, and the whole trajectory can be bounded exponentially:

$$\|\boldsymbol{\zeta}(t)\| \leq \frac{\max\{M, B\}e^{\lambda T_{\max}}}{\min\{1, \delta\}} e^{-\lambda t}\|\boldsymbol{\zeta}_0\|$$

The optimal controller is exponentially stabilizing. □

Learning Zero Dynamics Policies

We propose learning $\boldsymbol{\psi}$ so $\mathcal{M}_{\boldsymbol{\psi}}$ is controlled invariant and stable. Let $u^*(\boldsymbol{\zeta})$ solve the optimal control problem (6.17). Then $\mathcal{M}_{\boldsymbol{\psi}}$ is invariant under $u^*(\boldsymbol{\zeta})$ if for

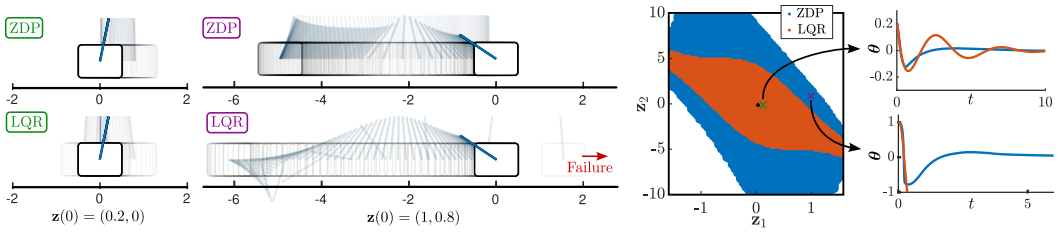


Figure 6.2: Zero Dynamics Policies (ZDPs) compared to LQR for the nonlinearly damped cartpole. Left: Simulated initial conditions for the cartpole, where LQR has slow response times and instabilities. Right: Region of attraction ($x = \dot{x} = 0$) for the two methods, as well as pendulum angle over time.

$\boldsymbol{\zeta}_{\boldsymbol{\psi}} = (\boldsymbol{\psi}(\mathbf{z}), \mathbf{z})$:

$$\begin{bmatrix} \mathbf{I} & -\frac{\partial \boldsymbol{\psi}}{\partial \mathbf{z}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{f}}(\boldsymbol{\zeta}_{\boldsymbol{\psi}}) + \hat{\mathbf{g}}(\boldsymbol{\zeta}_{\boldsymbol{\psi}}) u^*(\boldsymbol{\zeta}_{\boldsymbol{\psi}}) \\ \boldsymbol{\omega}(\boldsymbol{\zeta}_{\boldsymbol{\psi}}) \end{bmatrix} = \mathbf{0}$$

Given a neural network parameterization of $\boldsymbol{\psi}_{\boldsymbol{\theta}}$, we define the loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\mathbf{z} \sim D} \left\| \hat{\mathbf{f}}(\boldsymbol{\zeta}_{\boldsymbol{\theta}}) + \hat{\mathbf{g}}(\boldsymbol{\zeta}_{\boldsymbol{\theta}}) u^*(\boldsymbol{\zeta}_{\boldsymbol{\theta}}) - \frac{\partial \boldsymbol{\psi}_{\boldsymbol{\theta}}}{\partial \mathbf{z}} \boldsymbol{\omega}(\boldsymbol{\zeta}_{\boldsymbol{\theta}}) \right\| \tag{6.19}$$

for $\boldsymbol{\zeta}_{\boldsymbol{\theta}} = (\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}), \mathbf{z})$. Zero loss implies invariance of $\mathcal{M}_{\boldsymbol{\psi}}$ under the optimal control which gives stability, by Theorem 12. We minimize this loss using stochastic gradient descent.

Application to the Cartpole

We deploy the ZDPs on a classic underactuated system: the cartpole. We add nonlinear damping to the base coordinates of the form $d(\dot{x}) = \sigma(\dot{x})\dot{x}$ where $\sigma(\dot{x}) = 0$ if $|\dot{x}| < 1e^{-3}$ and 1 otherwise. This helps explore the effect of nonlinearities on the degradation of LQR performance, and how our nonlinear method compares. We take $\mathbf{Q} = \mathbf{I}$ and $\mathbf{r} = 0.01$. The ZDP policy was trained in the JAX module using iLQR to approximate u^* and its gradient for training. A 2 layer, 256 neuron feedforward neural network with ReLU activations was pretrained with LQR and then minimized (6.19). We stabilized $\mathcal{M}_{\boldsymbol{\psi}}$ with a PD controller and a feedforward term. Our code can be found at [4].

Observe the performance of LQR versus ZDPs in Figure 6.2. Even for initial conditions close to the origin and within the domain of attraction of LQR, the modified cartpole's unstable nonlinear damping significantly slowed the controller's by inducing oscillations. In comparison, ZDPs have smoother behavior and a larger region of attraction.

6.5 Discussion

This chapter established a theoretical foundation for Zero Dynamics Policies as a constructive method for stabilizing underactuated systems. The central result is that for any locally controllable nonlinear system, there exists a mapping from unactuated to actuated states such that stabilizing to the induced manifold guarantees full-state stability. The proof is constructive: linearization yields an explicit ZDP valid near the origin, and optimal control extends this to larger domains.

ZDPs exemplify the second strategy for structure-aware learning identified in Chapter 1: learning inputs to structured controllers. The optimal controller provides stability guarantees through Theorem 12; learning provides the manifold parameterization that makes this structure applicable globally. The invariance loss (6.19) has the same pointwise character as the Lyapunov and barrier losses in Chapters 2 and 3: we sample states and penalize local condition violations rather than rolling out trajectories. Like the Koopman methods of Chapter 4, ZDPs seek coordinates that simplify control, though ZDPs exploit underactuation structure to reduce dimension rather than lifting to achieve linearity.

Several limitations constrain the present results. The analysis assumes continuous-time dynamics without impacts or mode switches, whereas legged locomotion and dexterous manipulation involve hybrid dynamics. The cartpole demonstration is canonical but does not exhibit these phenomena. The existence proof provides only local guarantees; while optimal control extends the region empirically, we lack theoretical characterization of how large this region can be.

Subsequent chapters address these limitations. Chapter 7 applies barrier functions on zero dynamics for bipedal locomotion with hardware validation on AMBER-3M. Chapter 8 develops discrete-time ZDPs for hybrid systems with impacts, achieving robust hopping on the ARCHER platform. These extensions demonstrate that the continuous-time foundations established here translate to hybrid robotic systems. Application to dexterous manipulation, where object pose forms the unactuated coordinates, remains unexplored future work.

References

[1] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. “Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics”. In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[2] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. USA: Princeton University Press, 2008. ISBN: 0691135762.

[3] Nazareth Sarkis Bedrossian. "Nonlinear Control Using Linearizing Transformations". PhD thesis. Massachusetts Institute of Technology, 1991.

[4] "Code". In: (2024). URL: %7Bhttps://github.com/ivandariojr/LearnedZeroDynamicsPolicies%7D.

[5] Noel Csomay-Shanklin, Victor D Dorobantu, and Aaron D Ames. "Nonlinear Model Predictive Control of a 3D Hopping Robot: Leveraging Lie Group Integrators for Dynamically Stable Behaviors". In: *2023 IEEE (ICRA)*. IEEE. 2023, pp. 12106–12112.

[6] S. Devasia, Degang Chen, and B. Paden. "Nonlinear inversion-based output tracking". en. In: *IEEE TAC* 41.7 (July 1996), pp. 930–942. ISSN: 00189286. DOI: 10.1109/9.508898. (Visited on 03/22/2024).

[7] Robert J Full and Daniel E Koditschek. "Templates and anchors: neuromechanical hypotheses of legged locomotion on land". In: *Journal of experimental biology* 202.23 (1999), pp. 3325–3332.

[8] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. "Perceptive locomotion through nonlinear model-predictive control". In: *IEEE Transactions on Robotics* 39.5 (2023), pp. 3402–3421.

[9] A. Isidori and C.I. Byrnes. "Output regulation of nonlinear systems". en. In: *IEEE TAC* 35.2 (Feb. 1990), pp. 131–140. ISSN: 00189286. DOI: 10.1109/9.45168. (Visited on 03/22/2024).

[10] Alberto Isidori. "Elementary Theory of Nonlinear Feedback for Single-Input Single-Output Systems". en. In: *Nonlinear Control Systems*. Ed. by Alberto Isidori. Communications and Control Engineering. London: Springer, 1995, pp. 137–217. ISBN: 978-1-84628-615-5. DOI: 10.1007/978-1-84628-615-5_4. (Visited on 01/30/2024).

[11] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot". In: *Autonomous robots* 40 (2016), pp. 429–455.

[12] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012. ISBN: 978-0-691-15187-8. DOI: 10.2307/j.ctvcm4g0s. (Visited on 01/31/2024).

[13] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science robotics* 7.62 (2022), eabk2822.

[14] Marc H Raibert. “Hopping in legged systems—modeling and simulation for the two-dimensional one-legged case”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 3 (1984), pp. 451–463.

[15] Ivan Dario Jimenez Rodriguez, Noel Csomay-Shanklin, Yisong Yue, and Aaron D. Ames. “Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies”. en. In: *Proceedings of The 4th Annual L4DC*. ISSN: 2640-3498. PMLR, May 2022, pp. 1060–1072. (Visited on 03/22/2024).

[16] Shankar Sastry. “Linearization by State Feedback”. en. In: *Nonlinear Systems: Analysis, Stability, and Control*. Ed. by Shankar Sastry. Interdisciplinary Applied Mathematics. New York, NY: Springer, 1999, pp. 384–448. ISBN: 978-1-4757-3108-8. DOI: 10.1007/978-1-4757-3108-8_9. (Visited on 10/15/2023).

[17] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. “Blind bipedal stair traversal via sim-to-real reinforcement learning”. In: *arXiv preprint arXiv:2105.08328* (2021).

[18] Eduardo D Sontag. “A ‘universal’construction of Artstein’s theorem on nonlinear stabilization”. In: *Systems & control letters* 13.2 (1989), pp. 117–123.

[19] Eduardo D. Sontag. “Control-Lyapunov functions”. en. In: *Open Problems in Mathematical Systems and Control Theory*. Ed. by Vincent Blondel, Eduardo D. Sontag, Mathukumalli Vidyasagar, and Jan C. Willems. Springer, 1999, pp. 211–216. ISBN: 978-1-4471-0807-8. DOI: 10.1007/978-1-4471-0807-8_40. (Visited on 03/22/2024).

[20] Michael Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. en. Mathematics monograph series. CRC Press, Taylor & Francis Group, 2018. ISBN: 978-0-8053-9021-6.

[21] C. Tomlin, J. Lygeros, L. Benvenuti, and S. Sastry. “Output tracking for a non-minimum phase dynamic CTOL aircraft model”. en. In: *Proceedings of 1995 34th IEEE Conference on Decision and Control*. Vol. 2. New Orleans, LA, USA: IEEE, 1995, pp. 1867–1872. ISBN: 978-0-7803-2685-9. DOI: 10.1109/CDC.1995.480615. (Visited on 03/22/2024).

[22] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. “Hybrid zero dynamics of planar biped walkers”. In: *IEEE Transactions on Automatic Control* 48.1 (Jan. 2003), pp. 42–56. ISSN: 1558-2523. DOI: 10.1109/TAC.2002.806653.

Chapter 7

LEARNING BIPEDAL LOCOMOTION VIA CBFS AND ZDPS

7.1 Introduction

This chapter extends the Zero Dynamics Policy framework developed in Chapter 6 to hybrid dynamical systems with impacts, demonstrating stable bipedal locomotion on hardware. Chapter 6 established that for continuous-time underactuated systems, there exist mappings from unactuated to actuated states such that stabilizing to the induced manifold guarantees full-state stability. Walking robots present a more challenging setting: the dynamics alternate between continuous evolution and discrete impact events as feet make and break contact with the ground. This chapter develops a learning-based approach for synthesizing Zero Dynamics Policies that satisfy control barrier function certificates across both continuous and discrete phases of locomotion.

Bipedal locomotion poses fundamental challenges due to the compounded complexity of underactuation and hybrid dynamics. Underactuation makes the application of classic nonlinear control approaches challenging, necessitating the use of offline optimization to generate periodic walking gaits. Due to the combinatorics of contact conditions resulting from the hybrid dynamics, feasibility of this optimization problem requires either fixing the contact times and positions (which can be vulnerable to perturbations) or expensive planning through the set of possible contact points. Pushing this offline optimization problem online allows for reactive controllers but requires the use of reduced-order models that limit formal guarantees. Despite impressive implementations that achieve bipedal walking in practice, general bipedal locomotion with formal performance guarantees remains an open problem.

Prior Work in Control. In the control literature, bipedal locomotion follows two general branches: walking with guarantees of stability and predictive control approaches. Walking with guarantees usually relies on solving optimization programs offline to generate stable (periodic) gaits [13]. Above all, this approach relies on constraining walking to be a periodic orbit with assumed exponential stability on the underactuated coordinates of the robot. This underlying assumption can be problematic in safety critical settings when the gait must satisfy hard constraints such as staying on predetermined stepping stones [9, 21] and also precludes dif-

ferent walking modes such as period-two walking and, more generally, aperiodic locomotion [32, 4]. This is particularly important given that disturbance rejection can require aperiodic behaviors [24].

Predictive control approaches avoid these limitations by planning trajectories and/or policies online, and have shown great promise for quadrupedal robots [10, 11]. Their application to bipedal robots is comparatively sparse, and has predominantly required static stability [29, 26] or simplified models to mitigate the computational complexity [17, 5, 31]. This leads to challenges when seeking formal guarantees for dynamic bipedal locomotion in the presence of model mismatch between the planning and low-level control layers.

Prior Work in Learning. Prior work in machine learning has produced impressive results towards realizing legged locomotion using reinforcement learning [19, 27, 6, 12]. These methods use relatively simple reward functions along with sophisticated simulations to generate large amounts of data to train a policy capable of traversing a variety of terrains. Still, these algorithms can be fragile when facing environments outside of the training dataset and are data inefficient due to not exploiting the full dynamics structure. These challenges make it difficult to reliably apply these methods on complex hardware systems. Other approaches use reinforcement learning to train parameterizations of CBF-CLF Quadratic Program controllers [8, 9]. The approach developed here differs in that we learn the projection of modeling error onto the zero dynamics rather than onto CLF and CBF constraints, and we specify barrier conditions that imply walking as emergent behavior rather than tracking a desired trajectory.

Contributions. The Neural Gaits framework synthesizes walking policies by training on pointwise barrier conditions rather than optimizing a reward function. The Barrier Loss defined in Definition 18 penalizes pointwise violations of the CBF condition across sampled states; zero loss implies forward invariance of the safe set (Theorem 14). This approach connects directly to Chapter 6: the learned policy parameters $\boldsymbol{\theta}$ define Zero Dynamics Policies through the output function $\boldsymbol{y}_d(z;\boldsymbol{\theta})$, with full-state stability following from Theorem 15 when the barrier conditions are satisfied.

Three benefits follow from this formulation. First, training policies over a buffered guard region $\mathcal{S}_\epsilon$ rather than a nominal impact surface certifiably handles impact timing uncertainty. Second, operating on the two-dimensional zero dynamics rather than the ten-dimensional full state improves data efficiency while retaining formal

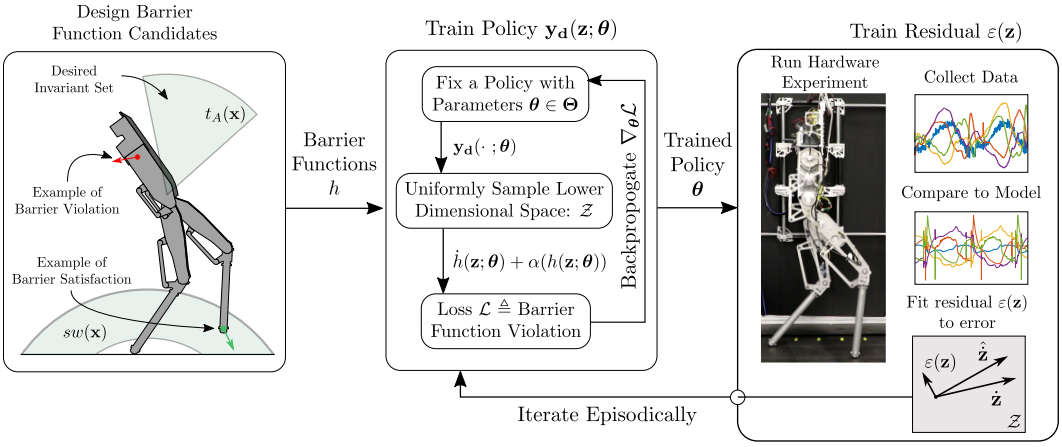


Figure 7.1: A depiction of the Neural Gaits framework. **Left:** Designing barrier function candidates that formally describe walking. **Middle:** Training a policy capable of satisfying all barrier conditions in the zero dynamics state space. **Right:** Collecting hardware data to train a residual zero dynamics model, then refining the policy episodically.

guarantees for the full-order system (Section 7.3). Third, characterizing walking as set invariance rather than convergence to a periodic orbit accommodates aperiodic locomotion and multi-period gaits.

