Meta-Learning-Based Adaptive Stability Certificates for Dynamical Systems

Amit Jena, Dileep Kalathil, Le Xie

Department of Electrical and Computer Engineering, Texas A&M University, USA
amit_jena@tamu.edu, dileep.kalathil@tamu.edu, le.xie@tamu.edu

Abstract

This paper addresses the problem of Neural Network (NN) based adaptive stability certification in a dynamical system. The state-of-the-art methods, such as Neural Lyapunov Functions (NLFs), use NN-based formulations to assess the stability of a non-linear dynamical system and compute a Region of Attraction (ROA) in the state space. However, under parametric uncertainty, if the values of system parameters vary over time, the NLF methods fail to adapt to such changes and may lead to conservative stability assessment performance. We circumvent this issue by integrating Model Agnostic Meta-learning (MAML) with NLFs and propose meta-NLFs. In this process, we train a meta-function that adapts to any parametric shifts and updates into an NLF for the system with new test-time parameter values. We demonstrate the stability assessment performance of meta-NLFs on some standard benchmark autonomous dynamical systems.

Introduction

Stability assessment of non-linear systems and ensuring their safe and reliable operation are of paramount importance in any real-world engineering system. While learning-based control schemes have received a lot of attention recently, the lack of stability guarantees is a fundamental issue that prevents their wide-scale deployment in the real world. One standard approach to estimate the stability region of a general nonlinear system is to first find a Lyapunov function for the system and characterize its region of attraction (ROA) as the stability region (Khalil 2015). A closed-loop system is stable in the sense of Lyapunov if the system trajectory converges to the origin as long as the initial condition is inside the ROA. The sum-of-squares approach is one popular method for finding a Lyapunov function for a dynamical system (Parrilo 2000; Henrion and Garulli 2005; Jarvis-Wloszek et al. 2003; Topcu et al. 2009; Topcu, Packard, and Seiler 2008). However, sum-of-squares approaches typically do not scale well to large systems since a large number of semidefinite programs need to be solved for the sum-of-squares decomposition of polynomial systems even with a few states (Parrilo 2000). Another approach is to employ local linearizations and use quadratic approximations to find Lyapunov functions. However, this approach is typically conservative in the sense that stability can only be certified in a small vicinity of an equilibrium point of a nonlinear system, which may be insufficient to cover the normal range of operation in several practical applications (Cheng, Guo, and Huang 2003; Huang, Gao, and Xie 2021).

Recently, a line of works has used learning-based approaches that make use of the function approximation capability of a neural network for finding the Lyapunov function (Kolter and Manek 2019; Chang, Roohi, and Gao 2019). The key idea is to use supervised learning to find a neural Lyapunov function (NLF) by minimizing a loss function that captures the Lyapunov constraints. The NLF approaches have shown impressive empirical performance in estimating the stability region for nontrivial nonlinear systems. However, training an NLF requires a large number of data samples and gradient update steps, which makes the real-time training infeasible. So, a typical fix is to use the system model or pre-collected data from the real-world system and perform offline training. The trained NLF can then be used for the stability estimation of the real-world system. However, this approach will fail if the real-world system dynamics is different from the model used for training the NLF. At the same time, the real-world system model can be different from the model estimated from the collected data due to various reasons, such as estimation error and changes in the system parameters over time. Repeating the training procedure every time whenever there is such a parametric mismatch turns impractical due to the unavailability of necessary data samples and the need to get a quick stability assessment. Thus, *learning a neural Lyapunov function for a real-world system using only a small number of data samples and through a few gradient updates*, remains an open problem.

In this paper, we address the problem of stability assessment of a closed-loop system, focusing on the setting where the real-world system parameters are different from the offline model parameters. Our goal is to learn a NLF that : (i) quickly adapts to a new system instance under parametric shifts, (ii) and does so with low adaptation-time data requirements. We follow a meta-learning procedure, and in particular, the framework of Model Agnostic Meta-learning (MAML) (Finn, Abbeel, and Levine 2017), to learn such a meta-neural Lyapunov function (meta-NLF). We compare the performance of the meta-NLF approach with other baseline methods on various benchmark control systems to demon-

strate the efficacy of our method. Our codes and appendix are available at https://github.com/amitjena1992/Meta-NLF.

