10-2-2023

Stability of Deep Neural Networks for Feedback-Optimal Pinpoint Landings

Omkar S. Mulekar

Hancheol Cho

Riccardo Bevilacqua

74th International Astronautical Congress (IAC), Baku, Azerbaijan, 2-6 October 2023.

IAC–23–C1.3.9

Stability of Deep Neural Networks for Feedback-Optimal Pinpoint Landings

Omkar S. Mulekar[a], **Hancheol Cho** [b] **and Riccardo Bevilacqua** [c,*]

[a] *Department of Mechanical and Aerospace Engineering, University of Florida, United States of America,* omkar.mulekar@ufl.edu
[b] *Department of Aerospace Engineering, Embry-Riddle Aeronautical University, United States of America,* choh15@erau.edu
[c] *Department of Aerospace Engineering, Embry-Riddle Aeronautical University, United States of America,* bevilacr@erau.edu
[*] Corresponding Author

Abstract

The ability to certify systems driven by neural networks is crucial for future rollouts of machine learning technologies in aerospace applications. In this study, the neural networks are used to represent a fuel-optimal feedback controller for two different 3-degree-of-freedom pinpoint landing problems. It is shown that the standard sum-of-squares Lyapunov candidate is too restrictive to assess the stability of systems with fuel-optimal control profiles. Instead, a parametric Lyapunov candidate (i.e. a neural network) can be trained to sufficiently evaluate the closed-loop stability of fuel-optimal control profiles. Then, a stability-constrained imitation learning method is applied, which simultaneously trains a neural network policy and neural network Lyapunov function such that feedback-optimal control is achieved, and Lyapunov stability is verified. Phase-space plots of the Lyapunov derivative show the improvement in stability assessment provided by the neural network Lyapunov function, and Monte Carlo simulations demonstrate the stable, feedback-optimal control provided by the policy.
Keywords: Optimal Control, Imitation Learning, Lyapunov Stability

1. Introduction

The development of feedback-optimal controllers has been investigated for a variety of aerospace applications including earth-based, lunar, and planetary pinpoint landing systems, as well as orbital and deep-space trajectory maneuvers [1, 2]. Over recent decades, increased computing power and the use of Graphics Processing Units (GPUs) has prompted significant advancements in the use of Deep Neural Networks (DNNs) for tasks like image recognition, speech recognition, and autonomous robotic control [3–5].

Many recent efforts have leveraged deep learning tools to develop feedback controllers (sometimes called Guidance and Control Networks or G&CNets) for a variety of autonomous tasks, including fuel-optimal and energy-optimal feedback control for pinpoint-soft-landings [1, 6–9]. Despite the progress provided by studies that address and measure issues like distribution shift and loss of optimality during disturbances, the machine learning black-box problem must still be addressed, possibly by the enforcement and assessment of stability certificates (e.g. Lyapunov stability) in DNN policies trained via imitation learning to perform a specific feedback-control task [10–13].

One study assesses stability of a DNN controller applied to a 3DOF hovering problem by linearizing about the hover point then applying classical linear control assessments (e.g. root-locus plots) [9]. Some studies address the stability issue in the context of an LQR cost function applied to nonlinear dynamics by balancing a DNN policy in nonlinear regions of the state-space with a standard linear state-feedback law in linearizable regions of the state-space near the equilibrium point [14]. Another performs stability-constrained imitation learning, but for linear time-invariant (LTI) systems [15]. One study makes use of a Lyapunov falsifier to iteratively retrain a neural network policy and parametric Lyapunov function, and it maintains that stability is still acquired despite violations of the Lyapunov condition due to the "Almost Lyapunov" theorem [16, 17]. Another study enforces barrier function constraints to certify forward-invariance of specific regions of the state-space, and it also leverages the almost-Lyapunov condition [18]. Despite these achievements, the enforcement of stability certificates for imitation learning applied to the fuel-optimal pinpoint-landing problems with nonlinear dynamics still remains unaddressed.

In this investigation, it is first shown that the standard sum-of-squares (SOS) Lyapunov function is too restric-

tive to be a reasonable certificate on the control profiles needed for fuel-optimal landing maneuvers. This restriction is removed via the application of a neural network Lyapunov function (NNLF) [19]. Then, a constrained imitation learning approach is applied, allowing for various levels of stability certificate assessment on different subdomains of the lander state-space. These topics are investigated in the context of an earth-based, fuel-optimal 3-degree-of-freedom (3DOF) point-mass lander and a 3DOF rigid-body lander (equations of motion provided).

The paper is organized as follows. Background on the utilized techniques are provided in Section 2. Then, the problem formulation and proposed approach is presented in Section 3. Results, including Lyapunov stability assessment and Monte Carlo simulations, are included in Section 4. Finally, the paper is concluded in Section 5.

2. Background

The methods presented in this paper make use of a stability-constrained imitation learning method to achieve feedback-optimal control of a pinpoint landers.

2.1 Imitation Learning

Imitation learning is a sub-field of machine learning that trains a policy to replicate a desired behavior from expert demonstrations. Previous investigations have made use of imitation learning to replicate human behavior in the context of self-driving cars, robotic arm movements, and video games [5, 10, 20]. Non-human experts have been used when optimal control is the desired behavior. The core parts of the imitation learning methods used are similar, and success has been seen with these techniques in the aerospace domain for landing, hovering, and signal tracking problems [7, 8].

Demonstrations in this study are provided in the form of open loop trajectories. The trajectories together compose a set of state-action pairs $\mathcal{D} = \{\{(\mathbf{x}_{i,j}, \mathbf{u}^*_{i,j}\}_{i=1}^{N_i}\}_{j=1}^{N_j}$, where N_j is the number of trajectories, and N_i is the number of state-action pairs per trajectory, yielding a total of $N_i N_j$ state-action pairs that characterize the open loop state history $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ and optimal control history $\mathbf{u}^*(t) \in \mathbb{R}^{n_u}$. In addition to these state-action pairs, an environment that defines how states transition based on previous states and control inputs must be defined. For continuous-time systems, this environment is typically defined by ordinary differential equations (ODEs).