The framework consists of two learning modules iterated episodically. The first trains a policy to minimize barrier condition violations using the Monte Carlo approach of LyaNet [25], extended from Lyapunov to barrier functions. The second trains a residual dynamics model on hardware data using Neural ODEs [7] to correct for model mismatch. Experiments on the AMBER-3M platform [1] demonstrate walking under significant model uncertainty, with barrier satisfaction improving across episodes. To the best of our knowledge, this is the first demonstration of integrated learning and control for bipedal locomotion with stability guarantees.

7.2 Preliminaries

We provide a brief introduction of zero dynamics, hybrid dynamical systems, and control barrier functions, which are necessary fundamentals to understand the proposed formulation in Section 7.3.

Output and Zero Dynamics

Consider the general nonlinear ordinary differential equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{7.1}$$

with states $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$, inputs $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$, and dynamics $\boldsymbol{f} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ with $\boldsymbol{f}$ locally Lipschitz in both arguments. For mechanical systems, we specialize to the control-affine case:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}, \tag{7.2}$$

where $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^n$ and $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^{n \times m}$ are assumed to be locally Lipschitz. Denoting a parameter in a parameter space $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, we can define a collection of k outputs $\boldsymbol{y} : \mathcal{X} \times \boldsymbol{\Theta} \to \mathbb{R}^k$ parameterized by $\boldsymbol{\theta}$ that we would like to converge to zero as:

$$\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{y}_{\boldsymbol{a}}(\boldsymbol{x}) - \boldsymbol{y}_{\boldsymbol{d}}(\boldsymbol{x}; \boldsymbol{\theta}), \tag{7.3}$$

where $\boldsymbol{y}_{\boldsymbol{a}} : \mathcal{X} \to \mathbb{R}^k$ are the measured outputs, and $\boldsymbol{y}_{\boldsymbol{d}} : \mathcal{X} \times \boldsymbol{\Theta} \to \mathbb{R}^k$ are the desired outputs. For locomotion, the outputs are typically taken either as joint angles (as done in this work), or as center of mass and foot positions. For the policy $\boldsymbol{y}_d$ learned in this work and shown in the center block of Figure 7.1, $\boldsymbol{\theta}$ corresponds to neural network parameters. Although all the concepts may be extended to systems with valid decomposition into output and zero dynamics coordinates (which includes all mechanical systems), for simplicity the remainder of the exposition will be restricted to the setting used in this work, namely with $k = 4$ and $\boldsymbol{y}_a$ taken to be the actuated joint angles of the robot. For a complete description of output coordinates and zero dynamics, we refer to [16].

Given these outputs $\boldsymbol{y}$, we can separate the actuated and the unactuated coordinates for the robot, which are shown in Figure 7.2a. As these outputs are *vector relative degree* 2, we can define error coordinates $\boldsymbol{\eta}_i : \mathcal{X} \to \mathcal{N}_i \subseteq \mathbb{R}^2$ for $i = 1, \dots 4$ as $\boldsymbol{\eta}_i = \begin{bmatrix} y_i^\top, & \dot{y}_i^\top \end{bmatrix}^\top$, as well as the collection of errors $\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}_1^\top, & \dots, & \boldsymbol{\eta}_4^\top \end{bmatrix}^\top$. Then, there exist 2 linearly independent functions $z_i : \mathcal{X} \to \mathcal{Z}_i \subseteq \mathbb{R}$ for $i = 1, 2$ such that $\nabla_{\mathbf{x}} z_i(\boldsymbol{x}) g(\boldsymbol{x}) \equiv 0$, and $\nabla_{\boldsymbol{x}} z_i(\boldsymbol{x})$ is linearly independent from $\nabla_{\boldsymbol{x}} \eta_{i,j}(\boldsymbol{x})$ for $i = 1, \dots, 4$ and $j = 1, 2$. We can then construct a diffeomorphism[1] $\boldsymbol{\Phi} : \mathcal{X} \times \boldsymbol{\Theta} \to \mathcal{N} \times \mathcal{Z}$:

$$\begin{bmatrix} \boldsymbol{\eta} \\ z \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{\eta}}(\boldsymbol{x}; \boldsymbol{\theta}) \\ \boldsymbol{\Phi}_z(\boldsymbol{x}) \end{bmatrix} \triangleq \boldsymbol{\Phi}(\boldsymbol{x}; \boldsymbol{\theta}), \qquad \boldsymbol{x} = \boldsymbol{\Phi}^{-1}\left(\begin{bmatrix} \boldsymbol{\eta} \\ z \end{bmatrix}; \boldsymbol{\theta} \right),$$

[1]. This is differentiable in the first argument and differentiable almost everywhere in the second argument.

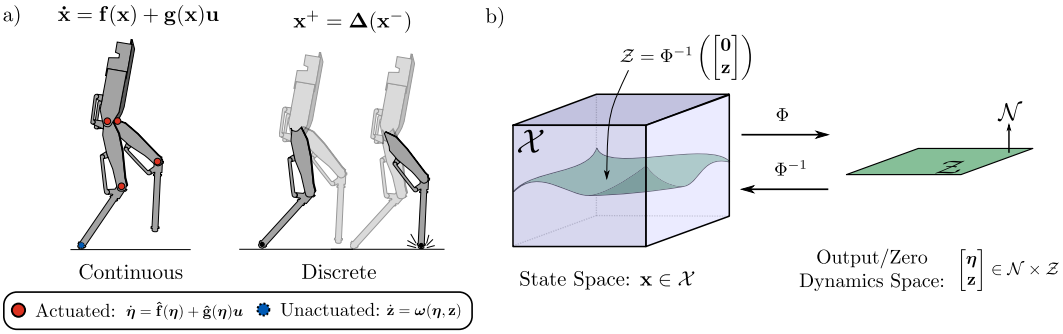


Figure 7.2: **a)** The continuous and discrete phases of the robot, with actuated ($\boldsymbol{\eta}$) and unactuated (z) coordinates depicted. **b)** The diffeomorphism $\boldsymbol{\Phi}$, and the relationship between state space coordinates and output/zero dynamics coordinates.

as shown in Figure 7.2b. Under this coordinate transformation, the system dynamics become:

$$\begin{bmatrix}\dot{\boldsymbol{\eta}}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}\hat{\boldsymbol{f}}(\boldsymbol{\eta};\boldsymbol{\theta}) + \hat{\boldsymbol{g}}(\boldsymbol{\eta};\boldsymbol{\theta})\boldsymbol{u}\\ \boldsymbol{\omega}(\boldsymbol{\eta}, z;\boldsymbol{\theta})\end{bmatrix}, \tag{7.4}$$

where $\hat{\boldsymbol{f}}, \hat{\boldsymbol{g}}$ and $\boldsymbol{\omega}$ are the projection of the dynamics through the diffeomorphism $\boldsymbol{\Phi}$.

The *zero dynamics manifold* $\mathcal{Z} \subset \mathcal{X}$ is thus the space where errors have been driven to zero:

$$\mathcal{Z} = \{\boldsymbol{x} \in \mathcal{X} : \ \boldsymbol{\eta}(\boldsymbol{x}) = 0\},$$

as seen in Figure 7.2a. Observe that for $z \in \mathcal{Z}$, we have that $\dot{z} = \boldsymbol{\omega}(\mathbf{0}, z;\boldsymbol{\theta})$. The power of the method of zero dynamics lies in that it allows for guarantees about the full nonlinear dynamics by considering only a subspace of significantly smaller dimensionality [16]. Notice that although the input does not appear in the zero dynamics $\boldsymbol{\omega}$ in (7.4), the parameters of the policy $\boldsymbol{\theta}$ do. This realization motivates the use of the policy as a way to influence the zero dynamics and enforce the desired barrier functions, as introduced below. Finally, in this work we will learn residual dynamics $\varepsilon(z)$ on the zero dynamics manifold for a corrected zero dynamics $\dot{z} = \boldsymbol{\omega}(\mathbf{0}, z) + \varepsilon(z)$ that compensate for modeling error. This process is captured in the episodic iteration of Figure 7.1.

Hybrid Dynamics

Walking consists of continuous evolution with discrete impacts occurring as contact is made and broken (e.g, the feet with the ground). This sequence of continuous

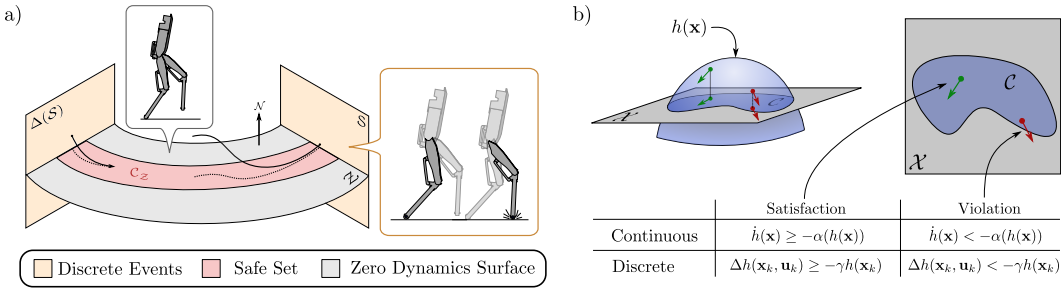


Figure 7.3: **a)** The guard $\mathcal{S}$, reset map $\Delta(\mathcal{S})$, and safe set $C_{\mathcal{Z}}$ are visualized in the state space decomposed into output $\mathcal{N}$ and zero dynamics $\mathcal{Z}$ coordinates. **b)** A safe set defined as regions of the state space (gray square) where h is positive (blue region). Satisfying the CBF condition implies that the flows/discrete updates of the system will not leave the safe set (although may approach the boundary). Violations imply a flow that could potentially leave the safe set.

and discrete dynamics is shown in Figure 7.3a can be modeled in the language of *hybrid systems* as:

$$\mathscr{HC} = \begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} & \boldsymbol{x} \in \mathcal{D} \backslash \mathcal{S} \\ \boldsymbol{x}^+ = \Delta(\boldsymbol{x}^-) & \boldsymbol{x} \in \mathcal{S} \subset \mathcal{D}, \end{cases}$$

where $\mathcal{D} \subset \mathcal{X}$ is the domain where $\boldsymbol{x}(t)$ evolves. The *guard*, $\mathcal{S} \subset \mathcal{X}$, corresponds to the set of states where the foot comes in contact with the floor. The *reset map*, $\Delta : \mathcal{S} \to \mathcal{D}$ models the instantaneous sign flip of velocities observed when two rigid bodies collide (the foot with the ground). Furthermore, $\mathscr{HC}$ can be projected through the diffeomorphism $\boldsymbol{\Phi}$ to exploit the decomposition into output and zero dynamics. For more details, we refer to [30].

Control Barrier Function Certificates

Barrier function certificates allow us to make the notion of *safety* rigorous in the context of the dynamical system in Equation (7.1). We begin by specifying a set that we wish to render safe:

$$C = \{\boldsymbol{x} \in \mathcal{X} : h(\boldsymbol{x}) \geq 0\} \subset \mathcal{X}, \tag{7.5}$$

where $h : \mathcal{X} \to \mathbb{R}$ is a continuously differentiable function. In the case of bipedal walking, safe sets can describe conditions such as admissible torso angles and reasonable foot placement as shown in the first block of Figure 7.1. We assume that the compact set C is nonempty, has a non-empty interior, and does not contain isolated fixed-points. We say that C is *safe* or *forward invariant* if $x(t_0) \in C$ implies that $x(t) \in C$ for all $t \geq t_0$. With this, we have the following condition for safety (see [3] for a brief history of this approach):

Definition 17 (*Control Barrier Function (CBF), [2]*). *Consider C as defined in Equation* (7.5) *where the continuously differentiable function h has nonvanishing gradients $Dh(\boldsymbol{x}) \neq 0$ for all $\boldsymbol{x}$ in the boundary of C defined as $\partial C = \{\boldsymbol{x} \in \mathcal{X} : h(\boldsymbol{x}) = 0\}$. The function h is a Control Barrier Function (CBF) for Equation* (7.1) *on C if there exists $\alpha \in \mathcal{K}_{\infty,e}$ such that for all $\boldsymbol{x} \in C$:*

$$\dot{h}(\boldsymbol{x}) = \frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \geq -\alpha(h(\boldsymbol{x})). \tag{7.6}$$

A function α is in the family of class-$\mathcal{K}_{\infty,e}$ functions if for all $a < b$, $\alpha(a) < \alpha(b)$, $\alpha(0) = 0$, $\lim_{a\to\infty} \alpha(a) = \infty$ and $\lim_{a\to-\infty} \alpha(a) = -\infty$. In defining the CBF, we can parameterize the set of all feedback controllers guaranteeing safety as:

$$K_{cbf}(\boldsymbol{x}) = \left\{\boldsymbol{u} \in \mathcal{U} : \dot{h}(\boldsymbol{x}, \boldsymbol{u}) \geq -\alpha(h(\boldsymbol{x}))\right\}. \tag{7.7}$$

Similarly, this notion can be extended to discrete-time dynamical systems via:

$$\Delta h(\boldsymbol{x}_k, \boldsymbol{u}_k) \triangleq h(\boldsymbol{x}_{k+1}) - h(\boldsymbol{x}_k) \geq -\gamma h(\boldsymbol{x}_k), \;\; 0 < \gamma \leq 1, \tag{7.8}$$

as seen in Figure 7.3b. This leads to the following necessary and sufficient condition for safety:

Theorem 13 (*Control Barrier Function Certificates, [2]*). *Given a feedback controller $\boldsymbol{u} = k(\boldsymbol{x})$, the set C is safe if and only if $\boldsymbol{u}(\boldsymbol{x}) \in K_{cbf}(\boldsymbol{x})$.*

7.3 Neural Gaits: Locomotion as a Barrier Satisfiability Problem

We now present our Neural Gaits approach, as depicted in Figure 7.1. Instead of taking a controller-design perspective, we will take one of reference trajectory design – specifically, we fix a controller structure $\boldsymbol{u}(\boldsymbol{x}; \boldsymbol{\theta})$, which is parameterized by $\boldsymbol{\theta}$ through the definition of $\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{\theta})$. Our method relies on the assumption that good walking can be characterized as a forward invariant set. Thus, the first step of the method requires us to define a set of barrier functions that imply good walking. In the following discussion, we will only consider barrier functions defined on the zero dynamics surface, i.e. $h : \mathcal{Z} \to \mathbb{R}$ as defined when the error coordinates are zero ($\boldsymbol{\eta} = \boldsymbol{0}$). Importantly, the guarantees made on $\mathcal{Z} \subset \mathcal{X}$ have relevance to the full state space, as is made precise in Section 7.3.

After constructing a collection of barrier functions, we train a policy $\boldsymbol{y_d}$ that ensures the system stays safe by minimizing the violation of the barrier function conditions (7.6) and (7.8) over regions of the state space. The resulting policy renders the

intersection of the safe set for all barriers forward invariant. Finally, to mitigate model mismatch, we train a residual term $\varepsilon(z)$ on the zero dynamics. These corrected zero dynamics are then used to refine the existing policy episodically until the desired walking performance is achieved.

Learning the Policy y_d

Our learning approach builds upon and unifies two lines of work. The first studies how to characterize good walking behavior as set invariance via a collection of barrier function candidates [4]. The second studies how to train neural ODEs to satisfy control-theoretic properties such as Lyapunov stability [25], which we extend to the barrier setting.

Learning in the Zero Dynamics.

Recall from Section 7.2 that the error and zero dynamics coordinates are computed from the states using the diffeomorphism $\mathbf{\Phi}$, which only depends on the policy through $\mathbf{\Phi}_{\eta}$. We thus parameterize the policy as a function of the projection of the state onto the zero dynamics manifold and parameters $\boldsymbol{\theta}$, i.e. $\boldsymbol{y_d}(\boldsymbol{x}) = \boldsymbol{y_d}(\mathbf{\Phi}_z(\boldsymbol{x}); \boldsymbol{\theta})$. In other words, $\boldsymbol{y}_d$ only depends on the unactuated degrees of freedom of the system (e.g., the unactuated joint in Figure 7.2) rather than the full state. Therefore, when there is no error (i.e. $\boldsymbol{\eta} = 0$) we have that $z \in \mathcal{Z}$ with dynamics $\dot{z} = \omega(\mathbf{0}, z; \boldsymbol{\theta})$. Note, importantly, that even when the error coordinates are zero, the zero dynamics are still a function of $\boldsymbol{y_d}$ and therefore $\boldsymbol{\theta}$. This implies that the zero dynamics are influenced by the parameters of the policy even though the control inputs are not present in ω.

Learning to Satisfy Barrier Conditions.

Taking inspiration from [14] and [4], we assume that walking can be characterized as set invariance via a collection of barrier function candidates $\mathcal{H} = \{h_i\}_{i=1}^{N}$ (see Table 7.1 discussed later in Section 7.3). To each h_i, we associate a *region at risk* $\overline{\mathcal{S}}_i \subseteq \mathcal{Z}$ where the barrier function is enforced. We define a set of neural network parameters that render the region at risk safe under the barrier definition:

$$\mathbf{\Theta}_i = \{\boldsymbol{\theta} \in \mathbf{\Theta} : \forall_{z \in \overline{\mathcal{S}}_i} \quad \dot{h}_i(z; \boldsymbol{\theta}) \geq -\alpha(h_i(z; \boldsymbol{\theta}))\}. \tag{7.9}$$

In other words, each $\mathbf{\Theta}_i$ corresponds to the set of policy parameters that render the set $\overline{\mathcal{S}}_i$ safe.