Related Work

Computing Lyapunov functions is one of the classical techniques for estimating the stability of non-linear systems. Various methods try to learn Lyapunov functions as polynomials (Kapinski et al. 2014; Ravanbakhsh and Sankaranarayanan 2016), support vectors (Khansari-Zadeh and Billard 2014), Gaussian processes (Umlauft, Pöhler, and Hirche 2018) and temporal logic formulae (Jha et al. 2017). A few widely-adopted ones among such methods are Sum of Squares (SOS) based polynomial approaches (Parrilo 2000), and quadratic Lyapunov functions (Cheng, Guo, and Huang 2003).

To overcome a few existing shortcomings of model-based approaches, a set of recent works (Richards, Berkenkamp, and Krause 2018; Kolter and Manek 2019; Chang, Roohi, and Gao 2019; Taylor et al. 2019; Choi et al. 2020; Mehrjou, Ghavamzadeh, and Schölkopf 2020; Jin et al. 2020; Dai et al. 2021; Dawson, Gao, and Fan 2022) employ neural networks to approximate Lyapunov functions. The core of these works is to train a neural network to optimize a loss function that captures the Lyapunov constraints. These methods require either the dynamical system's closed-form expression or offline trajectory data thereof (Boffi et al. 2021). Such methods usually perform well but remain susceptible to parameter shifts in dynamical systems.

Meta-learning provides a framework where a machine learning model gathers experience over multiple training tasks and leverages this experience to better its future learning performance on similar tasks (Schmidhuber, Zhao, and Wiering 1997; Bengio, Bengio, and Cloutier 1990; Thrun and Pratt 1998; Hochreiter, Younger, and Conwell 2001; Younger, Hochreiter, and Conwell 2001). In particular, Model Agnostic Meta-learning (MAML) (Finn, Abbeel, and Levine 2017) is a well-recognized method that is used in many areas, including computer vision (Achille et al. 2019), natural language processing (Huang et al. 2018), reinforcement learning (Nagabandi et al. 2018) and system identification and control (Shi et al. 2021). However, to the best of our knowledge, a meta-learning-based stability assessment of dynamical systems has not been addressed before.

Preliminaries

We consider a continuous-time (closed-loop) autonomous dynamical system of the form

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

where the state $x \in X \subset \mathbb{R}^d$ is fully observed, and the dynamics is represented by the continuous function $f_\vartheta : \mathbb{R}^d \to \mathbb{R}^d$ where $\vartheta \in \mathbb{R}^\mathfrak{d}$ is the parameter that characterizes the dynamics. For any given initial condition x_0, we assume that this dynamics gives a unique trajectory $x(t; x_0), t \geq 0$, with $x(0; x_0) = x_0$. In the following, we simply represent the trajectory as $x(t)$, making the dependence on the initial condition implicit.

A closed-loop system is said to be stable (in the sense of Lyapunov) at the origin ($x \equiv 0$) if for any $\epsilon > 0$; there exists a $\delta > 0$ such that whenever $\|x(0)\| < \delta \implies \|x(t)\| < \epsilon, \ \forall t > 0$. The system is said to be asymptotically stable at the origin if it is stable and there exists a $\delta > 0$ such that $\lim_{t\to\infty} \|x(t)\| = 0$ for any initial condition $\|x(0)\| < \delta$. One traditional way to certify the asymptotic stability of a system is through a Lyapunov function.

A continuously differentiable function $V : D \to \mathbb{R}$, $D \subset X$, is called a (strict) Lyapunov function for the system (1) if it satisfies the following conditions:

$$V(0) = 0, \tag{2a}$$

$$V(x) > 0, \ \forall x \in D \setminus \{0\}, \tag{2b}$$

$$\dot{V}(x) = \nabla V(x)^\top f(x) < 0, \ \forall x \in D \setminus \{0\}. \tag{2c}$$

If V is the Lyapunov function for the system (1), then the system is asymptotically stable, (Khalil 2015). The region D is called the valid region which contains the ROA in the form of the largest level set of V.