Deep imitation learning uses a DNN $\mathbf{u} = \pi_\theta(\mathbf{x})$ as the policy where $\theta \in \Theta$ are the parameters of the network $\pi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$, and Θ represents the parameter space of the policy. The optimal policy $\mathbf{u}^*$ can be represented as π_{θ^*} or π^* since the optimal parameters $\theta^* \notin \Theta$. The states in the set of demonstrations $\mathcal{D}$ can be described as being sampled from the distribution of states given the expert policy is used $\mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x}))$.

A loss function $l : \mathbb{R}^{n_u} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geq 0}$ that measures the error of policy predictions with respect to the expert prediction must be defined. A common loss function, and the one used in this study, is the squared error

$$l(\mathbf{u}_1, \mathbf{u}_2) = (\mathbf{u}_1 - \mathbf{u}_2)^T (\mathbf{u}_1 - \mathbf{u}_2)$$

The general imitation learning objective can then be formulated as

$$\arg\min_\theta \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\pi_\theta(\mathbf{x}))}[l(\pi^*(\mathbf{x}), \pi_\theta(\mathbf{x}))] \tag{1}$$

where $\mathbb{E}$ is the expected value given that states are sampled from *policy* behavior [21].

Behavioral cloning is the simplest imitation learning technique, which applies supervised learning on the provided example state-action pairs. Behavioral cloning does not rollout the policy during training, and it does not requery the expert for more training data, making it an attractive option when rollout or expert query is expensive. For parametric policy classes, it is formulated as

$$\arg\min_\theta \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x}))}[l(\pi^*(\mathbf{x}), \pi_\theta(\mathbf{x}))] \tag{2}$$

where $\mathbb{E}$ is the expected value given that states are sampled from expert behavior [21]. Behavioral cloning is simple to implement, however its reliance on the distribution of states provided by expert demonstrations can create distribution shift where $P_{test}(\mathbf{x}) \neq P_{train}(\mathbf{x})$ [22]. Some imitation learning techniques address this shift by either re-querying the expert during rollout and retraining for unfamiliar states like in Dataset Aggregation (DAgger) [10] or by injecting noise into the training data such as in DART [21]. Another study evaluated the distribution shift experienced from behavioral cloning applied to various formulations of the pinpoint landing problem [11].

Gradient based methods are common for parametric optimization, and they include simple methods like Stochastic Gradient Descent (SGD), and Adam [23, 24]. These algorithms make use of some update law γ to update the network parameters based on an evaluated gradient of the loss function $\partial l / \partial \theta$ and some update law hyper-parameters η_{learn} (e.g. learning rate, beta-decay, momentum, etc.).

$$\theta \leftarrow \gamma\left(\theta, \frac{\partial l}{\partial \theta}, \eta_{learn}\right) \tag{3}$$

The gradient evaluations and parameter updates are typically done empirically over batches of the data for computational effectiveness.

2.2 Optimal Control

Optimal control problems (OCPs) seek find control histories that minimize a defined cost functional subject to various differential and algebraic equality and inequality constraints that define the equations of motion, boundary conditions, and constraints on the system state and control spaces. Typically the ODE constraints are expressed in their first-order state-space form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{4}$$

where $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. The cost functional is defined as

$$J(\mathbf{x}(t), \mathbf{u}(t), t_f) = \Phi(\mathbf{x}(t_f), \mathbf{u}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t) dt \tag{5}$$

where L is a path cost and Φ is a terminal cost. The Single-Phase OCP is then formulated as

$$\begin{aligned} &\min_{\mathbf{u}(t), t_f} \; J(\mathbf{x}(t), \mathbf{u}(t), t_f) \\ &\text{subject to} \\ &\quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ &\quad \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{0} \\ &\quad \mathbf{e}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = \mathbf{0} \end{aligned} \tag{6}$$

where $\mathbf{f}$ defines the system equations of motion, $\mathbf{g}$ defines algebraic equality and inequality constraints, and $\mathbf{e}$ defines endpoint constraints [25].

Generally, OCPs do not have analytical solutions, and numerical methods must be employed. Direct methods transcript the OCP to a standard nonlinear programming problem (NLP). Then, an NLP solver is called to obtain the optimal control $\mathbf{u}^*(t)$ and the optimal trajectory of states $\mathbf{x}^*(t)$. We use the MATLAB package OpenOCL, which performs the transcription of the OCP to an NLP, and then calls IPOPT (Interior Point OPTimizer) to solve it [26–28].

2.3 Lyapunov Stability

The core topic that this study uses to enforce and evaluate safety of the closed-loop system is Lyapunov Stability. A system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}(\mathbf{x}))$ with equilibrium point $\bar{\mathbf{x}} = \mathbf{0}$ such that $\mathbf{f}(\mathbf{0}, \mathbf{u}(\mathbf{0})) = \mathbf{0}$ is stable in the sense of Lyapunov if for any $\varepsilon > 0$ there exists a $\delta > 0$ such that for any initial $\|\mathbf{x}_0\| < \delta \Rightarrow \|\mathbf{x}(t)\| < \varepsilon$ for all $t > 0$. Additionally, if $\lim_{t\to\infty} \|\mathbf{x}(t)\| = 0$, then the system is said to by asymptotically stable [29].

This notion of Lyapunov stability can be verified by defining a positive function $V : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$ such that $V(\mathbf{x}) > 0 \; \forall \mathbf{x} \neq \mathbf{0}$ and $V(\mathbf{0}) = 0$, and then showing that its derivative $\dot{V} = dV/dt$ is negative (i.e. $\dot{V}(\mathbf{x}) \leq 0$ for all $\mathbf{x} \neq \mathbf{0}$ for stability and $\dot{V}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{0}$ for asymptotic stability).

The standard Lyapunov candidate is the SOS, which has quadratic form

$$V_{SOS}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{x} \tag{7}$$

This candidate is foundational to nonlinear control theory and has yielded many successes in adaptive nonlinear control [29, 30]. However, its ellipsoidal level-sets have been shown to be too restrictive to fully describe stable regions of state-space for some systems.