Thus, our learning problem is equivalent to finding a set of parameters $\boldsymbol{\theta} \in \bigcap_{i=1}^{N} \boldsymbol{\Theta}_i$ that render the system safe in all regions at risk. Similar to the Lyapunov Loss studied in [25], we introduce the concept of *Barrier Loss* as a learning signal for training:

Definition 18 (Barrier Loss). *For a set of barrier function candidates $\mathcal{H} = \{h_i\}_{i=1}^{N}$ and corresponding regions at risk $\overline{\mathcal{S}}_i \subset \mathcal{Z}$ on the zero dynamics, a Barrier Loss, $\mathcal{L} : \boldsymbol{\Theta} \to \mathbb{R}_{\geq 0}$, is defined as:*

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \int_{\overline{\mathcal{S}}_i} \max\{0, -\dot{h}_i(z;\boldsymbol{\theta}) - \alpha(h_i(z;\boldsymbol{\theta}))\} dz. \tag{7.10}$$

When a choice of parameters $\boldsymbol{\theta}$ achieves zero Barrier Loss, then the safety of the zero dynamics is guaranteed by satisfying the forward invariance condition of the barrier functions:

Theorem 14 (Zero Barrier Loss Implies Safety of Zero Dynamics). *The zero dynamics is guaranteed to be safe in all its regions at risk if and only if we find a $\boldsymbol{\theta}^*$ that attains $\mathcal{L}(\boldsymbol{\theta}^*) = 0$.*

Proof. Notice that for all $i \in \{1 \dots N\}$ both $\dot{h}_i$ and $\alpha \circ h_i$ are continuous functions. This implies that for all $z \in \mathcal{Z}$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $\max\{0, -\dot{h}_i(z;\boldsymbol{\theta}) - \alpha(h_i(z;\boldsymbol{\theta}))\}$ is a continuous non-negative real function. It is well known that a continuous non-negative real function will have zero integral if and only if it is the zero function. We specialize this statement for the terms in our loss as follows:

$$\forall_{z \in \overline{\mathcal{S}}_i} \max\{0, -\dot{h}_i(z;\boldsymbol{\theta}) - \alpha(h_i(z;\boldsymbol{\theta}))\} = 0 \Leftrightarrow \int_{\overline{\mathcal{S}}_i} \max\{0, -\dot{h}_i(z;\boldsymbol{\theta}) - \alpha(h_i(z;\boldsymbol{\theta}))\} dz = 0 \tag{7.11}$$

It is clear that the sum in $\mathcal{L}(\boldsymbol{\theta})$ will be zero if an only if each integral term is zero since each integral is a non-negative function. Thus we can conclude that $\mathcal{L}(\boldsymbol{\theta}^*) = 0$ if and only if

$$\forall_{i \in \{1 \dots N\}, z \in \overline{\mathcal{S}}_i} \max\{0, -\dot{h}_i(z;\boldsymbol{\theta}^*) - \alpha(h_i(z;\boldsymbol{\theta}^*))\} = 0.$$

For any barrier h_i and $z \in \mathcal{Z}$ you can see that $\max\{0, -\dot{h}_i(z;\boldsymbol{\theta}^*) - \alpha(h_i(z;\boldsymbol{\theta}^*))\} = 0$ implies that:

$$-\dot{h}_i(z;\boldsymbol{\theta}^*) - \alpha(h_i(z;\boldsymbol{\theta}^*)) \leq 0 \implies \dot{h}_i(z;\boldsymbol{\theta}^*) \geq -\alpha(h_i(z;\boldsymbol{\theta}^*)),$$

i.e. the safety condition for the barrier is satisfied. □

Instantiation for Bipedal Walking

Table 7.1 describes the barrier functions used in our experiments, which take inspiration from [4]. We depict some of these conditions on the robot in Figure 7.4a. As all barrier functions $h_i : \mathcal{X} \to \mathbb{R}$ are enforced on the zero dynamics surface, we will write them implicitly as $h_i \circ \mathbf{\Phi}^{-1}(\mathbf{0}, \cdot\,; \boldsymbol{\theta}) : \mathcal{Z} \to \mathbb{R}$ for $i \in \{1 \dots N\}$ with $N = 5$ in this instantiation. In Table 7.1, t_A represents the torso angle, and p_x and p_z represent the x and z position of the swing foot, respectively. In addition to continuous time conditions, various conditions needed to be enforced on the guard, namely enforcing the location of the guard, symmetry of the model before and after impact, and a guard mapping condition. Interestingly, although these barriers would be relative degree two on the full state dynamics, they are directly enforceable as relative degree one barriers on the zero dynamics. This can be seen by treating $\boldsymbol{y_d}$ as the input to the zero dynamics, and observing that the zero dynamics themselves are functions of $\boldsymbol{y_d}$.

Note that these barrier functions h are defined over the space $\mathcal{Z}$, as, given a policy $\boldsymbol{y_d}(\,\cdot\,; \boldsymbol{\theta}) : \mathcal{Z} \to \mathbb{R}^4$, the mapping $\mathbf{\Phi}^{-1} : \mathcal{N} \times \mathcal{Z} \to \mathcal{X}$ is uniquely defined. We take inspiration from reduced order models, and specifically the notion of orbital energy [23] to define a set $\mathcal{Z}_O \subset \mathcal{Z}$ with reasonably bounded orbital energies as our first region at risk. We also define the set $\mathcal{S}_\epsilon \subset \mathcal{Z}_O$ which contains the part of the guard in $\mathcal{Z}_O$ as well as a small region around it where discrete-time guard conditions are enforced. We learn policies that satisfy the barrier conditions on these regions of the zero dynamics by penalizing the violation of the constraints shown in Table 7.1. Notice that penalizing guard constraints over a region results in policies that are robust to impact modeling error since the policy must be prepared to change stance foot at any point in $\mathcal{S}_\epsilon$ rather than just the guard $\mathcal{S}$.

Torso Angle	$\{z \in \mathcal{Z}_O\}$	$-\frac{\pi}{10} \leq \theta_t(z) \leq 0.05$
Swing Foot Clearance	$\{z \in \mathcal{Z}_O\}$	$0 \leq (p_x(z) - c_x)^2 + (p_z(z) - c_z)^2 - r^2 \leq 0.3$
Impact Mapping	$\{z \in \mathcal{S}_\epsilon\}$	$-0.15 \leq \Delta(z) + z \leq 0.15$
Symmetry	$\{z \in \mathcal{S}_\epsilon\}$	$\boldsymbol{y}(z) = \boldsymbol{y}(\Delta(z))$
Foot on Guard	$\{z \in \mathcal{S}_\epsilon\}$	$p_z(z) = 0$

Table 7.1: Barrier functions used to characterize bipedal walking, and the associated regions at risk in which they are enforced. The first two are enforced over the continuous dynamics, and the bottom three in a buffered region of the guard. The strict equality on symmetry and the foot on guard conditions were also enforced as a training loss.

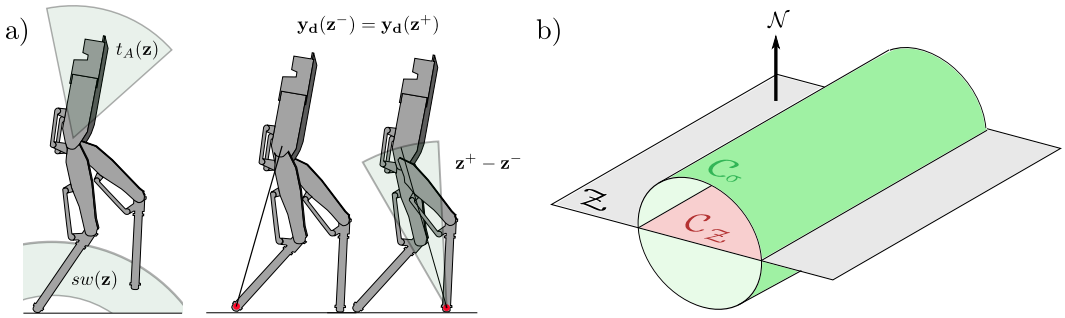


Figure 7.4: **a)** A depiction of the barrier functions used to enforce walking as set invariance. On the left are the two continuous time barrier conditions, and on the right the three barrier conditions enforced at the guard. The red dot on the foot indicates the stance foot, and **b)** The safe set on the zero dynamics $C_\mathcal{Z}$, as certified by the proposed learning method, and the combined safe set C_σ, and described in Theorem 5.

Learning Optimization Details. Evaluating the Barrier Loss in Equation (7.10) requires solving an integral that is in general intractable. Instead, we use Monte Carlo sampling to approximate the integral. Since our approach follows Algorithm 1 of [25] we refer to it for more details while noting that we optimize for the Barrier Loss rather than the Lyapunov Loss. A key ingredient in the Monte Carlo sampling approach in [25] is defining a compact support set to sample from (i.e., where the barrier condition should be satisfied). In our work this compact support set directly corresponds to the region at risk for each barrier condition.

Learning the Residual Zero Dynamics $\varepsilon(z)$

As outlined in Figure 7.1, we can improve upon the nominal zero dynamics model by collecting trajectories of the robot executing the resulting policy in hardware. We can then use those trajectories to learn a residual error term on the zero dynamics $\hat{\dot{z}} = \omega(0, z; \boldsymbol{\theta}) + \varepsilon(z)$ where ε is the learned residual term. We model this residual term using Neural ODEs [7], which are naturally compatible with our policy learning approach. We can iterate this process multiple times, alternating between learning $\boldsymbol{\theta}$ and ε until the resulting policy achieves the desired behavior.

Providing Guarantees in the Full State Space

Assuming a controller which exponentially converges the outputs $\boldsymbol{y}(\boldsymbol{x})$, for example a feedback linearizing or control Lyapunov function based controller, the converse Lyapunov theorem allows us to construct a Lyapunov function $V_{\boldsymbol{\eta}} : \mathcal{N} \to \mathbb{R}$ verifying the exponential convergence of the outputs. Along with a certificate of safety $h_Z : \mathcal{Z} \to \mathbb{R}$ on the zero dynamics space, we can construct a set in the combined

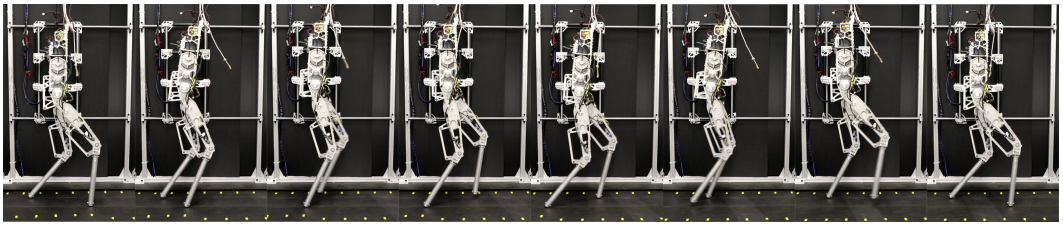


Figure 7.5: Gait tiles of the neural network encoding the final trained policy running in real time on the AMBER-3M robot. For a video discussing the methodology and summarizing the hardware results, please refer to [28].

space $\mathcal{N} \times \mathcal{Z}$ which is safe, and has a barrier function certificate. This is described in the following theorem.

Theorem 15. *Let $V_{\boldsymbol{\eta}} = \boldsymbol{\eta}^\top \boldsymbol{P} \boldsymbol{\eta} : \mathcal{N} \to \mathbb{R}$ be an exponential control Lyapunov function for the output dynamics with $\dot{V}_{\boldsymbol{\eta}} \leq -\gamma V_{\boldsymbol{\eta}}$ and $h_Z : \mathcal{Z} \to \mathbb{R}$ be a barrier function on the zero dynamics with safe set $C_{\mathcal{Z}}$. Then, there exists a constant $\sigma \geq 0$ and $c \geq 0$ such that if $\dot{h}_Z(z) \geq -\alpha h_Z(z) + c$ with $\alpha \leq \frac{\gamma}{2}$, the barrier function $h(\boldsymbol{\eta}, z) = h_Z(z) - \sigma V_{\boldsymbol{\eta}}(\boldsymbol{\eta})$ is safe with set C_σ.*

Proof. First note that the derivative of the function is given by:

$$\begin{aligned}
\dot{h} &= \frac{\partial h_Z}{\partial z}(z)\boldsymbol{w}(\boldsymbol{\eta}, z) - \sigma \dot{V}_{\boldsymbol{\eta}}(\boldsymbol{\eta}) \\
&\geq -\alpha h_Z(z) + c - \left| \frac{\partial h_Z}{\partial z}(z) \left(\boldsymbol{w}(\boldsymbol{\eta}, z) - \boldsymbol{w}(0, z)\right) \right| + \sigma\gamma V_{\boldsymbol{\eta}}(\boldsymbol{\eta}) \\
&\geq -\alpha h(\boldsymbol{\eta}, z) + c - L_{h_Z} L_{\omega_{\boldsymbol{\eta}}} \|\boldsymbol{\eta}\|_2 + \frac{\sigma\gamma}{2} \lambda_{min}(\boldsymbol{P}) \|\boldsymbol{\eta}\|_2^2,
\end{aligned} \tag{7.12}$$

where the third line follows from Cauchy Schwartz, the fact that h_Z and $\boldsymbol{\omega}(\boldsymbol{\eta}, z)$ are locally Lipschitz with Lipschitz constants L_{h_Z} and $L_{\omega_{\boldsymbol{\eta}}}$, respectively, converse Lyapunov, and the assumption that $\alpha \leq \frac{\gamma}{2}$. Taking $\beta_1 = L_{h_Z} L_{\omega_{\boldsymbol{\eta}}}$, and $\beta_2 = \frac{\gamma}{2}\lambda_{min}(\boldsymbol{P})$, we observe that $-\beta_1 \|\boldsymbol{\eta}\|_2 + \sigma\beta_2 \|\boldsymbol{\eta}\|_2^2 \geq -\frac{\beta_2^2}{4\sigma\beta_3} \triangleq c$. By taking c defined as such, we achieve the desired result. □

The above theorem motivates the perspective of this work: satisfying barrier function certificates in the zero dynamics enables reasoning about safe sets in the complete state space. Note that the hybrid case is not addressed here, and is an interesting direction for future theoretical work.

7.4 Simulation and Experimental Results

The hardware platform used in this work was the planar underactuated biped AMBER-3M [1], which has actuators on the hips and knees, and point contact

feet. Both in simulation, where the RaiSim [15] environment was used, and on hardware, the pipeline went as follows: the zero dynamics coordinate z was estimated, the Neural Network policy $\boldsymbol{y}_d(z;\boldsymbol{\theta})$ was evaluated, and the desired output values were passed to a PD controller running at 1kHz. The policy $\boldsymbol{y}_d(z;\boldsymbol{\theta})$ was randomly initialized and was trained for 1000 epochs. The AdamW optimizer was used in PyTorch [22] with an initial learning rate of 10^{-2}, weight decay of 10^{-4}, with a learning rate decay schedule at epochs 100, 400, and 800. Initially, the "gait" had the robots leg flailing randomly in the air, and when integrated resulted in the robot falling over. Once the loss converged, the policy had a loss in the order of 5×10^{-3}, and was able to walk stably in the simulation. The neural network ran in closed loop on the hardware platform and was called at approximately 500 Hz to produce desired outputs for the system to track. Unlike simulation, once tested on hardware, the policy resulted in the robot stumbling forward, unable to walk without falling. Data was collected over various trials, after which the methodology proposed in Section 7.3 was used to learn the residual of the model uncertainty, as projected to the zero dynamics space. During this process, Adam and other SGD methods were numerically unstable even with gradient clipping, so Nero [20] was used instead.

Once a residual term was learned, a new policy $\boldsymbol{y}_d(z,\boldsymbol{\theta})$ was trained with the updated dynamics (warm started with the policy from the previous iteration). After convergence, the gait was again tried on hardware. The gait was significantly more stable, and able to walk without assistance; however, the gait was not robust to walking speeds. Therefore, the process was repeated, and again a new policy was learned. When testing that policy, the robot was able to walk on its own, and was robust to different walking speeds. A sample gait is shown on Figure 7.5. The complete code can be found here [18].

7.5 Discussion

This chapter demonstrated that Zero Dynamics Policies can be synthesized for hybrid locomotion systems by training on pointwise barrier conditions. The Barrier Loss extends the Lyapunov Loss of Chapter 2 to forward invariance, and Theorem 15 provides the link between zero dynamics safety and full-state guarantees that parallels Theorem 10 from Chapter 6. The key extension to hybrid systems is enforcing barrier conditions over regions rather than along trajectories: training over the buffered guard region $\mathcal{S}_\epsilon$ handles impact timing uncertainty that would violate guarantees under trajectory-based certification.

The two-dimensional zero dynamics of AMBER-3M illustrate why this reduction is appropriate for underactuated locomotion. The passively evolving coordinates (torso angle and velocity) are precisely those that cannot be directly commanded, and the PD controller drives the actuated joints to track the learned policy outputs. Stability on the zero dynamics manifold propagates to the full system through the composite Lyapunov construction of Theorem 15, reducing both the learning problem and the certification problem to the unactuated subspace.