In general, sum-of-squares approaches (Parrilo 2000; Henrion and Garulli 2005; Jarvis-Wloszek et al. 2003; Topcu et al. 2009; Topcu, Packard, and Seiler 2008) can be used to assess the stability of nonlinear systems. Alternatively, local linearizations of the dynamics may be used to compute a quadratic Lyapunov function (Chiang 1989). However, these approaches often fail to find a meaningful Lyapunov function and ROA for large-scale, high-dimensional, networked systems because of the following challenges. Firstly, sum-of-squares approaches do not scale well computationally and will quickly become intractable when the system dimension increases (Parrilo 2000; Henrion and Garulli 2005; Jarvis-Wloszek et al. 2003; Topcu et al. 2009; Topcu, Packard, and Seiler 2008). Secondly, sum-of-squares and quadratic approximation-based approaches are typically very conservative in their estimation of ROA even for small systems (Chang, Roohi, and Gao 2019). This may lead to the design of conservative controllers and sub-optimal system operation.

In order to overcome these challenges, some recent works have exploited the data-based function approximation capability of a neural network to *learn* Lyapunov functions for nonlinear systems (Kolter and Manek 2019; Chang, Roohi, and Gao 2019). The goal is to learn an NLF V_θ where θ is the parameter of the neural network, which represents the Lyapunov function. In particular, Chang, Roohi, and Gao (2019) has defined the Lyapunov loss function as

$$L(\theta) = \mathbb{E}_{(x,y)\sim\rho_\vartheta}[\ell(\theta, (x, y))], \text{ where} \tag{3}$$

$$\ell(\theta, (x, y)) = [\max(0, -V_\theta(x)) + \max(0, \nabla V_\theta(x)^\top y) + V_\theta^2(0)], \tag{4}$$

with $y = f_\vartheta(x)$ and $\rho_\vartheta(x)$ being a joint distribution over (x, y). The loss function $\ell(\cdot, \cdot)$ is defined to translate each condition in (2) to an equivalent loss. In particular, the first term is to ensure the condition (2b) that the V_θ is non-negative, the second term is to ensure the condition (2c) that the Lie derivative of V_θ is negative and the third term enforces the condition (2a) that $V_\theta(0)$ is zero. The learning algorithm then finds a parameter by minimizing the empirical loss function

$$\widehat{L}(\theta, S) = \frac{1}{|S|} \sum_{z\in S} \ell(V_\theta, z), \tag{5}$$

where each $z = (x, y) \in S$ is generated according to the distribution ρ_ϑ with $y = f_\vartheta(x)$. The NLF approaches have shown impressive empirical performance in estimating the stability region for nontrivial nonlinear systems (Kolter and Manek 2019; Chang, Roohi, and Gao 2019; Huang, Gao, and Xie 2021).

Meta-Neural Lyapunov Function

The NLF approach follows a standard supervised learning method that requires a large number of training data samples from a fixed system and a large number of stochastic gradient descent-based parameter updates to learn the Lyapunov function. So, the NLF training can only be done offline, and the trained NLF can then be used to certify the stability of the real-world system. However, the operating conditions of many real-world engineering systems can vary over time, which will also change their closed-loop system behavior. For example, in electricity distribution systems, the load pattern, topology, and line impedance can change due to the presence of rooftop photovoltaics (PVs), electric vehicles, battery storage, power electronics devices, and other uncertainty-inducing components. This presents two main challenges for the NLF approach to be used for certifying the stability of the real-world systems: (i) data used for the offline training of the NLF may not represent the behavior of the real-world system, and hence the pre-trained NLF may not give the correct stability certificate for this system, (ii) it may not be possible to collect enough data from the real-world system and perform a large number of gradient updates to learn a new NLF for this system. We propose to overcome these challenges with a meta-learning approach. In particular, through the model-agnostic meta-learning (MAML) (Finn, Abbeel, and Levine 2017), we seek to formulate a meta-NLF capable of adapting to the real-world system with a small number of training samples and gradient steps. In what follows, we refer to the meta-NLF and its task-adapted NLF as V_θ and $V_{\theta'}$ respectively where θ' results from a K-step adaptation of the meta-parameter θ on a given dynamical system.