The NNLF presented in [19] addresses this restriction by using a parameterized Lyapunov candidate (i.e. a DNN) V_ϕ where $\phi \in \Phi$ are the network parameters and Φ is the network parameter space. The NNLF is defined as the inner-product of a neural network v_ϕ with itself

$$V_\phi(\mathbf{x}) = v_\phi^T(\mathbf{x}) v_\phi(\mathbf{x}) \tag{8}$$

NNLFs have certain restrictions on the structure of the network, the activation functions, and the parameters space in order to maintain them as valid Lyapunov candidates. For example, each activation function must have a trivial nullspace (e.g. tanh), and the weights of each layer l must maintain the trivial nullspace by having the form

$$W_l = \begin{bmatrix} G_{l1}^T G_{l1} + \varepsilon I_{n_{l-1}} \\ G_{l2} \end{bmatrix} \tag{9}$$

where n_l is the number of neurons in layer l, $G_{l1} \in \mathbb{R}^{q_l \times n_{l-1}}$ for some $q_l \in \mathbb{N}_{\geq 0}$, $G_{l2} \in \mathbb{R}^{(n_l - n_{l-1}) \times n_{l-1}}$, $I_{n_{l-1}} \in \mathbb{R}^{n_{l-1} \times n_{l-1}}$ is the identity matrix and $\varepsilon > 0$ is a constant to keep the top partition positive-definite. The parameter is then defined as $\phi = \{G_{l1}, G_{l2}\}_{l=1}^{N_l}$, where N_l is the number of layers in v_ϕ. More requirements on the NNLF parameters can be found in [19].

The time derivative (sometimes called a Lie derivative) can be calculated via the chain-rule

$$\dot{V}_\phi = \frac{dV_\phi}{dt} = \left(\frac{\partial V_\phi}{\partial \mathbf{x}}\right)^T \frac{d\mathbf{x}}{dt} = \left(\frac{\partial V_\phi}{\partial \mathbf{x}}\right)^T \dot{\mathbf{x}} \tag{10}$$

Using the equations of motion, the final derivative is

$$\dot{V}_\phi(\mathbf{x}, \mathbf{u}) = \left(\frac{\partial V_\phi}{\partial \mathbf{x}}\bigg|_{\mathbf{x}}\right)^T \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{11}$$

The gradient $\partial V_\phi / \partial \mathbf{x}$ can be evaluated at any $\mathbf{x}$ using back-propagation/automatic differentiation. In this study, TensorFlow is used to handle all neural network gradient calculations and parameter updates [31].

2.4 Certificate Levels

In this study, the time-derivative of the NNLF $\dot{V}_\phi(\mathbf{x}, \pi_\theta(\mathbf{x}))$is used as the safety/stability certificate for the trained policy. The certificate is evaluated across specific distributions in the state-space using the trained policy π_θ.

According to Brunke, certification $\mathbf{c}(\mathbf{x},\mathbf{u}) \leq \mathbf{0}$ can be applied in three levels [32]:

- Level III: Constraint satisfaction guaranteed where $\mathbf{c}(\mathbf{x},\mathbf{u}) \leq \mathbf{0}$.
- Level II: Constraint satisfaction guaranteed with high probability where $P(\mathbf{c}(\mathbf{x},\mathbf{u}) \leq \mathbf{0}) \geq p_{satisfy}$ for some $0 < p_{satisfy} < 1$.
- Level I: Constraint satisfaction encouraged where $\mathbf{c}(\mathbf{x},\mathbf{u}) \leq \varepsilon_{satisfy}$ for some $\varepsilon_{satisfy} > 0$.

It should be noted that if $p_{satisfy} = 1$ or $\varepsilon_{satisfy} = 0$, the Level II and Level I certificates, respectively, become Level III. This level framework is applied to the systems developed in this paper, evaluating $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0))$.

2.5 Applications to Pinpoint Landing

Imitation learning has been applied in several studies to perform landing maneuvers in a variety of lander formulations and scenarios. One study generated tens of millions of state-action pairs for optimal landing of a low-fidelity multicopter model using direct methods, and then trained a neural network to perform the landings [7]. An extention of the study used the same formulation, but generated the trajectories using indirect methods to reduce chatter in the optimal control profiles [33]. Another study combined the optimal states with a lunar surface image generator to train the feedback controller to map surface images to optimal control, rather than the state directly [8]. Reinforcement learning approaches have been applied to higher fidelity formulations (i.e. 6DOF) of the fuel-optimal pinpoint landing problem by using the Advantage-Actor-Critic algorithm [2].

3. Approach

In this investigation, many optimal trajectories are generated for pinpoint landings of a 3DOF point-mass lander and a 3DOF rigid-body lander. The OCP formulations (including equations of motion) for the two problems are provided, as well as the ranges of initial conditions used for the trajectory generation campaigns. A feedback controller policy as well as an NNLF are pretrained on the optimal open-loop state-action pairs, and then a constrained optimization that simultaneously retrains the policy and NNLF on a broader range of states is performed.

As a note on terminology, the lander equations of motion do not represent fuel consumption and/or a changing mass of the vehicle. It is common to model the lander mass as one of the states of the vehicle, and to define an ODE that describes how the lander mass decreases with applied control effort. In this stability investigation, the lander mass is considered to be constant. Despite there being no fuel-consumption, the term "fuel-optimal" that is maintained in this paper refers to the square-root in the OCP cost functional integrands, and it serves to distinguish the OCPs solved in this study from those with an "energy-optimal" integrand which does not have a square-root.

3.1 Optimal Data Generation

The 3DOF point-mass lander is of constant mass with number of states $n_x = 6$ and number of controls $n_u = 3$. The state vector is $\mathbf{x} = [\mathbf{r}^T, \mathbf{v}^T]^T$ where $\mathbf{r} = [x, y, z]^T$ and $\mathbf{v} = [v_x, v_y, v_z]^T$ are the position vector and velocity vector, respectively, expressed in the inertial frame with z pointing up. The control vector is $\mathbf{u} = [u_1, u_2, u_3]^T$. Each control represents a thrust on the point-mass lander applied in the directions of the inertial frame axes. The OCP is then formulated as

$$\begin{aligned} &\min_{\mathbf{u}(t), t_f} \int_{t_0}^{t_f} \sqrt{u_1^2 + u_2^2 + u_3^2}\, dt \\ &\text{subject to} \\ &\quad \dot{\mathbf{r}} = \mathbf{v} \\ &\quad \dot{\mathbf{v}} = \frac{1}{m}\mathbf{u} - \mathbf{g} \end{aligned} \tag{12}$$

where the gravity vector $\mathbf{g} = [0, 0, g]^T$ is earth's gravity (i.e. $g = 9.81$ m/s^2). The end point constraints contain the initial conditions $\mathbf{x}(t_0) = \mathbf{x}_0$ and the final conditions $\mathbf{x}(t_f) = \mathbf{x}_f$. The constraints on lander control are given as

$$\begin{aligned} -20 \text{ N} \leq u_i \leq 20 \text{ N for } i \in \{1, 2\} \\ 0 \leq u_3 \leq 20 \text{ N} \end{aligned}$$