The formal guarantees of Theorem 15 assume a controller achieving exponential output convergence; the PD controller used in practice provides this approximately but without formal certificate. The theorem also does not address hybrid dynamics, as noted in the chapter. Chapter 8 extends Zero Dynamics Policies to more aggressive hybrid dynamics using a discrete-time formulation, validating that these foundations transfer to platforms with flight phases and repeated impacts.

Appendix 7.A

7.A.1 Derivation of Zero Dynamics Coordinates

Let $\boldsymbol{q} \in \mathcal{Q} \subset \mathbb{R}^n$ the configuration coordinates of the robot, with full state given by $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}}) \in T\mathcal{Q} \subset \mathbb{R}^{2n}$. Assuming a no-slip condition, the constrained robot dynamics can be derived via the Lagrange D'Alembert principle:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{D}^{-1}(\boldsymbol{q})\left(-\boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{B}\boldsymbol{u}\right) \end{bmatrix}$$

where $\boldsymbol{D} : \mathcal{Q} \to \mathbb{R}^{n \times n}$ is the mass-inertia matrix, $\boldsymbol{B} : T\mathcal{Q} \to \mathbb{R}^n$ contains the Coriolis and Gravity terms, $\boldsymbol{B} \in \mathbb{R}^n \times \mathbb{R}^m$ is the actuation matrix, and $\boldsymbol{u} \in \mathbb{R}^m$ is the control input vector. Because of the mechanical system structure of the system, there exists a well defined diffeomorphism $\Phi : \mathbb{R}^{10} \to \mathbb{R}^{10}$ with coordinates $[\boldsymbol{\eta}, \boldsymbol{z}] = \boldsymbol{\Phi}(\boldsymbol{x})$ such

that:

$$\begin{bmatrix} \boldsymbol{\eta_1} \\ \boldsymbol{\eta_2} \\ z_1 \\ z_2 \end{bmatrix} = \boldsymbol{\Phi}(\boldsymbol{x}) := \begin{bmatrix} \boldsymbol{Hq} - \boldsymbol{y}_d(z) \\ \boldsymbol{H\dot{q}} - \dot{\boldsymbol{y}}_d(z) \\ \boldsymbol{cq} \\ \boldsymbol{D}(\boldsymbol{q})\dot{\boldsymbol{q}} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \boldsymbol{\Phi}^{-1}\left(\begin{bmatrix} z \\ \boldsymbol{\eta} \end{bmatrix}\right) := \begin{bmatrix} \boldsymbol{\varphi}(z) \\ \boldsymbol{\psi}(z) \end{bmatrix} := \begin{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{H} \end{bmatrix}^{-1} \begin{bmatrix} z_1 \\ \boldsymbol{y}_d(z) + \boldsymbol{\eta_1} \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{ND}(\boldsymbol{\varphi}(z)) \\ \frac{\partial \boldsymbol{y}_d}{\partial z_1}(z)\boldsymbol{c}^\top - \boldsymbol{H} \end{bmatrix}^{-1} \begin{bmatrix} z_2 \\ \frac{\partial \boldsymbol{y}_d}{\partial z_2}(z) + \boldsymbol{\eta_2} \end{bmatrix} \end{bmatrix}$$

where $\boldsymbol{c}$ and $\boldsymbol{H}$ are chosen such that $\begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{H} \end{bmatrix}$ is full rank. The guard is defined as:

$$\mathcal{S} = \{(\boldsymbol{\eta}, z) \in \mathbb{R}^{10} : p_{sw}(\boldsymbol{\Phi}^{-1}(z, \boldsymbol{\eta})) = 0\}$$

The zero dynamics can be written as:

$$\dot{z} = \boldsymbol{\omega}(\boldsymbol{\eta}, z) := \begin{bmatrix} \boldsymbol{c\psi}(\boldsymbol{\eta}, z) z_2 \\ \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{q_1}}|_{\boldsymbol{\theta}=\boldsymbol{\varphi}(\boldsymbol{\eta},z)} \end{bmatrix}, \quad z^+ = \Delta_Z(z^-) = ND$$

References

[1] Eric Ambrose, Wen-Loong Ma, Christian Hubicki, and Aaron D. Ames. "Toward benchmarking locomotion economy across design configurations on the modular robot: AMBER-3M". In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. Aug. 2017, pp. 1270–1276. DOI: 10.1109/CCTA.2017.8062633.

[2] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876.

[3] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. "Control Barrier Functions: Theory and Applications". en. In: *2019 18th European Control Conference (ECC)*. Naples, Italy: IEEE, June 2019, pp. 3420–3431. ISBN: 978-3-907144-00-8. DOI: 10.23919/ECC.2019.8796030. URL: https://ieeexplore.ieee.org/document/8796030/ (visited on 04/14/2021).

[4] Aaron D. Ames, Paulo Tabuada, Austin Jones, Wen-Loong Ma, Matthias Rungger, Bastian Schürmann, Shishir Kolathaya, and Jessy W. Grizzle. "First steps toward formal controller synthesis for bipedal robots with experimental implementation". en. In: *Nonlinear Analysis: Hybrid Systems* 25 (Aug. 2017), pp. 155–173. ISSN: 1751570X. DOI: 10.1016/j.nahs.2017.01.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S1751570X1730002X (visited on 12/05/2021).

[5] Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan W Hurst. "Fast Online Trajectory Optimization for the Bipedal Robot Cassie." In: *Robotics: Science and Systems*. Vol. 101. 2018, p. 14.

[6] Guillermo A. Castillo, Bowen Weng, Wei Zhang, and Ayonga Hereid. "Robust Feedback Motion Policy Design Using Reinforcement Learning on a 3D Digit Bipedal Robot". In: *CoRR* abs/2103.15309 (2021). arXiv: 2103.15309. URL: https://arxiv.org/abs/2103.15309.

[7] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. "Neural Ordinary Differential Equations". In: *Neural Information Processing Systems (NeurIPS)*. 2019. URL: http://arxiv.org/abs/1806.07366.

[8] Jason Choi, Fernando Castaneda, Claire J Tomlin, and Koushil Sreenath. "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions". In: *arXiv preprint arXiv:2004.07584* (2020).

[9] Noel Csomay-Shanklin, Ryan K Cosner, Min Dai, Andrew J Taylor, and Aaron D Ames. "Episodic learning for safe bipedal locomotion with control barrier functions and projection-to-state safety". In: *Learning for Dynamics and Control*. PMLR. 2021, pp. 1041–1053.

[10] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[11] Ruben Grandia, Farbod Farshidian, René Ranftl, and Marco Hutter. "Feedback mpc for torque-controlled legged robots". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4730–4737.

[12] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. "Emergence of locomotion behaviours in rich environments". In: *arXiv preprint arXiv:1707.02286* (2017).

[13] Ayonga Hereid and Aaron D. Ames. "FROST: Fast Robot Optimization and Simulation Toolkit". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ. Vancouver, BC, Canada, Sept. 2017.

[14] Shao-Chen Hsu, Xiangru Xu, and Aaron D Ames. "Control barrier function based quadratic programs with application to bipedal robotic walking". In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 4542–4548.

[15] Jemin Hwangbo, Joonho Lee, and Marco Hutter. "Per-Contact Iteration Method for Solving Contact Dynamics". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 895–902. DOI: 10.1109/LRA.2018.2792536.

[16] Alberto Isidori. "Elementary Theory of Nonlinear Feedback for Single-Input Single-Output Systems". en. In: *Nonlinear Control Systems*. Ed. by Alberto Isidori. Communications and Control Engineering. London: Springer, 1995, pp. 137–217. ISBN: 978-1-84628-615-5. DOI: 10.1007/978-1-84628-615-5_4. (Visited on 01/30/2024).

[17] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot". In: *Autonomous robots* 40 (2016), pp. 429–455.

[18] *Learning Code*. URL: https://github.com/ivandariojr/NeuralGaits (visited on 12/07/2021).

[19] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. "Learning quadrupedal locomotion over challenging terrain". In: *Science Robotics* 5.47 (Oct. 2020). Publisher: American Association for the Advancement of Science, eabc5986. DOI: 10.1126/scirobotics.abc5986. URL: https://www.science.org/doi/10.1126/scirobotics.abc5986 (visited on 12/07/2021).

[20] Yang Liu, Jeremy Bernstein, Markus Meister, and Yisong Yue. "Learning by turning: Neural architecture aware optimisation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6748–6758.

[21] Quan Nguyen and Koushil Sreenath. "Safety-Critical Control for Dynamical Bipedal Walking with Precise Footstep Placement**This work is partially supported through funding from the Google Faculty Award and NSF Grant IIS-1464337." In: *IFAC-PapersOnLine* 48.27 (2015). Analysis and Design of Hybrid Systems ADHS, pp. 147–154. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.11.167. URL: https://www.sciencedirect.com/science/article/pii/S2405896315024258.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F.

d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035.

[23] Jerry E Pratt and Sergey V Drakunov. "Derivation and application of a conserved orbital energy for the inverted pendulum bipedal walking model". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 4653–4660.

[24] Marc H. Raibert and Ernest R. Tello. "Legged Robots That Balance". In: *IEEE Expert* 1.4 (1986), pp. 89–89. DOI: 10.1109/MEX.1986.4307016.

[25] Ivan Dario Jimenez Rodriguez, Aaron D. Ames, and Yisong Yue. *LyaNet: A Lyapunov Framework for Training Neural ODEs*. 2022. DOI: 10.48550/ARXIV.2202.02526. URL: https://arxiv.org/abs/2202.02526.

[26] Nicola Scianca, Daniele De Simone, Leonardo Lanari, and Giuseppe Oriolo. "MPC for humanoid gait generation: Stability and feasibility". In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1171–1188.

[27] Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan W. Hurst. "Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition". In: *CoRR* abs/2011.01387 (2020). arXiv: 2011.01387. URL: https://arxiv.org/abs/2011.01387.

[28] *Supplementary Video*. URL: https://www.youtube.com/watch?v=8TeXd0AYtpA (visited on 12/07/2021).

[29] Russ Tedrake, Scott Kuindersma, Robin Deits, and Kanako Miura. "A closed-form solution for real-time ZMP gait generation and feedback stabilization". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, pp. 936–940.

[30] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*. en. 1st ed. CRC Press, Oct. 2018. ISBN: 978-1-315-21942-4. DOI: 10.1201/9781420053739. URL: https://www.taylorfrancis.com/books/9781420053739 (visited on 12/05/2021).

[31] Xiaobin Xiong and Aaron Ames. "3d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization". In: *arXiv preprint arXiv:2101.09588* (2021).

[32] Xiaobin Xiong and Aaron D Ames. "Orbit characterization, stabilization and composition on 3d underactuated bipedal walking via hybrid passive linear inverted pendulum model". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4644–4651.

Chapter 8

ROBUST AGILITY VIA LEARNED ZERO DYNAMICS POLICIES

8.1 Introduction

The preceding chapters developed methods for training neural networks to satisfy pointwise conditions that imply global stability and safety guarantees. Those approaches addressed continuous-time systems where Lyapunov and barrier function conditions can be evaluated directly along flows. Hybrid systems with impacts present a distinct challenge: discontinuities at contact events disrupt continuous-time certificates, and underactuation prevents arbitrary shaping of the post-impact dynamics. This chapter extends the thesis framework to this setting by exploiting the zero dynamics decomposition as a structural scaffold for learning.

The underactuated dynamics inherent to legged locomotion impose fundamental limits on controller performance and necessitate understanding of the system's flow to achieve complex behaviors. Underactuation prevents arbitrarily shaping a system's dynamics, undermining the assumptions of many control-theoretic methods such as feedback linearization [25] and offline trajectory tracking. This work leverages recent advances in controller design for underactuated systems [24, 7], optimal control [18], and their integration with computational learning methods to design feedback strategies that exploit the structure of underactuation, enabling the agile and robust behavior shown in Figure 8.1.

A predominant method for controlling underactuated systems is Model Predictive Control (MPC) [4, 19], which leverages concepts from optimal control over a prediction horizon to achieve stabilization [30]. Performance of MPC controllers improves with longer horizons and finer time discretizations, both of which conflict with its strict real-time computational requirements. To address the high computational cost of full-model optimization problems, some methods leverage a gradation of model fidelities along a time horizon [15, 16]. Other methods rely on offline trajectory optimization to generate desirable behaviors, and then track these behaviors online [31]. For underactuated systems, the online tracking problem can be non-trivial, often requiring additional feedback mechanisms to stabilize the underactuated states such as regulators [22].

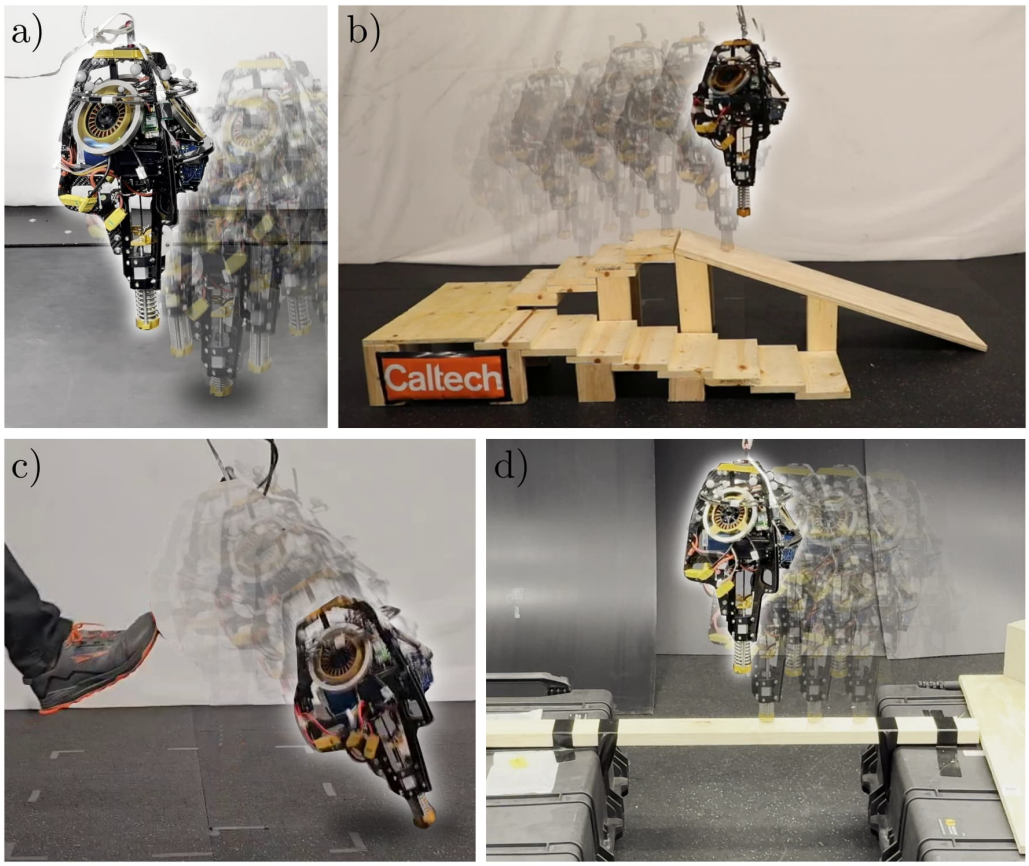


Figure 8.1: Experiments run with Zero Dynamics Policies: a) treadmill hopping with disturbances up to 1 mile per hour, b) 1.5" stair climbing and 20° ramp descending, c) disturbance rejection, and d) hopping across a 2x4.

Reinforcement learning (RL) [26] takes the concept of offline computation even further, using concepts from stochastic optimal control and parallelized simulation environments to synthesize feedback controllers. RL methods have shown robust performance [20, 17] when the policy is trained in sufficiently randomized domains. Current methods in RL improve policies through simulator rollouts [27], typically at the expense of high data complexity. Although these can work well, they exhibit extreme sensitivity to cost function parameters and ignore the underlying system structure.

Heuristics, on the other hand, are able to leverage intuition about system structure, and can achieve stabilization with minimal online or offline computational overhead. In the context of legged locomotion, the Raibert Heuristic for hopping [21], inverted pendulum models for walking [14], and spring-loaded pendulums for running [12] all reason about where a legged robot's feet should be placed in order to stabilize

the center of mass. While these methods may be less formal than the methods above and require significant domain expertise to implement, they tend to reason (perhaps implicitly) about the fundamental control structure needed to address the underactuation.

The above methods generally intersect in two places: first, an application of feedback to the actuated states based on the position of underactuated states (either explicitly or through replanning), and second, a dependence on optimality to generate stable, desirable behaviors. We propose a method which combines these two ideas, using optimality to ensure stability while reasoning explicitly about the structure of underactuation. Specifically, we leverage the notion of *zero dynamics* to explicitly decompose the system into actuated and unactuated coordinates [13, 32, 23, 10]. We pair this paradigm with optimal control to learn a mapping from the unactuated state to a desired actuated state, termed a Zero Dynamics Policy (ZDP), which is then stabilized using a tracking controller. This perspective aligns with prior work on Hybrid Zero Dynamics (HZD) [32]; however, rather than assuming stability of the zero dynamics manifold or relying on phasing variables and periodicity, we use optimal control to provably and constructively synthesize stable output-zeroing manifolds.