Verification of Meta-NLF

For a Lyapunov function to be valid, constraints (2a), (2b) and (2c) must satisfy everywhere in D. But, the minimization of the Lyapunov loss function (as stated in (4)) doesn't automatically translate to the satisfaction of Lyapunov constraints in D. Thus, a verification process such as an SMT solver (Chang, Roohi, and Gao 2019) or the Lipschitz technique (Richards, Berkenkamp, and Krause 2018) becomes necessary. Using a logic formula that is made up of negation of all Lyapunov constraints, the SMT solver filters out state vectors that satisfy the logic formula *or violate the Lyapunov conditions*. However, from an implementation standpoint, this method doesn't scale well to larger systems, as reported in (Jena et al. 2022). Hence, we adopt the Lipschitz method introduced in (Richards, Berkenkamp, and Krause 2018). The idea here is that in order to enforce $\dot{V}(x) < 0$ for $x \in D$, the tightened condition $\dot{V}(x) < -K\tau = -\epsilon_2$ for finitely many samples in D serves as a sufficient condition, where the time derivative of the Lyapunov function $\dot{V}(x)$ is assumed to be a $K-$Lipschitz function (with respect to 1-norm) and $\tau > 0$ signifies how densely the samples cover D. We make use of this result to ensure the time derivative of a task-adapted NLF stays valid across D once it meets the tightened conditions at each node in a fine-grained discretized grid in D. A detailed proof of this result can be found in (Berkenkamp et al. 2017). For positive definiteness, we argue along similar lines that for a Lipschitz task-adapted NLF $V_{\theta'}$, if the tightened condition $V_{\theta'}(x) > \epsilon_1 > 0$ holds for finitely many sample points in D, then $V_{\theta'}$ is positive definite throughout D. Proof of this result is provided in the appendix. Finally, we use the same technique as in (Chang, Roohi, and Gao 2019), and subtract a bias term $V_{\theta'}(0)$ from a task-adpated NLF to ensure the satisfaction of condition (2a). For meticulously-optimized meta-NLFs, this bias term attains a small value on any training-time or test-time system, thus not affecting the stability performance significantly.

Next, we modify the loss function presented in (4) to account for the new tightened versions of (2b) and (2c) as the following:

$$\ell(\theta, (x, y)) = [\max(0, \epsilon_1 - V_\theta(x)) + \max(0, \epsilon_2 + \nabla V_\theta(x)^\top f_\vartheta(x)) + V_\theta^2(0)]. \quad (6)$$

Here, we consider ϵ_1 and ϵ_2 tunable hyperparameters. In addition to including these constants in the loss function, we also rigorously check the tightened conditions on a discretized grid in D once a trained meta-NLF adapts to a test-time system. We provide a visual presentation of this validation in Figure 1d where if the task-adapted NLF satisfies the tightened Lyapunov constraints at a node in the grid, the node is marked green and marked red otherwise.

Next, we provide the Lipschitz assumptions on the dynamics $f(x)$, meta-NLF $V_\theta(x)$, and gradient of meta-NLF ∇V_θ. It's straightforward to prove that Lipschitzness of $\nabla V_\theta(x)$ and $f(x)$ result in $\dot{V}_\theta(x) = \nabla V_\theta(x)^T f(x)$ being Lipschitz.

Assumption 1. For any $x_1, x_2 \in X$, the following satisfy for any $\theta \in \Theta$:

$$|V_\theta(x_1) - V_\theta(x_2)| \le K_V \|x_1 - x_2\|, \quad (7a)$$

$$\|\nabla V_\theta(x_1) - \nabla V_\theta(x_2)\|_1 \le K_{\nabla V} \|x_1 - x_2\|_1, \quad (7b)$$

$$\|f(x_1) - f(x_2)\|_1 \le K_f \|x_1 - x_2\|_1 \quad (7c)$$

where K_V, $K_{\nabla V}$ and K_f are positive Lipschitz constants and $\|.\|_1$ is the standard ℓ_1 norm.