The 3DOF rigid-body lander is of constant mass with number of states $n_x = 6$ and number of controls $n_u = 2$. The state vector is $\mathbf{x} = [\mathbf{r}^T, \phi, \mathbf{v}^T, \omega]^T$ where $\mathbf{r} = [x, y]^T$ is the position vector with y pointing up, ϕ is the tilt angle, $\mathbf{v} = [v_x, v_y]^T$ is the velocity vector in the inertial frame, and ω is the angular velocity. The control u_1 represents an axial thrust pointing up the body axis of the lander, and the control u_2 represents a lateral thrust applied at some

moment arm r from the lander center of mass.

$$\begin{aligned}
&\min_{\mathbf{u}(t),t_f} \int_{t_0}^{t_f} \sqrt{u_1^2+u_2^2}dt \\
&\text{subject to} \\
&\quad \dot{\mathbf{r}} = \mathbf{v} \\
&\quad \dot{\phi} = \omega \\
&\quad \dot{v}_x = \frac{1}{m}(u_2\cos(\phi) - u_1\sin(\phi)) \\
&\quad \dot{v}_y = \frac{1}{m}(u_2\sin(\phi) + u_1\cos(\phi)) - g \\
&\quad \dot{\omega} = \frac{r}{J}u_2
\end{aligned} \tag{13}$$

where the gravity g is earth's gravity (i.e. $g = 9.81$ m/s^2) and J is the lander moment of inertia. The end point constraints contain the initial conditions $\mathbf{x}(t_0) = \mathbf{x}_0$ and the final conditions $\mathbf{x}(t_f) = \mathbf{x}_f$. A single state constraint of

$$-50 \text{ deg} \leq \phi \leq 50 \text{ deg}$$

is enforced to represent vehicle "overtilting" requirements. The constraints on lander control are given as

$$\begin{aligned}
0 \leq &u_1 \leq 20 \text{ N} \\
-20 \text{ N} \leq &u_2 \leq 20 \text{ N}
\end{aligned}$$

A diagram of the 3DOF rigid-body lander is shown in Fig. 1. The point-mass formulation serves as a set of simple dynamics on which to test the developed algorithms. The rigid-body formulation serves as an incremental extention to a more complicated system that has coupled dynamics and fewer control inputs. Future studies can extend the techniques presented to 6DOF formulations of the landing problem, such as the ones presented in [2, 11, 13].

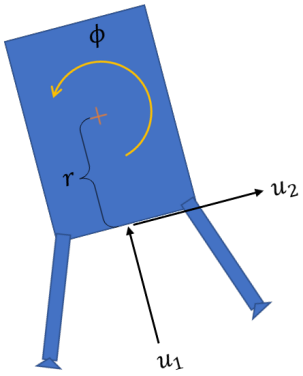

Fig. 1: Diagram of 3DOF rigid-body lander

For both formulations, a total of 90,000 trajectories were generated, each containing 100 optimal time-steps yielding a total of 9 million optimal state-action pairs $\mathcal{D}$. Of the 9 million pairs, 4 million were reserved as a testing set $\mathcal{D}_{test}$ (i.e. not used in any training). Of the remaining 5 million pairs, 1 million were used as a validation set $\mathcal{D}_{val}$ (i.e. evaluated during training, but not used to update parameters), yielding a training dataset $\mathcal{D}_{train}$ of size of 4 million pairs for both formulations

3.2 Pretraining

Before implementing the full constrained optimization, some pretraining of both the policy π_θ and the NNLF V_ϕ was done to prepare the neural networks.

The pretraining of the policy consisted of the standard behavioral cloning problem formulated by Eqn. (2), using the training dataset $\mathcal{D}_{train}$ to represent the distribution of states using optimal/expert behavior $p(\mathbf{x}|\pi^*(\mathbf{x}))$. The result is a policy that approximates π^* in open loop and demonstrates near-optimal feedback control when simulated.

Pretraining the NNLF V_ϕ is done in a two-step process. The first step fits V_ϕ to verify stability of the expert π^*, since this is control profile that our policy π_θ ultimately approximates. The second step expands, in an unsupervised fashion, the region of the state-space for which V_ϕ verifies policy behavior.

An alternate way of representing the Lyapunov stability condition $\dot{V}(\mathbf{x}) \leq 0$ is by using the max function evaluated on a given distribution of states. Verifying the condition

$$\max_{\mathbf{x}\in D}[\dot{V}(\mathbf{x})] < 0 \tag{14}$$

also verifies that $\dot{V}(\mathbf{x}) < 0$ for all $\mathbf{x}$ in some subdomain $D \subseteq \mathbb{R}^{n_x}$.

The first step of the NNLF pretraining is then formulated as

$$\arg\min_{\phi} \left[\max_{\mathbf{x}\sim p(\mathbf{x}|\pi^*(\mathbf{x}))} [\dot{V}_\phi(\mathbf{x},\pi^*(\mathbf{x}))] \right] \tag{15}$$

In accordance with Eqn. (3), this formulation requires calculation of the gradient

$$\frac{\partial}{\partial\phi}\left(\max_{\mathbf{x}\sim p(\mathbf{x}|\pi^*(\mathbf{x}))} [\dot{V}_\phi(\mathbf{x},\pi^*(\mathbf{x}))] \right) \tag{16}$$

for updates of the NNLF parameter ϕ. The dataset $\mathcal{D}_{train}$ represents the states samples from distribution $p(\mathbf{x}|\pi^*(\mathbf{x}))$. The result of this first step is verification that $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0)|\mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x})), \mathbf{u} = \pi^*(\mathbf{x})) > p_{satisfy}$.