The key insight connecting this work to the thesis framework is that stability of hybrid systems can be converted into a pointwise training objective. Rather than training on Lyapunov conditions evaluated along continuous flows, we train on manifold invariance: the learned surface $\mathcal{M}_\psi$ must satisfy $(\eta_{k+1}, z_{k+1}) \in \mathcal{M}_\psi$ whenever $(\eta_k, z_k) \in \mathcal{M}_\psi$ under optimal control. This condition can be evaluated at sampled states without trajectory rollouts, paralleling the Monte Carlo training approach developed in earlier chapters. Theorem 16 then guarantees that convergence of the pointwise loss implies global stability of the full-order system.

We propose a general framework for the control of hybrid underactuated systems and apply it to hopping, which exemplifies the challenges of such systems due to the large number of passive degrees of freedom, tight input constraints, and short ground phases. Our empirical validation of ZDPs on the ARCHER 3D hopping robot showcases an agile and stable controller as seen in Figure 8.1 and the supplemental video [28]. Over the course of more than 3000 hops, our method achieves state of the art disturbance rejection, hops over long distances on a treadmill, navigates an obstacle course and rough terrain without vision, and is precise enough to reliably hop across narrow bridges.

8.2 Preliminaries

Hybrid Dynamics and Lyapunov Stability

Consider an $n-$degree of freedom robotic system with coordinates $\mathbf{q} \in \mathcal{Q}$ and state $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{X} \triangleq \mathsf{T}\mathcal{Q}$. Using the Euler Lagrange equations, we write the continuous-time dynamics in control-affine form as:

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{D}(\mathbf{q})^{-1}\mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix}}_{\mathbf{f}(\mathbf{x})} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{D}(\mathbf{q})^{-1}\mathbf{B} \end{bmatrix}}_{\mathbf{g}(\mathbf{x})} \mathbf{u} \tag{8.1}$$

where $\mathbf{D} : \mathcal{Q} \to \mathbb{R}^{n \times n}$ is the positive-definite mass-inertia matrix, $\mathbf{H} : \mathcal{X} \to \mathbb{R}^n$ contains the Coriolis and gravity terms, $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the selection matrix, and $\mathbf{u} \in \mathbb{R}^m$ is the control input. For the following discussion we assume that $\mathbf{B}$ has (column) rank $m < n$, i.e. (8.1) is underactuated.

As the robot experiences impulsive effects, it is subject to the instantaneous momentum transfer equation:

$$\mathbf{x}^+ = \mathbf{\Delta}(\mathbf{x}^-), \tag{8.2}$$

with $\mathbf{\Delta} : \mathcal{X} \to \mathcal{X}$ representing the impact map. Combining (8.1) and (8.2), the complete hybrid dynamics can be written as:

$$\mathscr{H} = \begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} & \mathbf{x} \notin \mathcal{S} \\ \mathbf{x}^+ = \mathbf{\Delta}(\mathbf{x}^-) & \mathbf{x}^- \in \mathcal{S} \end{cases}$$

where $\mathcal{S} \subset \mathcal{X}$ is an appropriately defined switching surface, for example the foot making or breaking contact with the ground [31].

Towards developing a stabilizing feedback controller for (8.1), define a collection of continuous time outputs $\mathbf{y} : \mathcal{X} \to \mathbb{R}^m$ that we would like to drive to zero. For outputs of relative degree two [25], consider the error coordinates $\mathbf{e} = (\mathbf{y}, \dot{\mathbf{y}}) \in \mathcal{E} \subseteq \mathbb{R}^{2m}$. These errors can be constructively stabilized via a RES-CLF, defined as:

Definition 19. *[2] For the system* (8.1)*, $V_\varepsilon : \mathcal{E} \to \mathbb{R}$ is said to be a rapidly exponentially stabilizing control Lyapunov function (RES-CLF) if there exists a $\lambda, k_1, k_2 > 0$, such that for all $\varepsilon \in (0, 1)$:*

$$\begin{aligned} k_1\|\mathbf{e}\|^2 \leq V_\varepsilon(\mathbf{e}) \leq k_2\|\mathbf{e}\|^2 \\ \inf_{\mathbf{u}} \dot{V}_\varepsilon(\mathbf{x}, \mathbf{u}) \leq -\frac{\lambda}{\varepsilon} V_\varepsilon(\mathbf{e}). \end{aligned} \tag{8.3}$$

Valid relative degree ensures the existence of a nonempty set $\mathcal{K}$, defined to be the set of all controllers satisfying the inequality (8.3). Any controller $\mathbf{k} \in \mathcal{K}$ renders the continuous time output exponentially stable, i.e. there exists $M, \tilde{\lambda} > 0$ such that:

$$\|\mathbf{e}(t)\| \leq Me^{-\frac{\tilde{\lambda}}{\varepsilon}t}\|\mathbf{e}(0)\|,$$

whereby tuning ε down enables arbitrarily fast convergence.

From Hybrid Dynamics to Discrete-Time Dynamics

We will be interested in modeling $\mathscr{H}$ as a discrete-time dynamical system via its impact-to-impact dynamics. To this end, let $\mathbf{x}_k \in \mathcal{X}$ denote the robot state just before impact, P denote an admissible parameter set for $\mathbf{v}_k \in P$, a discrete parameterization of the control input over a single continuous phase, and $t_k \in \mathbb{R}_{\geq 0}$ be the duration of the continuous phase. We reformulate our hybrid control system into discrete dynamics via:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{v}_k), \tag{8.4}$$

where $\mathbf{F} : \mathcal{X} \times P \to \mathcal{X}$ composes the impact map (8.2) with the flow of (8.1) under a parameterized feedback controller $\mathbf{u} = \mathbf{k}(\mathbf{x}(t), \mathbf{v}_k) \in \mathcal{K}$. In the context of hopping, we take $\mathbf{v}_k$ to be the desired impact angle. This parameterization of control input allows us to reason about the effect of impact conditions on the resulting system dynamics, which are the primary means of stabilizing legged systems. Note that here we assume the existence of a lower bound between impact times so that $\mathbf{F}$ is well defined. For a complete discussion of how to achieve this representation from the underlying hybrid dynamics, see [10]. Similar to the continuous-time case, the stability of the discrete time error dynamics can be reasoned about via Lyapunov theory:

Definition 20. *For the system* $\mathbf{e}_{k+1} = \mathbf{F}(\mathbf{e}_k)$, $V : \mathcal{E} \to \mathbb{R}$ *is a discrete exponential Lyapunov function if it is positive definite and there exists an* $\alpha \in (0, 1]$, $k_1, k_2 > 0$ *such that:*

$$k_1\|\mathbf{e}_k\|^2 \leq V(\mathbf{e}_k) \leq k_2\|\mathbf{e}_k\|^2$$
$$\Delta V(\mathbf{e}) = V(\mathbf{e}_{k+1}) - V(\mathbf{e}_k) \leq -\alpha V(\mathbf{e}_k).$$

The existence of such a Lyapunov function is necessary and sufficient for exponential stability of a system, i.e. the existence of $M > 0$, $\beta \in [0, 1)$ such that:

$$\|\mathbf{e}_k\| \leq M\beta^k\|\mathbf{e}_0\|.$$

Discrete-Time Optimal Control

We leverage optimal control to synthesize inputs $\mathbf{v}_k$ which stabilize the discrete time system in (8.2) while satisfying input constraints. To this end, consider the following infinite-time optimal control problem:

$$\begin{aligned} V(\mathbf{x}_0) \triangleq \min_{\mathbf{x}_k, \mathbf{v}_k} \quad & \sum_{k=0}^{\infty} c(\mathbf{x}_k, \mathbf{v}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{v}_k) \\ & \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) \leq 0 \end{aligned} \tag{8.5}$$

where $V : \mathcal{X} \to \mathbb{R}$ is termed the value function, $c : \mathcal{X} \times P \to \mathbb{R}$ is a positive-definite cost function and $\mathbf{h} : \mathcal{X} \times P \to \mathbb{R}^p$ contains any state-input constraints. With this, we can define the state-action value function $Q : \mathcal{X} \times P \to \mathbb{R}$ as:

$$Q(\mathbf{x}_k, \mathbf{v}_k) = c(\mathbf{x}_k, \mathbf{v}_k) + V(\mathbf{x}_{k+1}),$$

which defines the optimal control input at any state $\mathbf{x}_k$ through following optimization program:

$$\begin{aligned} \mathbf{v}_k^*(\mathbf{x}_k) = \arg\min_{\mathbf{v}_k} \quad & Q(\mathbf{x}_k, \mathbf{v}_k) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{v}_k, \mathbf{x}_k) \leq \mathbf{0} \end{aligned} \tag{8.6}$$

We rely on iteratively solving convex approximations of this nonconvex problem via iLQR. In Section 8.3 we show that tracking the output of optimal controllers in continuous time results in exponential stability of the discrete time dynamics.

Outputs and Zero Dynamics

Understanding the structure of underactuation provides key insight into constructing stabilizing controllers for these systems. To analyze the states that actuation directly impacts, consider the following coordinate change:

$$\boldsymbol{\eta} = \boldsymbol{\Phi}_{\boldsymbol{\eta}}(\mathbf{x}) \triangleq \begin{bmatrix} \mathbf{B}^\top \mathbf{q} \\ \mathbf{B}^\top \dot{\mathbf{q}} \end{bmatrix}, \quad \mathbf{z} = \boldsymbol{\Phi}_{\mathbf{z}}(\mathbf{x}) \triangleq \begin{bmatrix} \mathbf{N}\mathbf{q} \\ \mathbf{N}\mathbf{D}(\mathbf{q})\dot{\mathbf{q}} \end{bmatrix} \tag{8.7}$$

for $\boldsymbol{\eta} \in \mathcal{N} \subset \mathcal{X}$ and $\mathbf{z} \in \mathcal{Z} \subset \mathcal{X}$, where $\mathbf{N} \in \mathbb{R}^{(n-m)\times n}$ is chosen to be a basis for the left nullspace of $\mathbf{B}$. It is easily verified that the coordinate change $\boldsymbol{\Phi}(\mathbf{x}) \triangleq (\boldsymbol{\Phi}_{\boldsymbol{\eta}}(\mathbf{x}), \boldsymbol{\Phi}_{\mathbf{z}}(\mathbf{x}))$ is a diffeomorphism between $\mathcal{X}$ and $\mathcal{N} \times \mathcal{Z}$; therefore, $\boldsymbol{\Phi}^{-1}$ exists and any conclusions of stability of $(\boldsymbol{\eta}, \mathbf{z})$ are directly transferable back to $\mathbf{x}$.

In these coordinates, the hybrid dynamics are given by:

$$\dot{\boldsymbol{\eta}} = \hat{\mathbf{f}}(\boldsymbol{\eta}, \mathbf{z}) + \hat{\mathbf{g}}(\boldsymbol{\eta}, \mathbf{z})\mathbf{u}, \quad \dot{\mathbf{z}} = \boldsymbol{\omega}(\boldsymbol{\eta}, \mathbf{z}), \quad \boldsymbol{\Phi}^{-1}(\boldsymbol{\eta}, \mathbf{z}) \notin \mathcal{S}$$
$$\boldsymbol{\eta}^+ = \boldsymbol{\Delta}_{\boldsymbol{\eta}}(\boldsymbol{\eta}^-, \mathbf{z}^-), \quad \mathbf{z}^+ = \boldsymbol{\Delta}_{\mathbf{z}}(\boldsymbol{\eta}^-, \mathbf{z}^-), \quad \boldsymbol{\Phi}^{-1}(\boldsymbol{\eta}, \mathbf{z}) \in \mathcal{S}$$

termed the *actuated* dynamics and the *unactuated* dynamics, respectively. Note that these coordinates were exactly chosen such that $\hat{\mathbf{g}}(\boldsymbol{\eta}, \mathbf{z})$ is full rank and $\frac{d\mathbf{z}}{d\mathbf{x}}\mathbf{g}(\mathbf{x}) \equiv \mathbf{0}$; as such, this mapping decomposes the state space into coordinates which can directly be controlled, and those which cannot.

Assuming the continuous time input does not effect the impact map or impact time[1], applying $\boldsymbol{\Phi}$ to the discrete dynamics (8.4) results in:

$$\boldsymbol{\eta}_{k+1} = \hat{\mathbf{F}}(\boldsymbol{\eta}_k, \mathbf{z}_k, \mathbf{v}_k), \quad \mathbf{z}_{k+1} = \boldsymbol{\Omega}(\boldsymbol{\eta}_k, \mathbf{z}_k). \tag{8.8}$$

Now, consider a mapping $\boldsymbol{\psi}_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{N}$ and associated discrete-time error $\mathbf{e}_k = \boldsymbol{\eta}_k - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)$. The goal will be to design $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ such that driving $\mathbf{e}_k$ to zero results in stability of the overall system. This choice of error parameterization is inspired by other successful results in robotics; the Raibert Heuristic [21], reduced order models [12], and regulators for HZD gaits [23] all reason about where to place a robot's feet (the actuated state) as a function of their center of mass state (the underactuated state). We aim to generalize these methods and reason explicitly about constructive methods to generate provably stable behaviors. The construction of the mapping $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ induces an associated manifold $\mathcal{M}_{\boldsymbol{\psi}} \subset \mathcal{X}$ via:

$$\mathcal{M}_{\boldsymbol{\psi}} \triangleq \{(\boldsymbol{\eta}_k, \mathbf{z}_k) \mid \boldsymbol{\eta}_k = \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)\}. \tag{8.9}$$

We will be interested in enforcing conditions such that $\mathcal{M}_{\boldsymbol{\psi}}$ is controlled invariant, defined as:

Definition 21. *The manifold $\mathcal{M}_{\boldsymbol{\psi}}$ is controlled invariant if for all $(\boldsymbol{\eta}_k, \mathbf{z}_k) \in \mathcal{M}_{\boldsymbol{\psi}}$ there exists a $\mathbf{v}_k \in P$ such that the next state remains on the manifold, i.e.:*

$$\Big(\mathbf{F}(\boldsymbol{\eta}_k, \mathbf{z}_k, \mathbf{v}_k),\ \boldsymbol{\Omega}(\boldsymbol{\eta}_k, \mathbf{z}_k)\Big) \in \mathcal{M}_{\boldsymbol{\psi}}.$$

Assuming a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$, we now have the notion of discrete-time zero dynamics:

[1] This assumption is needed so that $\boldsymbol{\Omega}$ is not a function of $\mathbf{v}_k$ and is well justified on ARCHER as impact angle weakly effects impact time.

Definition 22. *The discrete-time zero dynamics associated with a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$ are given by:*

$$\mathbf{z}_{k+1} = \boldsymbol{\Omega}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k), \mathbf{z}_k).$$

These dynamics are autonomous but determined by choice of $\boldsymbol{\psi}_{\boldsymbol{\theta}}$; therefore, the goal of this work will be to design $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ such that the zero dynamics are stable. We show that stability on $\mathcal{M}_{\boldsymbol{\psi}}$ paired with a suitably defined output controller results in stability of the overall system.

8.3 Discrete-Time Zero Dynamics Policies

We propose a discrete-time mapping from the underactuated state, $\mathbf{z}_k$, to a desired actuated state, $\boldsymbol{\eta}_k$. This mapping, $\boldsymbol{\psi}_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{N}$, will encode the desired position of the actuated coordinates given the location of the unactuated coordinates at impact. The job of the continuous time controller is to drive $\boldsymbol{\eta}(t)$ to the desired preimpact location, $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_{k+1})$.

In this section, we will first reason about the ability of continuous time controllers to render $\mathcal{M}_{\boldsymbol{\psi}}$ attractive and invariant by driving the error $\mathbf{e}$ to zero. Second, we demonstrate that if the manifold has stable zero dynamics (trajectories on the manifold converge to the origin), then stabilizing the manifold stabilizes the entire system. Finally, we propose a learning pipeline which leverages optimal control to find a manifold with the desired properties.