According to the above assumption, the Lipschitzness is independent of the choice of θ. Thus, it's easy to verify that the task-adapted NLF $V_{\theta'}(x)$ and its gradient will naturally inherit the Lipschitz property from the meta-NLF.

Problem Formulation

The MAML framework learns a meta-parameter using the data from multiple tasks. In our context, a task i corresponds to learning the NLF for the system $\dot{x} = f_{\vartheta_i}(x)$ with parameter ϑ_i. For any i-th task, we assume that the algorithm has access to a dataset S_i consisting of m_i mini data-batches $S_i = \{(S_{i,j}^{\text{tr}}, S_{i,j}^{\text{te}})\}_{j=1}^{m_i}$. Each mini data-batch $(S_{i,j}^{\text{tr}}, S_{i,j}^{\text{te}})$ comprises of K and J number of samples, where each sample $z \in S_i$ is $z = (x, y) \sim \rho_i$ with $y = f_{\vartheta_i}(x)$. The task-specific

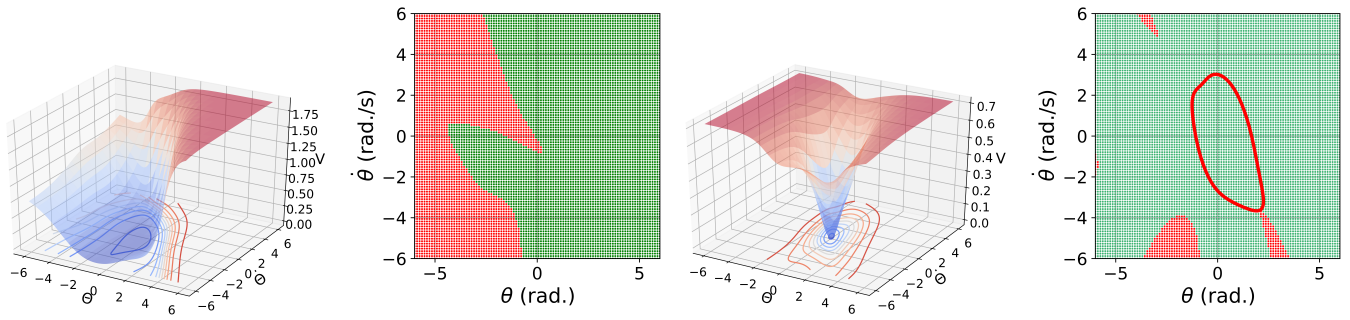

Figure 1: *Plots describing the transition of meta-NLF into a task-adapted NLF*. All figures correspond to the Inverted Pendulum setup with stochastic l. A well-calibrated meta-training, results in the meta-NLF shown in (a). In (b), the area in D where the meta-NLF satisfy the Lyapunov constraints is colored in green, and the other region is colored in red. A test-time system and a few samples thereof lead to the task-adapted NLF in (c). The valid and invalid regions (Similar to Figure 1b.) for the task-adapted NLF are presented in (d), along with an ROA shown as a contour in red. Although the meta-NLF trains with the Lyapunov loss across a set of system instances, it itself doesn't serve as a Lyapunov function on any test-time system which is evident from Figure. 1b, where sketching an ROA isn't possible. However, a one-step gradient update leads to the task-adapted NLF whose stability assessment performance considerably improves than that of the meta-NLF.

loss function L_i is defined as $L_i(\theta) = \mathbb{E}_{z\sim\rho_i}[\ell(\theta, z)]$, where $\ell(\cdot,\cdot)$ is as defined in (6). The empirical loss function $\widehat{L}_i$ is then defined as in (5). During the training, the MAML algorithm uses the data from n tasks (containing a total of $\sum_{i=1}^{n} m_i = m$ mini data-batches). The goal is to learn the meta-parameter that can quickly adapt to a new task by using k-step of stochastic gradient descent with a batch of size K.