The second step of the NNLF pretraining uses the pretrained policy π_θ to evaluate and train V_ϕ on a broader

range of states without re-querying the expert π^*, which would require running a full OCP optimization routine (using OpenOCL) for each queried state. The broader range of states is sampled from a uniform distribution defined by upper and lower values of each state in the state-space to create dataset $\mathcal{X}_{MC}$. The second step of the NNLF pretraining is then formulated as

$$\arg\min_{\phi} \left[\max_{\mathbf{x}\in\mathcal{X}_{MC}} [\dot{V}_\phi(\mathbf{x}, \pi_\theta(\mathbf{x}))] \right] \tag{17}$$

The result of this second step is verification that $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \in \mathcal{X}_{MC}, \mathbf{u} = \pi_\theta(\mathbf{x})) > p_{satisfy}$.

3.3 Constrained Optimization

Ultimately, the objective of stability-constrained imitation learning is to achieve verifiably stable feedback-optimal controller that performs pinpoint landings. The objective is not to achieve stability alone, or to simply perform stable landings, but to do both of those things with a fuel-optimal control profile in a feedback loop. The behavioral cloning problem is therefore altered to reflect this as a constrained optimization problem

$$\begin{aligned} &\arg\min_{\substack{\theta\in\Theta \\ \phi\in\Phi}} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\pi^*(\mathbf{x}))}[l(\pi^*(\mathbf{x}), \pi_\theta(\mathbf{x}))] \\ &\text{subject to} \\ &\quad \max_{\mathbf{x}\in\mathcal{X}_{MC}} [\dot{V}_\phi(\mathbf{x}, \pi_\theta(\mathbf{x}))] \leq 0 \end{aligned} \tag{18}$$

From this formulation, a Lagrangian can be defined as

$$\begin{aligned} \mathcal{L}(\theta,\phi;\lambda) =& \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\pi^*(\mathbf{x}))}[l(\pi^*(\mathbf{x}), \pi_\theta(\mathbf{x}))] \\ &+ \lambda \max_{\mathbf{x}\in\mathcal{X}_{MC}} [\dot{V}_\phi(\mathbf{x}, \pi_\theta(\mathbf{x}))] \end{aligned} \tag{19}$$

where $\lambda \geq 0$ is the Lagrange multiplier. The solution to (18) is then given by the minimax problem

$$\min_{\substack{\theta\in\Theta \\ \phi\in\Phi}} \max_{\lambda\geq 0} \mathcal{L}(\theta,\phi;\lambda) \tag{20}$$

Gradient-based optimization methods require calculation of the gradients $\partial\mathcal{L}/\partial\theta$ and $\partial\mathcal{L}/\partial\phi$ which are calculated via back-propagation, and

$$\frac{\partial\mathcal{L}}{\partial\lambda} = \max_{\mathbf{x}\in\mathcal{X}_{MC}} [\dot{V}_\phi(\mathbf{x}, \pi_\theta(\mathbf{x}))] \tag{21}$$

which is evaluated directly and is effectively a measure of the constraint violation. In implementation the minimax problem is unstable when the initial parameters θ and ϕ violate the constraint since the maximizing inner loop would not converge, resulting in updates to λ that increase $\mathcal{L}$ without bound. The maximin problem is solved instead to address this

$$\max_{\lambda\geq 0} \min_{\substack{\theta\in\Theta \\ \phi\in\Phi}} \mathcal{L}(\theta,\phi;\lambda) \tag{22}$$

The optimization routine has an inner loop that simultaneously trains π_θ and V_ϕ based on the gradient of $\mathcal{L}$ over datasets $\mathcal{D}_{train}$ and $\mathcal{X}_{MC}$. Then, an outer optimization maximizes $\mathcal{L}$ over λ by incrementing it based on constraint violation/satisfaction.

3.4 Networks and Training

The same neural network structure is used for the policy and NNLF in both formulations of the landing problem.

The policy π_θ contains two fully-connected hidden layers, each with 100 neurons using the tanh activation function. The output layer has n_u number of neurons with a linear activation function. The Adam optimizer was used to pretrain the policy, as well as update the policy parameters during the inner loop of the constrained optimization (i.e. Lagrangian minimization).

The NNLF V_ϕ is defined from an internal neural network v_ϕ, as shown in Eqn. (8). This network contained two hidden layers with 100 and 200 neurons from first to last. The output layer has 300 neurons, and each layer used the tanh activation function. The Adam optimizer was used to pretrain the NNLF for the point-mass formulation, and SGD was used to pretrain the NNLF for the rigid-body formulation. Finally, the Adam optimizer was used to update the policy parameters during the inner loop of the constrained optimization (i.e. Lagrangian minimization) for both formulations.

4. Results

4.1 SOS Lyapunov

First, the SOS Lyapunov candidate is evaluated on the dataset $\mathcal{D}_{test}$ to show that it is too restrictive to be a useful certificate, given the desired control profiles of this investigation. This evaluation provides the value of $P(\dot{V}_{SOS}(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x})), \mathbf{u} = \pi^*(\mathbf{x}))$.

To visualize, the positive and negative $\dot{V}_{SOS}(\mathbf{x})$ locations are plotted on phase-spaces. The z phase-space of the point-mass problem can be seen in Fig. 2, and the ϕ phase-space of the rigid-body problem can be seen in Fig. 3.

From these evaluations of V_{SOS},

- $P(\dot{V}_{SOS}(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x})), \mathbf{u} = \pi^*(\mathbf{x})) = 74.8\%$ for the point-mass problem, and

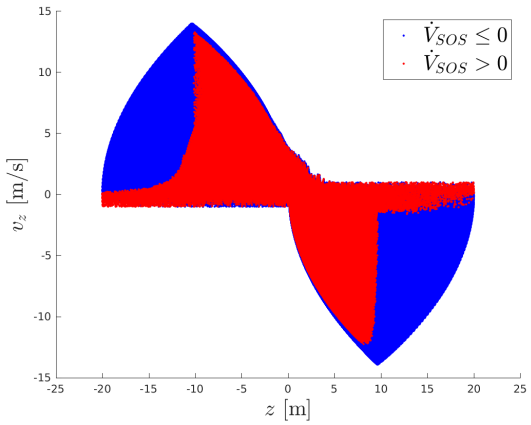

Fig. 2: z phase-space of V_{SOS} evaluated on optimal data

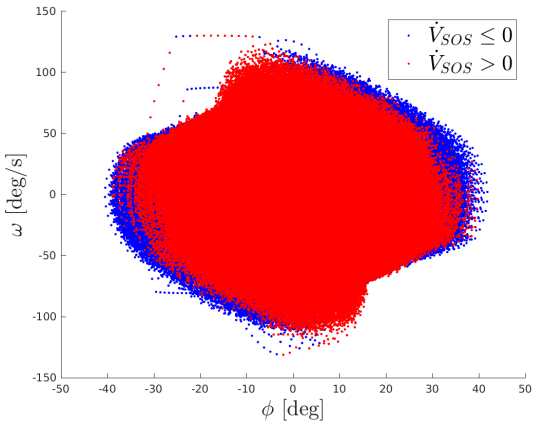

Fig. 3: ϕ phase-space of V_{SOS} evaluated on optimal data

- $P(\dot{V}_{SOS}(\mathbf{x},\mathbf{u}) \leq 0)|\mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x})), \mathbf{u} = \pi^*(\mathbf{x})) =$ 72.7% for the rigid-body problem

which further prompt the need for parametric lyapunov candidates (i.e. the NNLF).