Constructive Stabilization of the Zeroing Manifold

We show that the structure of the proposed manifold allows constructive stabilization techniques:

Lemma 7. *Consider a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$. There exists a continuous-time control law $\mathbf{k} \in \mathcal{K}$ which results in exponential stabilization of $\|\boldsymbol{\eta}_k - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)\|$.*

Proof. Consider a point $(\boldsymbol{\eta}_k, \mathbf{z}_k)$ and the evaluation of the current and next states on the manifold: $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)$ and $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_{k+1})$, respectively. As the $\boldsymbol{\eta}(t)$ dynamics are feedback linearizable, there exists a dynamically feasible trajectory $\boldsymbol{\eta}_d(t)$ such that $\boldsymbol{\eta}_d(0) = (\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k))^+$, and $\boldsymbol{\eta}_d(t_k) = \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_{k+1})$, where t_k is the impact time and $(\cdot)^+$ denotes a postimpact state. For example, $\boldsymbol{\eta}_d(t)$ can be constructed using Bezier polynomials [9]. Using a controller $\mathbf{k} \in \mathcal{K}$, i.e. satisfying the RES-CLF condition

(8.3), we can obtain exponential convergence to this trajectory in continuous time:

$$\|\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(t)\| \leq Me^{-\frac{\lambda}{\varepsilon}t}\|\boldsymbol{\eta}_k^+ - (\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k))^+\|,$$

for $M, \lambda > 0$. Taking $T_* > 0$ to be the lower bound between impact times, the impact states are uniformly bounded by:

$$\|\boldsymbol{\eta}_{k+1} - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_{k+1})\| \leq Me^{-\frac{\lambda}{\varepsilon}T_*}\|\boldsymbol{\eta}_k^+ - (\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k))^+\|.$$

Then, using the properties of the impact map we have:

$$\begin{aligned}\|\boldsymbol{\eta}_k^+ - (\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k))^+\| &= \|\boldsymbol{\Delta}_{\boldsymbol{\eta}}(\boldsymbol{\eta}_k, \mathbf{z}_k) - \boldsymbol{\Delta}_{\boldsymbol{\eta}}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k), \mathbf{z}_k)\| \\ &\leq L_\Delta\|\boldsymbol{\eta}_k - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)\|,\end{aligned}$$

substituting into the bound above, and choosing $\varepsilon > 0$ sufficiently small that $\alpha = ML_\Delta e^{-\frac{\lambda}{\varepsilon}T_*} \in (0, 1]$, we have:

$$\|\boldsymbol{\eta}_{k+1} - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_{k+1})\| \leq \alpha\|\boldsymbol{\eta}_k - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)\|,$$

proving exponential stability to the manifold, as desired. □ □

Remark 5. The desired trajectory $\boldsymbol{\eta}_d(t)$ is being implicitly replanned at impact via $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ as a function of the underactuated state $\mathbf{z}_k$. Additionally, the manifold $\mathcal{M}_{\boldsymbol{\psi}}$ is invariant under the discrete dynamics $\mathbf{F}$, but is notably not hybrid invariant.

Composite Stability

The previous section demonstrated a method for constructing a controller to exponentially stabilize the system to a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$. We now show that exponentially stabilizing the system to a manifold with stable zero dynamics results in composite exponential stability of the entire system:

Theorem 16. *Consider a controlled invariant manifold $\mathcal{M}_{\boldsymbol{\psi}}$ whose zero dynamics are exponentially stable. Any control law exponentially stabilizing $\|\boldsymbol{\eta}_k - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)\|$ stabilizes the discrete-time composite system $(\boldsymbol{\eta}_k, \mathbf{z}_k)$ to the origin.*

Proof. Define $\mathbf{e}_k = \boldsymbol{\eta}_k - \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k)$. By Lemma 1, there exists a continuous-time controller $\mathbf{k} \in \mathcal{K}$ rendering the discrete error dynamics exponentially stable. As such, converse Lyapunov theory guarantees the existence of a Lyapunov function $V_{\mathbf{e}} : \mathcal{E} \to \mathbb{R}$ satisfying:

$$\begin{aligned}k_1\|\mathbf{e}_k\|^2 \leq V_{\mathbf{e}}(\mathbf{e}_k) &\leq k_2\|\mathbf{e}_k\|^2 \\ \Delta V_{\mathbf{e}}(\mathbf{e}_k) &\leq -k_3\|\mathbf{e}_k\|^2\end{aligned}$$

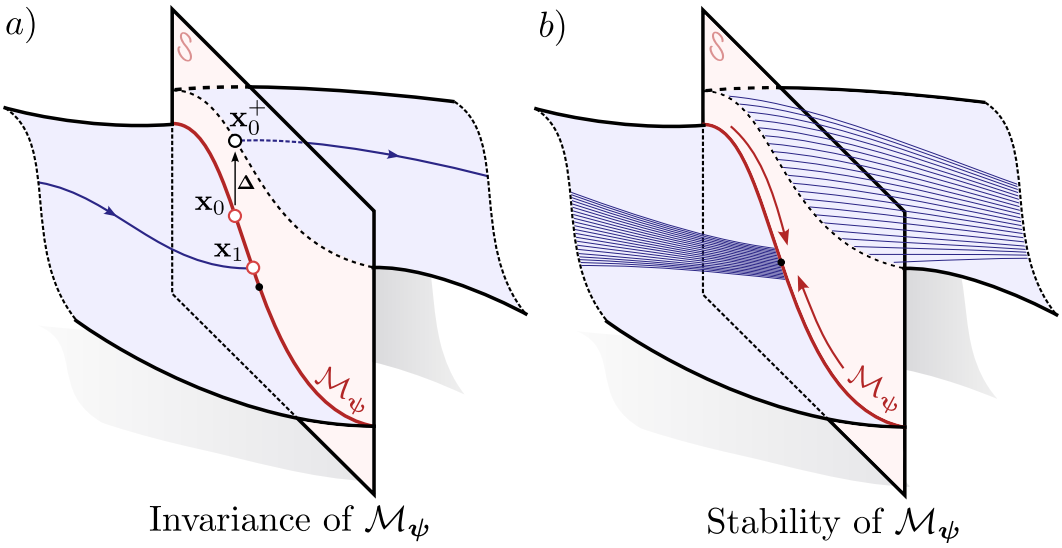


Figure 8.2: A depiction of the two necessary properties of $\mathcal{M}_{\psi}$: a) invariance under the discrete map $\mathbf{F}$, and b) stability.

Similarly, the stability of $\mathcal{M}_{\psi}$ implies the existence of a Lyapunov function $V_{\mathbf{z}} : \mathcal{Z} \to \mathbb{R}$ satisfying:

$$k_4\|\mathbf{z}_k\|^2 \leq V_{\mathbf{z}}(\mathbf{z}_k) \leq k_5\|\mathbf{z}_k\|^2$$
$$\Delta V_{\mathbf{z}}(\mathbf{z}_k) = V_{\mathbf{z}}(\boldsymbol{\Omega}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k), \mathbf{z}_k)) - V_{\mathbf{z}}(\mathbf{z}_k) \leq -k_6\|\mathbf{z}_k\|^2$$

The Lyapunov function $V_{\mathbf{z}}$ will additionally satisfy [2]:

$$|V_{\mathbf{z}}(\mathbf{z}) - V_{\mathbf{z}}(\mathbf{z}')| \leq k_7\|\mathbf{z} - \mathbf{z}'\| \left(\|\mathbf{z}\| - \|\mathbf{z}'\|\right) \triangleq \Gamma(\mathbf{z}, \mathbf{z}').$$

Consider the composite Lyapunov function candidate $V(\mathbf{e}_k, \mathbf{z}_k) \triangleq \sigma V_{\mathbf{e}}(\mathbf{e}_k) + V_{\mathbf{z}}(\mathbf{z}_k)$ with $\sigma > 0$, whereby:

$$\min\{\sigma k_1, k_4\}\|\mathbf{e}, \mathbf{z}\|^2 \leq V(\mathbf{e}, \mathbf{z}) \leq \max\{\sigma k_2, k_5\}\|\mathbf{e}, \mathbf{z}\|^2.$$

Furthermore, since $\mathbf{z}_k$ is exponentially stable on $\mathcal{M}_{\psi}$, discrete sequences on $\mathcal{M}_{\psi}$ will be exponentially decreasing:

$$\|\mathbf{z}_{k+1}\| = \|\boldsymbol{\Omega}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k), \mathbf{z}_k)\| \leq M\lambda\|\mathbf{z}_k\|,$$

for $\lambda \in [0, 1)$ and $M > 0$. Compute the difference of ΔV:

$$\begin{aligned}
\Delta V &= \sigma \Delta V_{\mathbf{e}}(\mathbf{e}_k) + V_{\mathbf{z}}(\boldsymbol{\Omega}(\boldsymbol{\eta}, \mathbf{z}_k)) - V_{\mathbf{z}}(\mathbf{z}_k) \\
&= \sigma \Delta V_{\mathbf{e}}(\mathbf{e}_k) + \Delta V_{\mathbf{z}}(\mathbf{z}_k) \\
&\qquad + V_{\mathbf{z}}(\boldsymbol{\Omega}(\boldsymbol{\eta}_k, \mathbf{z}_k)) - V_{\mathbf{z}}(\boldsymbol{\Omega}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k), \mathbf{z}_k)) \\
&\leq -\sigma k_1 \|\mathbf{e}_k\|^2 - k_6 \|\mathbf{z}_k\|^2 \\
&\qquad + \Gamma(\boldsymbol{\Omega}(\boldsymbol{\eta}_k, \mathbf{z}_k), \boldsymbol{\Omega}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}_k), \mathbf{z}_k)) \\
&= -\sigma k_1 \|\mathbf{e}_k\|^2 - k_6 \|\mathbf{z}_k\|^2 \\
&\qquad + k_7 L_{\boldsymbol{\Omega}}^2 \|\mathbf{e}_k\|^2 + 2M\lambda k_7 L_{\boldsymbol{\Omega}} \|\mathbf{e}_k\| \|\mathbf{z}_k\| \\
&= -\begin{bmatrix} \|\mathbf{e}_k\| \\ \|\mathbf{z}_k\| \end{bmatrix}^\top \begin{bmatrix} \frac{\sigma k_1}{2} - c(\sigma) & -M\lambda k_7 L_{\boldsymbol{\Omega}} \\ -M\lambda k_7 L_{\boldsymbol{\Omega}} & k_6 \end{bmatrix} \begin{bmatrix} \|\mathbf{e}_k\| \\ \|\mathbf{z}_k\| \end{bmatrix}
\end{aligned}$$

where $c(\sigma) = k_7 L_{\boldsymbol{\Omega}}^2 - \frac{\sigma}{2} k_1$, and $\Gamma(\boldsymbol{\Omega}(\boldsymbol{\eta}, \mathbf{z}), \boldsymbol{\Omega}(\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}), \mathbf{z}))$ is bounded using Lipschitz properties of the dynamics. Choosing $\sigma > \max\left\{\frac{2M^2\lambda^2 k_7^2 L_{\boldsymbol{\Omega}}^2}{k_1 k_6}, \frac{2k_7 L_{\boldsymbol{\Omega}}^2}{k_1}\right\}$ ensures the matrix is positive definite; therefore, V is a Lyapunov function certifying composite stability. □ □

Remark 6. Figure 8.2 depicts each of the assumptions used to prove stability in Theorem 16, namely discrete invariance and exponential stability of $\mathcal{M}_{\boldsymbol{\psi}}$. Subsequent sections will develop constructive techniques leveraging optimal control and learning for finding such manifolds.

Stability via Optimal Control

We will leverage optimality to enforce the stability on $\mathcal{M}_{\boldsymbol{\psi}}$. This choice is motivated by the fact that asymptotic stability is a necessary condition for an optimal controller to be well defined [18]. As Theorem 16 rests on assumptions of exponential stability, we define conditions under which optimality implies exponential stability:

Theorem 17. *Let $V(\mathbf{x}_k)$ be the value function for the optimal control problem defined in Equation* (8.5)*, where the cost function is quadratic, $c(\mathbf{x}_k, \mathbf{v}_k) = \mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k + \mathbf{v}_k^\top \mathbf{R}\mathbf{v}_k$, and the domain $\mathcal{X}$ is compact. If there exists an $\varepsilon > 0$ such that the LQR approximation of Equation* (8.5) *taken by linearizing the dynamics around the equilibrium point satisfies:*

$$\mathbf{v}_{LQR}(\mathbf{x}_k) = -\mathbf{K}\mathbf{x}_k \in \mathcal{H}(\mathbf{x}_k) \quad \forall \mathbf{x}_k \in B_\varepsilon(\mathbf{0}), \tag{8.10}$$

with $\mathcal{H}(\mathbf{x}_k) \triangleq \{\mathbf{v}_k \in P \mid \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) \leq 0\}$, then the nonlinear system is exponentially stable under the optimal controller.

Proof. We begin by showing the optimal controller Equation (8.5) is exponentially stabilizing in a neighborhood of the origin. Then, we extend this claim to the entire state space. In a sufficiently small ball around the origin, LQR (8.10) will be exponentially stabilizing for the nonlinear system [25], as it locally satisfies input bounds. This implies constants $M_{\text{LQR}}, \delta > 0$ and $\lambda_{\text{LQR}} \in [0, 1)$ such that:

$$\|\mathbf{x}_k\| \leq M_{\text{LQR}} \lambda_{\text{LQR}}^k \|\mathbf{x}_0\| \quad \forall \mathbf{x}_0 \in B_\delta(\mathbf{0}), \ \ \forall k \in \mathbb{Z}_+.$$

We first show that the optimal trajectory emanating from an initial condition $\mathbf{x}_0 \in B_\delta(\mathbf{0})$ is similarly exponentially stable. For any $M > 0, \ \lambda \in (0, 1)$, consider two cases:

Case 1: There exists a finite index set $\{k_i\}_{i=0}^N$ satisfying:

$$\|\mathbf{x}_{k_i}\| \geq M \lambda^{k_i} \|\mathbf{x}_0\|.$$

Compute the maximum violation ratio $R \geq 1$ given by:

$$R \triangleq \max_{i \in \{0,\ldots,N\}} \frac{\|\mathbf{x}_{k_i}\|}{M \lambda^{k_i} \|\mathbf{x}_0\|}.$$

If the index set is empty, take $R = 1$. Then

$$\|\mathbf{x}_k\| \leq R M \lambda^k \|\mathbf{x}_0\| \quad \forall k \in \mathbb{Z}_+$$

And the trajectory is exponentially stable.

Case 2: There exists a infinite index set $\{k_j\}_{j=0}^\infty$ satisfying:

$$\|\mathbf{x}_{k_j}\| \geq M \lambda^{k_j} \|\mathbf{x}_0\|. \tag{8.11}$$

We will establish that $V(\mathbf{x}_k)$ is an exponential Lyapunov function (Definition 20) along the trajectory, and thus the trajectory is exponentially stable. First, we bound the value function difference:

$$\begin{aligned} \Delta V(\mathbf{x}_k) = V(\mathbf{x}_k) - V(\mathbf{x}_{k-1}) &= -\mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k - \mathbf{v}_k^\top \mathbf{R} \mathbf{v}_k \\ &\leq -\lambda_{\min}(\mathbf{Q}) \|\mathbf{x}_k\|^2 \end{aligned} \tag{8.12}$$

Next, we need to show that $V(\mathbf{x}_k)$ is bounded by quadratics. Because the LQR controller is suboptimal for the nonlinear system, applying it increases the cost

relative to $V(\mathbf{x}_k)$:

$$\begin{aligned} V(\mathbf{x}_0) &\leq \sum_{k=0}^{\infty} \mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k + (\mathbf{K}\mathbf{x}_k)^\top \mathbf{R}(\mathbf{K}\mathbf{x}_k) \\ &\leq \sum_{k=0}^{\infty} \left(\bar{\lambda}(\mathbf{Q}) + \bar{\lambda}(\mathbf{K}^\top \mathbf{R}\mathbf{K})\right) \|\mathbf{x}_k\|^2 \\ &\leq \sum_{k=0}^{\infty} \left(\bar{\lambda}(\mathbf{Q}) + \bar{\lambda}(\mathbf{K}^\top \mathbf{R}\mathbf{K})\right) M_{\text{LQR}}^2 \lambda_{\text{LQR}}^{2k} \|\mathbf{x}_0\|^2 \\ &= \frac{M_{\text{LQR}}^2}{1 - \lambda_{\text{LQR}}^2} \left(\bar{\lambda}(\mathbf{Q}) + \bar{\lambda}(\mathbf{K}^\top \mathbf{R}\mathbf{K})\right) \|\mathbf{x}_0\|^2 \end{aligned}$$

where $\underline{\lambda}$ and $\overline{\lambda}$ are the minimum and maximum eigenvalue oeprators, respectively.

Finally, using (8.11), we can lower bound $V(\mathbf{x}_k)$ by:

$$\begin{aligned} V(\mathbf{x}_0) &= \sum_{j=0}^{\infty} \mathbf{x}_{k_j}^\top \mathbf{Q}\mathbf{x}_{k_j} + \mathbf{v}_{k_j}^\top \mathbf{R}\mathbf{v}_{k_j} \\ &\geq \sum_{j=0}^{\infty} \underline{\lambda}(\mathbf{Q}) \|\mathbf{x}_{k_j}\|^2 \\ &\geq \sum_{j=0}^{\infty} \underline{\lambda}(\mathbf{Q}) M^2 \lambda^{2k_j} \|\mathbf{x}_0\|^2 \\ &= \left[\frac{M^2}{1 - \lambda^2} \left(\bar{\lambda}(\mathbf{Q}) + \bar{\lambda}(\mathbf{K}^\top \mathbf{R}\mathbf{K})\right) - c\right] \|\mathbf{x}_k\|^2 \end{aligned}$$

Where c is the sum of the terms removed from the geometric series. Lastly, The above bounds hold for each point on the trajectory; therefore, V is a Lyapunov function certifying exponential stability of the trajectory.

Finally, we extend the claim outside of the ball around the origin. As $V \succ 0$ and $\Delta V \prec 0$, the optimal controller is asymptotically stable [18]. By compactness of $\mathcal{X}$ and (8.12), the time to enter $B_\delta(\mathbf{0})$ is bounded by:

$$K \triangleq \frac{\sup_{\mathbf{x}_0 \in \mathcal{X}} V(\mathbf{x}_0)}{\inf_{\mathbf{x}_0 \in \mathcal{X} \setminus B_\delta(\mathbf{0})} \Delta V(\mathbf{x}_0)} \leq \frac{\sup_{\mathbf{x}_0 \in \mathcal{X}} V(\mathbf{x}_0)}{\underline{\lambda}(\mathbf{Q})\delta^2}.$$

Because trajectories converge exponentially in $B_\delta(\mathbf{0})$,

$$\|\mathbf{x}_k\| \leq M\lambda^{k-K} \|\mathbf{x}_K\| \quad \forall \mathbf{x}_0 \in B_\delta(\mathbf{0}), \ \ \forall k \geq K$$

for $M > 0$, $\lambda \in [0, 1)$. By compactness of $\mathcal{X}$, trajectories are uniformly bounded $\|\mathbf{x}_k\| \leq B$; therefore:

$$\|\mathbf{x}_k\| \leq \frac{\max\{B, M\}\lambda^{-K}}{\min\{1, \delta\}} \lambda^k \|\mathbf{x}_0\| \quad \forall k \in \mathbb{Z}_+$$

is an exponential upper bound for the entire trajectory. □ □

Constructing the Zeroing Manifold via Learning

By Theorem 17, a manifold which is invariant under the optimal controller will be exponentially stable. Such a manifold then satisfies the assumptions of Theorem 16 and can be constructively stabilized resulting in composite stability of the entire system.

We will now present a learning method which leverages optimal control to ensure the assumptions of controlled invariance and stability of $\mathcal{M}_{\psi}$ as depicted in Figure 8.2 are met. Specifically, we will search for a manifold that is invariant under the optimal action, i.e. the controller that keeps sequences of states in the manifold coincides with the optimal controller for Equation (8.5).

To concisely define the loss function consider the variable

$$\zeta_{\theta}(\mathbf{z}) \triangleq \begin{bmatrix} \psi_{\theta}(\mathbf{z}) \\ \mathbf{z} \end{bmatrix} \tag{8.13}$$

which encodes a point on the manifold. The loss function is:

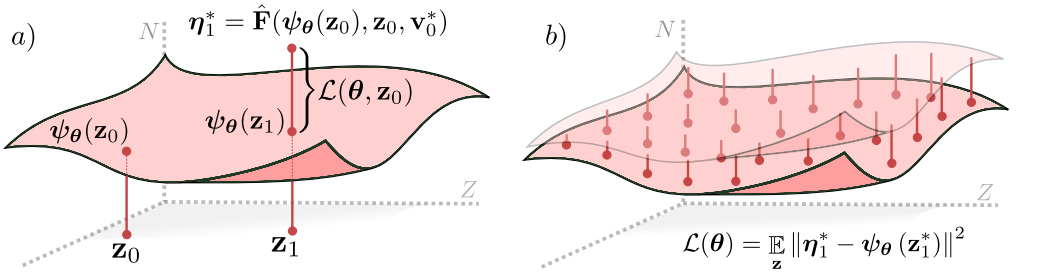
$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\mathbf{z}\sim\text{UNIFORM}} \left\| \eta_1^* \left(\zeta_{\theta}(\mathbf{z}) \right) - \psi_{\theta} \left(\mathbf{z}_1^* \left(\zeta_{\theta}(\mathbf{z}) \right) \right) \right\|_2^2, \tag{8.14}$$

where $\mathbf{z}_1^* = \mathbf{\Omega}(\psi(\mathbf{z}), \mathbf{z})$ and $\eta_1^* = \hat{\mathbf{F}}(\psi(\mathbf{z}), \mathbf{z}, \mathbf{v}^*)$, with $\mathbf{v}^*$ the optimal control input. The expectation is taken over a uniform distribution over $\mathcal{Z}$. The loss function directly measures how far an initial condition on the manifold deviates from the manifold under one discrete step of the optimal controller as depicted in Figure 8.3.

The learning pipeline outlined in Algorithm 8 starts an epoch by sampling a batch of points from $\mathcal{Z}$, therefore enabling a dimension reduction as compared to the complete state space. The network is then evaluated to produce a set of points on the current manifold, $\{\zeta_{\theta}(\mathbf{z}_i)\}_{i=1}^{N}$. We then approximately solve the optimal control problem Equation (8.5). Finally, we simulate the system forwards one step to obtain $\left(\eta_1^*, \mathbf{z}_1^*\right)$ which the loss computation in Equation (8.14) requires. If ψ_{θ} attains zero loss, because of continuity of the network and the loss function we can conclude that the resulting manifold $\mathcal{M}_{\psi}$ is invariant under the optimal control and can render the full order system stable by satisfaction of the preconditions for Theorem 16.

8.4 Application of ZDP to ARCHER

We deployed the ZDP method on the 3D hopping robot ARCHER. To discuss the application of ZDPs to ARHCER, consider the pose of the robot $\mathbf{q} = (\mathbf{p}, q) \in Q$



Loss at a point $\mathbf{z}_0$ Monte Carlo Sampling and Backpropagation

Figure 8.3: a) The loss function exactly measures the extent to which the manifold is not invariant under optimal action b) a Monte Carlo approximation of the spatial loss is used, wherein the optimal policy is backpropogated through to update the surface.

where $\mathbf{p} \in \mathbb{R}^3$ represents the global position in world frame and $q \in \mathbb{S}^3$ the robot's orientation quaternion. Taking the velocities to be $\mathbf{v} = (\dot{\mathbf{p}}, \boldsymbol{\omega}) \in T_{\mathbf{q}}\mathcal{Q}$ for $\dot{\mathbf{p}} \in \mathbb{R}^3$ the global linear velocity and $\boldsymbol{\omega} \in \mathfrak{s}^3$ the body frame angular rates, we can represent the full state as $\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in \mathcal{X} \triangleq T\mathcal{Q}$.

ARCHER evolves under hybrid dynamics. As such, its flight and ground phase dynamics are governed by (8.1) and it has two impact maps of the form (8.2) (one for the ground to flight transition, and another for flight to ground). We treat the vertical hopping as an autonomous system, and we will focus our attention on how to stabilize the position of the robot via orientation. The flight dynamics can be decomposed into actuated states, i.e. the orientation coordinates, and unactuated states, i.e. position coordinates:

$$\boldsymbol{\eta} = \begin{bmatrix} q \\ \boldsymbol{\omega} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \dot{\mathbf{p}} \end{bmatrix}.$$

Take $(\boldsymbol{\eta}_k, \mathbf{z}_k)$ to be a preimpact state. The ground phase does not depend on the control input, and the continuous-time evolution of the $\mathbf{z}$ coordinates has an extremely weak dependence on the discrete-time control input $\mathbf{v}_k$. We can assume $\boldsymbol{\Omega}$ is independent from $\mathbf{v}_k$ because the effect of different control inputs on impact time is negligible.

Online Control Implementation

Given a function $\psi_{\boldsymbol{\theta}}$, the controller aims to stabilize its associated zeroing manifold $\mathcal{M}_{\psi}$. Consider a state $(\boldsymbol{\eta}(t), \mathbf{z}(t))$ during the flight phase. We set the desired orientation to $\boldsymbol{\eta}_d(t) = \psi_{\boldsymbol{\theta}}(\mathbf{z}(t))$, and update this continuously throughout the flight phase. The desired set point is converted to a quaternion, q_d, which we stabilize

Algorithm 8 Monte Carlo Zero Dynamics Policy Training

hyperparameters: (Ξ, ρ, Υ)
Number of MC samples, Learning Rate and Number of Steps
Initialize $\boldsymbol{\theta}$ ▷ Pretrained with reasonable policy
for $i = 1 : \Upsilon$ **do**
  $\mathbf{z} \sim \text{UNIFORM}(\underline{\mathbf{z}}, \overline{\mathbf{z}})$
  $\zeta_{\boldsymbol{\theta}} \leftarrow \begin{bmatrix} \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{z}) \\ \mathbf{z} \end{bmatrix}$
  $\mathbf{x}_0 \leftarrow \boldsymbol{\Phi}^{-1}(\zeta_{\boldsymbol{\theta}})$
  $\mathbf{x}^*_{1:T}, \mathbf{v}^*_{1:T} \leftarrow \text{iLQR}(\mathbf{x}_0)$
  $\begin{bmatrix} \boldsymbol{\eta}^*_1(\zeta_{\boldsymbol{\theta}}(\mathbf{z})) \\ \mathbf{z}^*_1(\zeta_{\boldsymbol{\theta}}(\mathbf{z})) \end{bmatrix} \leftarrow \boldsymbol{\Phi}(\mathbf{x}_1)$
  $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i - \rho \nabla_{\boldsymbol{\theta}} \sum_{\mathbf{z}} \left\| \boldsymbol{\eta}^*_1(\zeta_{\boldsymbol{\theta}}(\mathbf{z})) - \boldsymbol{\psi}_{\boldsymbol{\theta}}\left(\mathbf{z}^*_1(\zeta_{\boldsymbol{\theta}}(\mathbf{z}))\right) \right\|_2^2$
end for
return $\boldsymbol{\theta}$

using the following quaternion PD controller in the flight phase:

$$\mathbf{u} = -\mathbf{K}_p \log(q_d^{-1} q) - \mathbf{K}_d \omega,$$

for suitable gains $\mathbf{K}_p, \mathbf{K}_d$. This controller is applied at 1kHz.

One key addition to the controller as compared to previous work [8] is the application of flywheel spindown in the ground phase. When the robot is in contact with the floor, the following control action is applied:

$$\mathbf{u} = -\gamma \dot{\boldsymbol{\vartheta}},$$

where $\dot{\boldsymbol{\vartheta}} \in \mathbb{R}^3$ represents the flywheel speed. This allows the system to maintain lower flywheel speeds and mitigates the problem of speed-torque constraints. This ground phase controller preserves the theoretical assumptions since the ground phase control is independent of output of the policy.

There are a few implementation differences from our theoretical implementation. The controller used in the proof of Lemma 7 differs from ours by (1) predicting the preimpact state $\mathbf{z}_{k+1}$, (2) tracking a trajectory $\boldsymbol{\eta}_d(t)$ defined by a bezier polynomial, and (3), using a RES-CLF. Empirically, a well tuned PD controller was sufficient to stabilize the continuous time system, and the feedforward input tracking that a trajectory would provide was not necessary.

ZDP Optimization and Learning Details

Notice that for discrete-time systems, Equation (8.5) is a nonlinear program even if the value function is available. To solve this optimal control problem, we employ

Figure 8.4: A snapshot of the experiments conducted with ARCHER, including set point tracking, disturbance rejection, and hopping over rough terrain.

Iterative LQR (iLQR), subject to box input constraints [29]. The iLQR problem is solved in the $\mathbf{x}$ variable, so the initial condition is obtain via $\mathbf{x} = \mathbf{\Phi}^{-1}(\boldsymbol{\eta}, \mathbf{z})$. We implemented Algorithm 8 in the JAX [5] and used a Network of 2 Layers with 256 hidden units each using ReLu activations. In our implementation of iLQR, we assume that the low-level controller has perfect tracking and exactly achieves the desired angle with zero angular velocity. This considerably simplifies the flight dynamics and therefore the trajectory optimization, allowing them to be solved for in closed form. The input bounds $\mathcal{H}(\mathbf{x}_k)$ were chosen such that the torque applied during flight is bounded by the difference between the post-impact state and the desired preimpact state. We require gradients of the optimal control, $\frac{d\mathbf{v}}{d\mathbf{x}}$, as presented in [3] – note that if no constraints are active, then this gradient is exactly

the feedback matrix $\mathbf{K} = \mathbf{Q}_{\mathbf{vv}}^{-1}\mathbf{Q}_{\mathbf{vx}}$ from the iLQR algorithm.

iLQR requires a stabilizing initial guess in order to converge; therefore, we use a Raibert heuristic for the first rollout. To eliminate this dependence, other optimal control methods could be used, for instance SQP. The authors experienced difficulty with the speed and accuracy of large-scale QP solvers in JAX and leveraged the fact that iLQR solves many small QPs for speed and stability. Additionally, for computational efficiency, we limit the number of iLQR iterations to five (empirically enough to obtain convergence for this system). The full code base for this project can be found at [6].

8.5 Results and Limitations

Hardware Results

A collection of the experiments conducted on ARCHER can be seen in Figure 8.4. The ARCHER hardware platform [1] consists of three KV115 T-Motors with 250 g flywheel masses attached for orientation control, and one U10-plus T-Motor attached to a 3-1 gear reduction to the foot via a cable and pulley system. The robot is powered by two 6 cell LiPo betteries connected in series, which can supply up to 50.8 V at over 100 A of current to the four ELMO Gold Solo Twitter motor controllers. The policy $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ was exported from JAX to an ONNX file, which is evaluated at 1kHz on an Ubuntu 20.04 machine with AMD Ryzen 5950x @ 3.4 GHz and 64 Gb RAM and torques are passed directly to the robot over ethernet. This controller does not require this amount of compute to run, and could be feasibly implemented on an NVIDIA Jetson or comparable board. A Kalman filter with projectile dynamics is used to filter the position estimates from optritrack in the flight phase. The manif library [11] is used to compute the log map for the quaternion PD controller.

We logged over 3,000 stable hops when deploying the ZDP method on the ARCHER hardware platform, a selection of which can be seen in Figure 8.4 and in the supplemental video [28]. Figure 8.5 depicts the desired impact angle, i.e. the learned policy evaluation, and the actual impact angle over the complete collection of all hardware tests. In general, as predicted by the theory, this manifold is both invariant under the feedback controller, and stable. Also interesting to note is that around the origin, the learned policy alignes with LQR, as presented in Theorem 17. Notably, away from the origin, the learned policy diverges from LQR in order to maintain stability under the enforced input contstraints. A comparison between the trained policy and the application of a naive LQR controller when trying to track

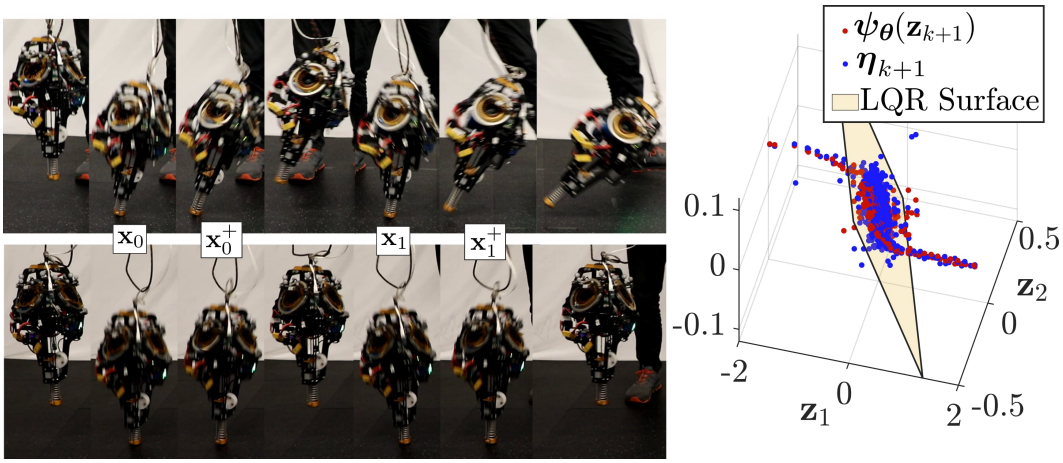


Figure 8.5: Left: A comparison between LQR (top) and ZDPs (bottom) while tracking a 2 m setpoint. Right: The output of the trained policy and the actual state at impact over 3000 hops, as compared to an LQR controller.

a setpoint 2 m away is seen in the left part of Figure 8.5, wherein ZDPs maintain stability by implicitly enforcing discrete invariance and optimality over a horizon.

The tight trajectory tracking and system behavior is seen in Figure 8.6, where ARCHER was asked to follow two laps of a 1 m square trajectory. As seen on the right of Figure 8.6, using a PD controller at the feedback level empirically resulted in the error (and therefore the torques) converging exponentially fast to a small neighborhood of zero during the flight phase. During this torque application, the flywheel speed can be seen to grow, while the ground phase controller is able to successfully regulate them close to zero.

Limitations

As training this policy involves querying the optimal control input and its gradients, each iteration of the training process is computationally expensive (2 seconds per iteration for a batch size of 30). The use of iLQR required a stabilizing controller to initialize the rollout and therefore can only do local improvements on a stabilizing policy. Furthermore, to avoid sampling initial conditions in the training pipeline which the hopper cannot stabilize, the policy $\boldsymbol{\psi}_{\boldsymbol{\theta}}$ was pretrained with a conservative Raibert heuristic.