4.2 Pretrained Networks

After applying the first step of the NNLF pretraining process, V_ϕ was evaluated on the dataset $\mathcal{D}_{test}$ to show the capability of the NNLF to more accurately characterize the expert behavior as stable. This evaluation provides the value of $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0)|\mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x})), \mathbf{u} = \pi^*(\mathbf{x}))$.

Again, to visualize, the positive and negative $\dot{V}_\phi(\mathbf{x})$ locations are plotted on phase-spaces. The z phase-space of the point-mass problem can be seen in Fig. 4, and the ϕ phase-space of the rigid-body problem can be seen in Fig. 5.

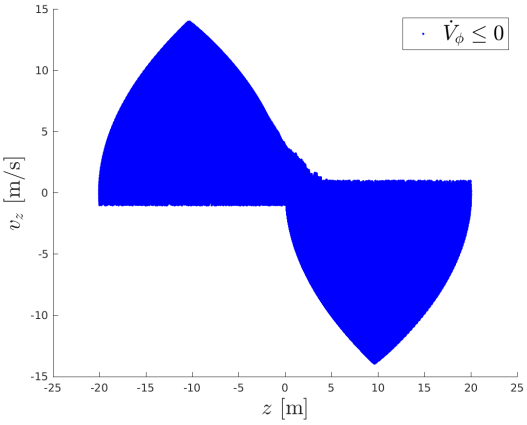

Fig. 4: z phase-space of V_ϕ evaluated on optimal data

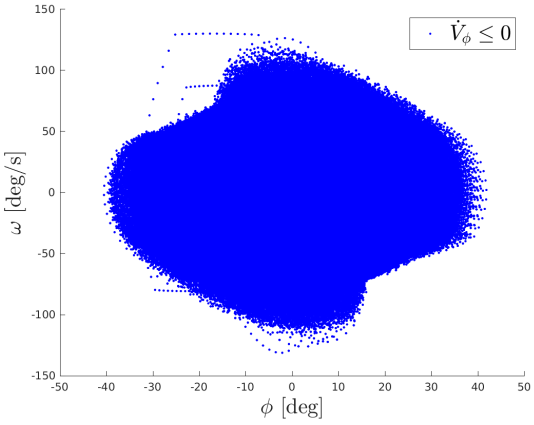

Fig. 5: ϕ phase-space of V_ϕ evaluated on optimal data

From these evaluations of V_ϕ,

- $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \sim p(\mathbf{x}|\pi^*(\mathbf{x})), \mathbf{u} = \pi^*(\mathbf{x})) = 100\%$ for both lander formulations.

which shows that the NNLF is able to certify the expert policy as stable, which is not the case for the SOS Lyapunov candidate.

This initial step is crucial to the certification problem since the optimal state action pairs are what the policy ultimately imitates. Any test of stability that fails to certify the expert would not work well as a constraint on the imitation learning problem.

Behavioral cloning was used to pretrain the policy π_θ on dataset $\mathcal{D}_{train}$. This step alone results in a policy that very closely approximates the optimal control profiles. An open-loop evaluation of $\pi_\theta(\mathbf{x})$ on states in $\mathcal{D}_{test}$ was performed, and the open-loop test plot for 300 evaluations can be seen in Fig. 6 for the point-mass formulation.

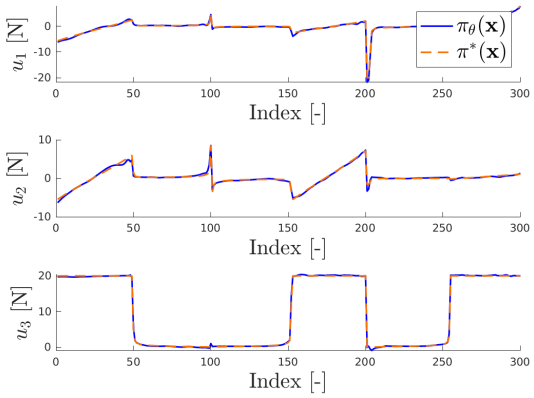

Fig. 6: Open loop test of the pretrained π_θ for point-mass formulation

4.3 Constrained Optimization

After pretraining both the policy and the NNLF, the full constrained optimization was performed. The policy and NNLF were trained simultaneously to achieve a policy that closely matches the fuel-optimal trajectories in $\mathcal{D}$, while also satisfying the Lyapunov stability requirement across a broader range of states $\mathcal{X}_{MC}$.

The trained NNLF V_ϕ was evaluated on a new, randomly sampled dataset $\mathcal{X}_{MC,test}$ to show the capability of the NNLF to more accurately characterize the trained policy behavior as stable. This evaluation provides the value of $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \in \mathcal{X}_{MC,test}, \mathbf{u} = \pi_\theta(\mathbf{x}))$.

Again, to visualize, the positive and negative $\dot{V}_\phi(\mathbf{x})$ locations are plotted on phase-spaces. The z phase-space of the point-mass problem can be seen in Fig. 7, and the ϕ phase-space of the rigid-body problem can be seen in Fig. 8.