8.6 Discussion

We have proposed a method of synthesizing stabilizing feedback controllers for hybrid underactuated systems. By exploiting the zero dynamics decomposition,

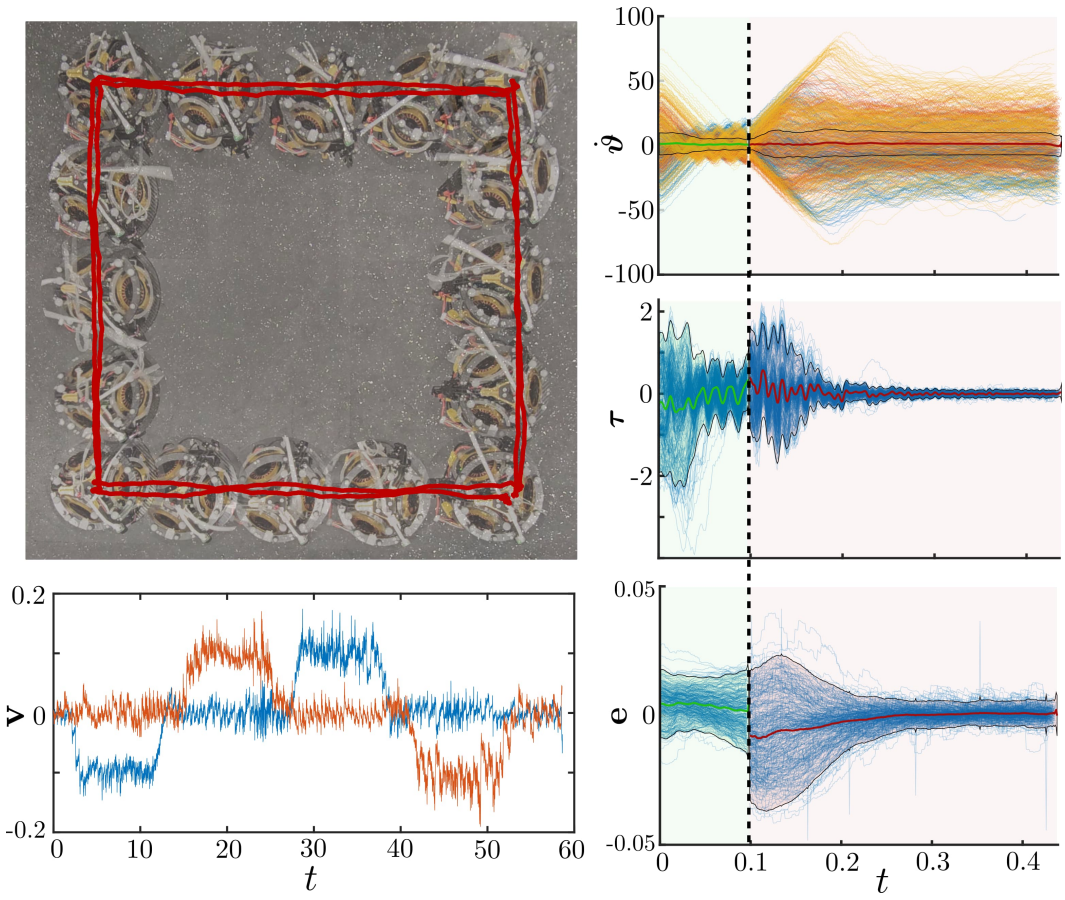


Figure 8.6: Square trajectory tracking. Left pane: overhead view with positional hardware data overlayed (top) and velocity tracking (bottom). Right pane: wheel velocities (top), torque (mid), and error (bottom) in the ground (green) and flight (red) phase with mean and 2σ deviation.

we demonstrated both theoretically and experimentally that stabilizing such systems can effectively be decomposed into designing a mapping which renders the discrete zeroing manifold invariant under optimal controllers and pairing it with a suitable tracking controller. The training procedure exemplifies the thesis strategy of converting trajectory-level properties into pointwise objectives: rather than optimizing over rollouts, we sample states and penalize deviation from manifold invariance, with Theorem 16 guaranteeing that convergence of this pointwise loss implies global stability. The hardware results on ARCHER, including over 3000 stable hops across varied terrain, demonstrate that this structure-aware approach achieves robust performance on a system where the zero dynamics decomposition provides the right inductive bias for learning.

References

[1] Eric Ryan Ambrose. "Creating ARCHER: A 3D Hopping Robot with Flywheels for Attitude Control". en. PhD thesis. California Institute of Technology, 2022. DOI: 10.7907/gbts-va63. (Visited on 09/16/2022).

[2] Aaron D Ames and Ioannis Poulakakis. "Hybrid zero dynamics control of legged robots". In: *Bioinspired Legged Locomotion: Models, Concepts, Control and Applications* (2017), pp. 292–331.

[3] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. "Differentiable mpc for end-to-end planning and control". In: *Advances in neural information processing systems* 31 (2018).

[4] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018.

[6] "Code". In: (2024). URL: %7Bhttps://github.com/ivandariojr/LearnedZeroDynamicsPolicies%7D.

[7] William Compton, Ivan Dario Jimenez Rodriguez, Noel Csomay-Shanklin, Yisong Yue, and Aaron D. Ames. "Constructive Nonlinear Control of Underactuated Systems via Zero Dynamics Policies". In: *preprint arXiv:2408.14749* (2024). arXiv: 2408.14749 [eess.SY].

[8] Noel Csomay-Shanklin, Victor D. Dorobantu, and Aaron D. Ames. "Nonlinear Model Predictive Control of a 3D Hopping Robot: Leveraging Lie Group Integrators for Dynamically Stable Behaviors". en. In: *2023 ICRA*. London, United Kingdom: IEEE, May 2023, pp. 12106–12112. ISBN: 9798350323658. DOI: 10.1109/ICRA48891.2023.10160873. (Visited on 03/15/2024).

[9] Noel Csomay-Shanklin, Andrew J Taylor, Ugo Rosolia, and Aaron D Ames. "Multi-rate planning and control of uncertain nonlinear systems: Model predictive control and control lyapunov functions". In: *2022 IEEE 61st CDC*. IEEE. 2022, pp. 3732–3739.

[10] Xingye Da and Jessy Grizzle. "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots". In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1063–1097. DOI: 10.1177/0278364919859425. eprint: https://doi.org/10.1177/0278364919859425.

[11] Jérémie Deray and Joan Solà. "Manif: A micro Lie theory library for state estimation in robotics applications". In: *Journal of Open Source Software* 5.46 (2020), p. 1371. DOI: 10.21105/joss.01371.

[12] Bin Han, Haoyuan Yi, Zhenyu Xu, Xin Yang, and Xin Luo. "3D-SLIP model based dynamic stability strategy for legged robots with impact disturbance rejection". In: *Scientific Reports* 12.1 (2022), p. 5892.

[13] Alberto Isidori. "Elementary Theory of Nonlinear Feedback for Single-Input Single-Output Systems". en. In: *Nonlinear Control Systems*. Ed. by Alberto Isidori. Communications and Control Engineering. London: Springer, 1995, pp. 137–217. ISBN: 978-1-84628-615-5. DOI: 10.1007/978-1-84628-615-5_4. (Visited on 01/30/2024).

[14] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 1. IEEE. 2001, pp. 239–246.

[15] Charles Khazoom, Seungwoo Hong, Matthew Chignoli, Elijah Stanger-Jones, and Sangbae Kim. "Tailoring Solution Accuracy for Fast Whole-body Model Predictive Control of Legged Robots". In: *preprint arXiv:2407.10789* (2024). arXiv: 2403.03995 [cs.RO].

[16] He Li and Patrick M. Wensing. "Cafe-Mpc: A Cascaded-Fidelity Model Predictive Control Framework with Tuning-Free Whole-Body Control". In: *preprint arXiv:2403.03995* (2024). arXiv: 2403.03995 [cs.RO].

[17] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. "Reinforcement Learning for Versatile, Dynamic, and Robust Bipedal Locomotion Control". In: *preprint arXiv:2401.16889* (2024). arXiv: 2401.16889 [cs.RO].

[18] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012. ISBN: 978-0-691-15187-8. DOI: 10.2307/j.ctvcm4g0s. (Visited on 01/31/2024).

[19] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (2000), pp. 789–814. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(99)00214-9.

[20] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science robotics* 7.62 (2022), eabk2822.

[21] Marc H. Raibert, H. Benjamin Brown, and Michael Chepponis. "Experiments in Balance with a 3D One-Legged Hopping Machine". en. In: *IJRR* 3.2 (June 1984). Publisher: SAGE Publications Ltd STM, pp. 75–92. ISSN: 0278-3649. DOI: 10.1177/027836498400300207. (Visited on 09/16/2022).

[22] Jenna Reher. *Dynamic Bipedal Locomotion: From Hybrid Zero Dynamics to Control Lyapunov Functions Via Experimentally Realizable Methods*. California Institute of Technology, 2021.

[23] Jenna Reher and Aaron D Ames. "Control lyapunov functions for compliant hybrid zero dynamic walking". In: *preprint arXiv:2107.04241* (2021).

[24] Ivan Dario Jimenez Rodriguez, Noel Csomay-Shanklin, Yisong Yue, and Aaron D. Ames. "Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies". In: *Proceedings of The 4th Annual L4DC*. Vol. 168. PMLR, June 2022, pp. 1060–1072.

[25] Shankar Sastry. "Linearization by State Feedback". en. In: *Nonlinear Systems: Analysis, Stability, and Control*. Ed. by Shankar Sastry. Interdisciplinary Applied Mathematics. New York, NY: Springer, 1999, pp. 384–448. ISBN: 978-1-4757-3108-8. DOI: 10.1007/978-1-4757-3108-8_9. (Visited on 10/15/2023).

[26] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *Procaeedings of ICLR*. 2016.

[27] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. "Do differentiable simulators give better policy gradients?" In: *ICML*. PMLR. 2022, pp. 20668–20696.

[28] "Supplemental Video". In: (). URL: %7Bhttps://vimeo.com/923800815%7D.

[29] Yuval Tassa, Nicolas Mansard, and Emo Todorov. "Control-limited differential dynamic programming". In: *2014 ICRA*. IEEE. 2014, pp. 1168–1175.

[30] Patrick M. Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. "Optimization-Based Control for Dynamic Legged Robots". In: *Trans. Rob.* 40 (Oct. 2023), pp. 43–63. ISSN: 1552-3098. DOI: 10.1109/TRO.2023.3324580.

[31] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. "Hybrid zero dynamics of planar biped walkers". In: *IEEE Transactions on Automatic Control* 48.1 (Jan. 2003), pp. 42–56. ISSN: 1558-2523. DOI: 10.1109/TAC.2002.806653.

[32] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. "Hybrid zero dynamics of planar biped walkers". In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 42–56.

Part V

Discussion and Conclusion

Chapter 9

DISCUSSION AND CONCLUSION

9.1 Technical Synthesis

This thesis developed three complementary strategies for integrating control-theoretic structure with learning. Each strategy addresses a different aspect of the learning-control interface, and together they provide a toolkit for constructive control of underactuated systems.

Training on Pointwise Conditions. The first strategy exploits the observation that trajectory-level properties reduce to pointwise algebraic conditions. Lyapunov stability requires $\dot{V}(x) < 0$ at each state; barrier safety requires $\dot{h}(x) \geq -\alpha(h(x))$ at each state; manifold invariance requires $(\eta_{k+1}, z_{k+1}) \in \mathcal{M}_\psi$ for each $(\eta_k, z_k) \in \mathcal{M}_\psi$. Monte Carlo sampling converts these conditions into loss functions that can be minimized without trajectory rollouts. LyaNet demonstrated this approach for stability, FI-ODE extended it to safety with interval-based certification, Neural Gaits applied barrier conditions on zero dynamics for bipedal walking, and the hopper work used manifold invariance losses for discrete-time hybrid systems.

Learning Inputs to Structured Controllers. The second strategy retains fixed control architectures and learns the quantities they require. Robust CBF controllers guarantee safety given bounds on state uncertainty; learning provides these bounds. The stereo vision chapter demonstrated this approach: multibaseline consistency yields self-supervised uncertainty estimates that serve as inputs to a measurement-robust CBF. Zero Dynamics Policies follow a similar pattern: optimal control provides stability guarantees; learning provides the manifold parameterization that makes these guarantees applicable globally.

Enforcing Structure Architecturally. The third strategy builds desired properties into the model architecture. KALIKO enforces globally linear latent dynamics by using the Kalman filter as an implicit encoder. The linearity holds regardless of training outcome because it is imposed by construction, not learned from data. This approach trades some expressiveness for guaranteed structural properties that simplify downstream control and analysis.

9.2 Guarantees and Achievements

The contributions of this thesis provide formal guarantees under stated assumptions, with hardware validation demonstrating practical applicability.

Stability Certificates. Theorem 10 establishes that for continuous-time systems, stabilizing outputs to a manifold with exponentially stable zero dynamics renders the full state exponentially stable. Theorem 16 extends this result to discrete-time hybrid dynamics. The local existence proof (Theorem 11) shows that stabilizing ZDPs exist for any locally controllable nonlinear system and can be constructed explicitly via linearization. These theorems assume exact models and perfect state knowledge; hardware results on AMBER-3M (walking) and ARCHER (3000+ hops) demonstrate that the approach remains effective under practical conditions where these assumptions are violated.

Safety Certificates. FI-ODE provides certified forward invariance for Neural ODE controllers using interval bound propagation. The stereo vision work provides empirical safety through learned uncertainty bounds, with the guarantee conditional on the learned distribution matching the true error distribution. Neural Gaits characterizes walking as barrier satisfaction on zero dynamics, with Theorem 15 relating zero dynamics safety to full-state safe sets.

Prediction and Interpretability. KALIKO achieves state-of-the-art prediction on wave dynamics while recovering interpretable Koopman eigenfunctions (limit cycles, energy invariants) that match theoretical expectations. The linear latent structure enables closed-loop MPC for crane stabilization with near-oracle performance.

9.3 Future Work: Control Structure as Transferable Representation

The introduction framed this thesis around a central question: what role should structure play in learning-based control? The bitter lesson suggests that general methods outperform hand-designed features given sufficient compute and data [4]. Yet the most successful sim-to-real transfer methods in legged locomotion rely on a specific structural choice: policies output joint position targets, and a PID controller tracks these targets [3, 2]. This observation points toward a promising direction for future research.

PID tracking succeeds for sim-to-real transfer because it provides a model-free interface between the learned policy and the physical actuators. The policy learns in simulation against an idealized PID response. On hardware, the same policy operates with different PID gains tuned to the physical system. The representation, i.e. joint position targets, transfers because PID tracking is robust to the model mismatch between simulation and reality. The policy need not learn actuator dynamics, friction, or delay; these concerns are absorbed by the low-level controller.

This observation suggests a design principle worth exploring: the choice of control structure determines what the policy must learn. A policy outputting torques must implicitly learn inverse dynamics, contact modeling, and actuator physics. A policy outputting joint targets delegates these concerns to PID and learns only the kinematic mapping from observations to desired configurations. The latter representation is more transferable because it is invariant to dynamics details that differ between simulation and hardware.

Current diffusion-based visuomotor policies [1] typically output end-effector trajectories or joint position sequences, relying on the same PID tracking paradigm. The learned diffusion model captures the distribution of successful trajectories; the low-level controller handles execution. This separation enables impressive manipulation results, but the guarantees are purely empirical. PID provides no stability certificate for the closed-loop system, and the diffusion model provides no guarantee that generated trajectories are dynamically feasible.

The structures developed in this thesis could offer an alternative worth investigating. Zero Dynamics Policies provide a representation where the policy outputs define a manifold $\eta = \psi(z)$, and stabilizing this manifold guarantees full-state stability. The policy learns the manifold shape; the low-level controller (feedback linearization, CLF-QP, or PD tracking) handles convergence to the manifold. This separation mirrors the PID paradigm but would add formal guarantees: if the zero dynamics are stable, the composite system is stable.

For visuomotor policies, this suggests a potential "diffusion on the manifold" approach. Rather than generating unconstrained trajectories, a diffusion model could generate parameters of Zero Dynamics Policies conditioned on visual observations. The generated manifold would be tracked by a stabilizing controller, ensuring that every sample from the diffusion model produces stable closed-loop behavior. Similarly, barrier functions could constrain the manifold to safe regions, providing safety guarantees for the generated policies.

The broader research direction is that control structure should be chosen not only for its guarantees but for the learning problem it induces. PID succeeds because it simplifies the learning problem to kinematic reasoning. ZDPs succeed because they reduce the learning problem to the unactuated subspace. Koopman methods succeed because they transform nonlinear prediction into linear algebra. Investigating how these structures can serve as transferable representations for modern learning pipelines, e.g. reinforcement learning, imitation learning, and diffusion models, constitutes a natural continuation of this work.

9.4 Conclusion

This thesis studied how control-theoretic structure can serve as an inductive bias for learning, producing controllers and models that combine the flexibility of neural networks with the guarantees of classical control. The three strategies developed, pointwise training, learning inputs to structured controllers, and architectural enforcement, address complementary aspects of this integration. Hardware validation on walking and hopping robots demonstrates that these methods achieve robust performance on challenging underactuated systems.

The paradigm shift from model-based to learning-based control is well underway. Reinforcement learning and diffusion policies achieve behaviors that would be difficult to specify analytically. Yet deployment in safety-critical settings requires answering: what guarantees does the learned controller provide? The contribution of this thesis is to show that such guarantees are achievable. Structure is not an obstacle to learning but a scaffold that makes learning tractable, transferable, and certifiable.

References

[1] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. “Diffusion Policy: Visuomotor Policy Learning via Action Diffusion”. In: *Robotics: Science and Systems (RSS)*. 2023.

[2] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. “Reinforcement learning for robust parameterized locomotion control of bipedal robots”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2811–2817.

[3] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science robotics* 7.62 (2022), eabk2822.

[4] Richard Sutton. “The bitter lesson”. In: *Incomplete Ideas (blog)* 13.1 (2019), p. 38.