From these evaluations of V_ϕ,

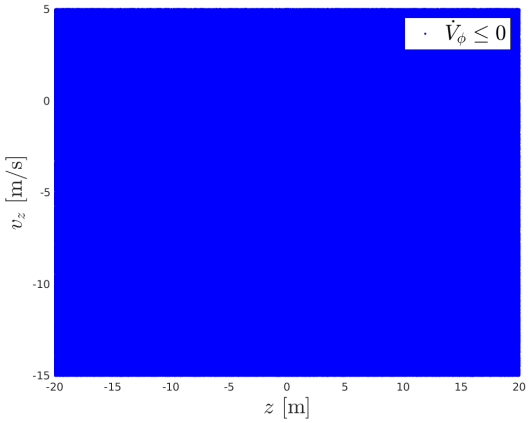

Fig. 7: z phase-space of V_ϕ evaluated on broader range of states

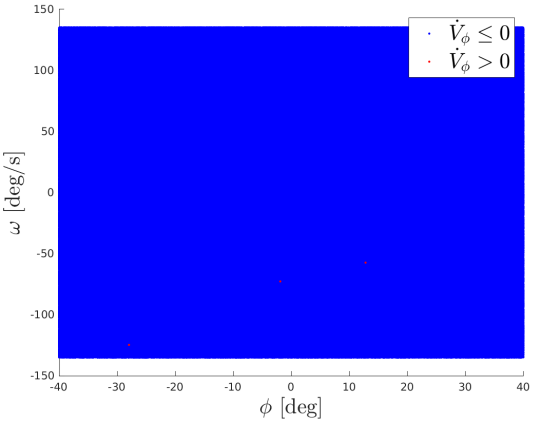

Fig. 8: ϕ phase-space of V_ϕ evaluated on broader range of states

- $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \in \mathcal{X}_{MC,test}, \mathbf{u} = \pi_\theta(\mathbf{x})) = 100\%$ for the point-mass problem, and
- $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \in \mathcal{X}_{MC,test}, \mathbf{u} = \pi_\theta(\mathbf{x})) = 99.9999\%$ for the rigid-body problem

When evaluated over a larger region of the state-space characterized by $\mathcal{X}_{MC}$, the trained policy and NNLF yield a Level III certification of Lyapunov stability for the point-mass, and a Level II certification of stability for the rigid-body.

4.4 Simulation Results

To more fully demonstrate the stability of the trained system, a Monte Carlo simulation of 1,000 trajectories from random initial states was performed. The position and velocity histories for the point-mass Monte Carlo simulation are shown in Figs. 9 and 10.

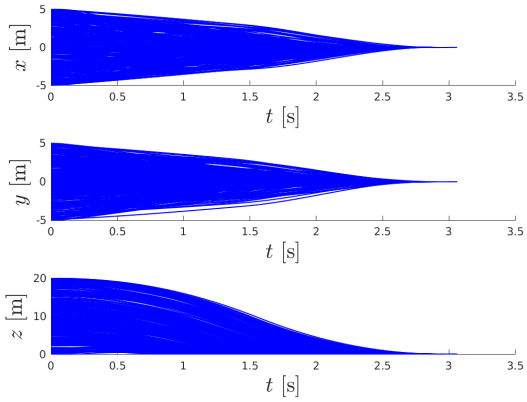

Fig. 9: z phase-space of V_ϕ evaluated on optimal data

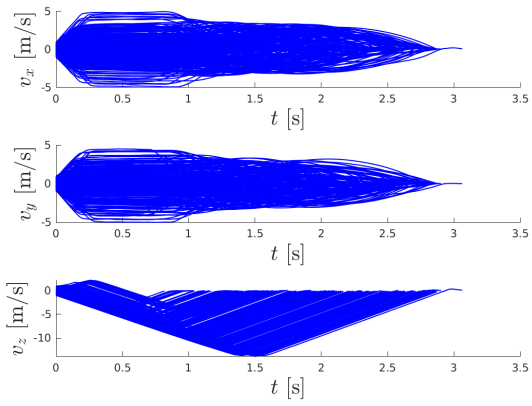

Fig. 10: ϕ phase-space of V_ϕ evaluated on optimal data

Each of the simulated trajectories demonstrated a successful near-optimal pinpoint landing at the landing site. To further evaluate the assessment capability of the trained NNLF, V_ϕ was evaluated on each of the states in the Monte Carlo simulation. This evaluation provides the value of $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \sim p(\mathbf{x}|\pi_\theta(\mathbf{x})), \mathbf{u} = \pi_\theta(\mathbf{x}))$. To visualize, the positive and negative $\dot{V}_\phi(\mathbf{x})$ locations are plotted on phase-spaces. The z phase-space of the point-mass problem can be seen in Fig. 11, and the ϕ phase-space of the rigid-body problem can be seen in Fig. 12.

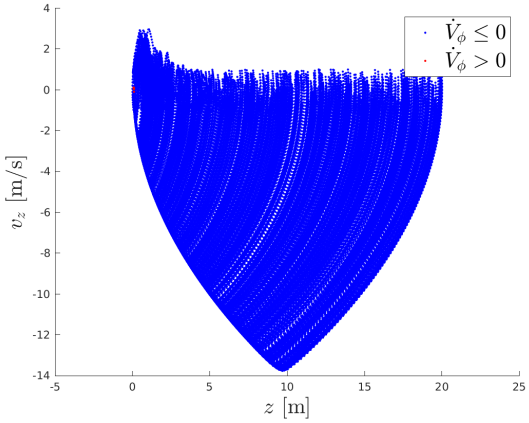

Fig. 11: z phase-space of V_ϕ evaluated on simulated data

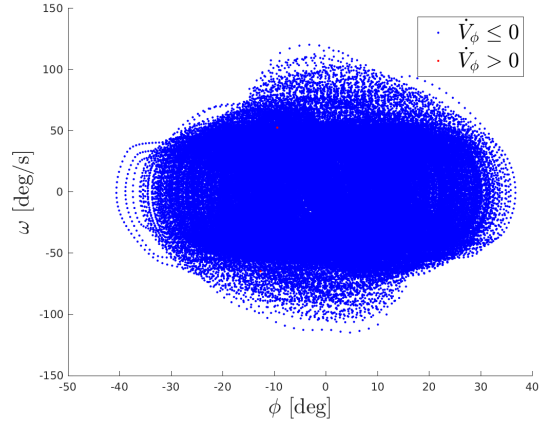

Fig. 12: ϕ phase-space of V_ϕ evaluated on simulated data

From these evaluations of V_ϕ,

- $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \sim p(\mathbf{x}|\pi_\theta(\mathbf{x})), \mathbf{u} = \pi_\theta(\mathbf{x})) = 99.9935\%$ for the point-mass problem, and
- $P(\dot{V}_\phi(\mathbf{x},\mathbf{u}) \leq 0) | \mathbf{x} \sim p(\mathbf{x}|\pi_\theta(\mathbf{x})), \mathbf{u} = \pi_\theta(\mathbf{x})) = 99.9999\%$ for the rigid-body problem

When evaluated over the simulated data, the trained policy and NNLF yield a Level II certification of Lyapunov stability for both landing problem formulations.

5. Conclusion

The study presented shows a full constrained optimization technique that trains the parameters of a neural network policy and a neural network Lyapunov function to perform stable, fuel-optimal feedback control of a pinpoint landing system. It was shown how the imitation learning problem can be constrained by the time-derivative of the NNLF, which allows for certifiable control profiles not possible with the standard sum of squares Lyapunov candidate.

This paper showed how open-loop optimal trajectories can be used to train deep neural networks to perform optimal feedback control of two 3DOF pinpoint landing formulations. By simultaneously updating the parameters of the policy and the NNLF, Level II and Level III certifications of the system stability were achieved.

Future work may involve extending the Lyapunov-based training to systems with model-parameter uncertainties, prompting the training of adaptive controllers with adaptive update laws. Similar techniques that make use of sampling states from a broader range of the state-space to be used in an unsupervised fashion in training can be investigated.

Ultimately, this investigation has advanced the stability enforcement and assessment capabilities for neural network systems used for optimal control. Since closed loop controllers cannot be analytically derived, the use of imitation learning to replicate numerically generated open-loop trajectories was combined with notions of closed-loop stability assessment. These topics can be extended to other aerospace applications from robotic control to orbit transfers and deep-space trajectory maneuvers.

References

[1] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astorynamics*, vol. 3, no. 4, pp. 287–299, 2019. DOI: 10.1007/s42064-018-0053-6. eprint: https://doi.org/10.1007/s42064-018-0053-6. [Online]. Available: https://doi.org/10.1007/s42064-018-0053-6.

[2] B. Gaudet, R. Linares, and R. Furfaro, "Deep reinforcement learning for six degree-of-freedom planetary landing," *Advances in Space Research*, vol. 65, Jan. 2020. DOI: 10.1016/j.asr.2019.12.030.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[4] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Mach. Learn. J*, vol. 84, pp. 171–203, Jan. 2008. DOI: 10.1007/s10994-010-5223-6.

[6] A. Zavoli and L. Federici, "Reinforcement learning for robust trajectory design of interplanetary missions," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 8, pp. 1440–1453, 2021. DOI: 10.2514/1.G005794. eprint: https://doi.org/10.2514/1.G005794. [Online]. Available: https://doi.org/10.2514/1.G005794.

[7] C. Sánchez-Sánchez, D. Izzo, and D. Hennes, "Learning the optimal state-feedback using deep networks," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2016.

[8] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Topputo, R. Linares, *et al.*, "Deep learning for autonomous lunar landing," in *2018 AAS/AIAA Astrodynamics Specialist Conference*, Univelt, vol. 167, 2018, pp. 3285–3306.

[9] D. Izzo, D. Tailor, and T. Vasileiou, "On the stability analysis of deep neural network representations of an optimal state feedback," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 1, pp. 145–154, 2021. DOI: 10.1109/TAES.2020.3010670.

[10] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[11] O. S. Mulekar, R. Bevilacqua, and H. Cho, "Metric to evaluate distribution shift from behavioral cloning for fuel-optimal landing policies," *Acta Astronautica*, vol. 203, pp. 421–428, 2023, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2022.12.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/%5Clinebreak%20S009457652200683X.

[12] O. Mulekar, H. Cho, and R. Bevilacqua, "Six-degree-of-freedom optimal feedback control of pinpoint landing using deep neural networks," in *AIAA SCITECH 2023 Forum*. DOI: 10.2514/6.2023-0689. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2023-0689. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2023-0689.

[13] O. S. Mulekar, H. Cho, and R. Bevilacqua, "Neural network based feedback optimal control for pinpoint landers under disturbances," *Acta Astronautica*, vol. 211, pp. 353–367, 2023, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2023.06.033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576523003326.

[14] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, "Neural network optimal feedback control with guaranteed local stability," *IEEE Open Journal of Control Systems*, vol. 1, pp. 210–222, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248496024.

[15] H. Yin, P. Seiler, M. Jin, and M. Arcak, "Imitation learning with stability and safety guarantees," *IEEE Control Systems Letters*, vol. 6, pp. 409–414, 2022. DOI: 10.1109/LCSYS.2021.3077861.

[16] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/2647c1dba23bc0e0f9cdf75339e120d2-Paper.pdf.

[17] S. Liu, D. Liberzon, and V. Zharnitsky, "Almost lyapunov functions for nonlinear systems," *Automatica*, vol. 113, p. 108 758, 2020, ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2019.108758. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109819306211.

[18] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 1724–1735. [Online]. Available: https://proceedings.mlr.press/v164/dawson22a.html.

[19] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:51906510.

[20] M. Bojarski *et al.*, "End to end learning for self-driving cars," Apr. 2016.

[21] M. Laskey, J. N. Lee, R. Fox, A. D. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *CoRL*, 2017.

[22] J. G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, pp. 521–530, 2012.

[23] C. M. Bishop, "Probability distributions," in *Pattern Recognition and Machine Learning*, Springer, 2006, pp. 67–136.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[25] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd. USA: Cambridge University Press, 2009, ISBN: 0898716888.

[26] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[27] J. Koenemann, G. Licitra, M. Alp, and M. Diehl, "Openocl - open optimal control library," Jun. 2019.

[28] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[29] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, The book can be consulted by contacting: PH-AID: Wallet, Lionel. [Online]. Available: https://cds.cern.ch/record/1173048.

[30] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. USA: Prentice-Hall, Inc., 1989, ISBN: 0138399948.

[31] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[32] L. Brunke *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022. DOI: 10.1146/annurev-control-042920-020211. eprint: https://doi.org/10.1146/annurev-control-042920-020211. [Online]. Available: https://doi.org/10.1146/annurev-control-042920-020211.

[33] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: Study on landing problems," *ArXiv*, vol. abs/1610.08668, 2016.