To formally introduce this problem, define the function $\hat{\mathcal{L}}_{i,j}(\theta)$, which captures the performance of the meta-parameter θ for task i once it is updated by a single step of gradient descent on $S_{i,j} = (S^{\text{tr}}_{i,j}, S^{\text{te}}_{i,j})$,

$$\hat{\mathcal{L}}_{i,j}(\theta) = \hat{L}_i(\theta - \alpha\nabla\hat{L}_i(\theta, S^{\text{tr}}_{i,j}), S^{\text{te}}_{i,j}). \tag{8}$$

Next, for any i-th task, we define $\mathcal{L}_i(\theta) = \mathbb{E}_{S_{i,j}\sim\rho_i^{K+J}}[\hat{\mathcal{L}}_{i,j}(\theta)]$. The MAML problem is to find the optimal meta-parameter that performs well after one step of adaptation on each task within a given set of tasks. This is posed as the optimization problem $\min_\theta \mathcal{L}(\theta)$, where $\mathcal{L}(\theta) = \mathbb{E}_{i\sim[n]}[\mathcal{L}_i(\theta)]$. The MAML algorithm then solves the empirical problem $\min_\theta \widehat{\mathcal{L}}(\theta, S)$ where $\widehat{\mathcal{L}}(\theta, S) = \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\widehat{\mathcal{L}}_{i,j}(\theta)$

Meta-NLF Training and Adaptation

Our motivation is to learn a meta-NLF that can be adapted to learn an NLF for the real-world system whose dynamics can be different from the nominal system model $\dot{x} = f_{\vartheta_o}(x)$ where ϑ_o is the parameter of the nominal system. For training, we consider multiple tasks by sampling the system parameters $\vartheta_i \sim \mathcal{N}(\vartheta_o, \Sigma_\vartheta)$. For $i-$th task pertaining to parameter ϑ_i, we sample the state $x_{i,j}$ from the domain D and get $y_{i,j} = f_{\vartheta_i}(x_{i,j})$ to constitute the data sample $z_{i,j} = (x_{i,j}, y_{i,j})$. A collection of such data samples $(z_{i,j})_{j=1}^{(K+J)m_i}$ form the dataset $S_i = \{(S^{\text{tr}}_{i,j}, S^{\text{te}}_{i,j})\}_{j=1}^{m_i}$. The training is similar to the standard MAML training, which we summarize in Algorithm 1. The output of the algorithm is a meta-parameter θ_{mnlf} that builds the meta-NLF $V_{\theta_{\text{mnlf}}}$.

Algorithm 1: Meta-NLF Training Algorithm

Input: n closed-loop systems $\dot{x} = f_{\vartheta_i}(x), 1 \le i \le n$, and the corresponding datasets $\{S_i\}_{i=1}^n$, adaptation and meta step sizes $(\alpha, \tilde{\alpha})$, meta-training batch size P, total meta-training steps $\mathcal{K}$.
Initialize: Meta-parameter θ^0
for $k = 0, \ldots, \mathcal{K}-1$ **do**
  for $p = 1, \ldots, P$ **do**
    Randomly sample a task index i_p in $\{1, \ldots, n\}$.
    Randomly select a mini-batch $S_{i_p,j_p} = (S^{\text{tr}}_{i_p,j_p}, S^{\text{te}}_{i_p,j_p})$ from all mini-batches of i_p-th task.
    Adaptation: $\theta^k_{i_p} \leftarrow \theta^k - \alpha\nabla\hat{L}_{i_p}(\theta^k, S^{\text{tr}}_{i_p,j_p})$
  end for
  Meta-parameter update:

$$\theta^{k+1} = \theta^k - \tilde{\alpha}\nabla\frac{1}{P}\sum_{p=1}^{P}\hat{L}_{i_p}(\theta^k_{i_p}, S^{\text{te}}_{i_p,j_p})$$

end for
Output: $\theta_{\text{mnlf}} = \theta^{\mathcal{K}}$

During testing, we assume that a new system with parameter ϑ_{n+1} is realized, and a dataset $S_{n+1} = (S^{\text{tr}}_{n+1})$ has been made available. We then perform a k-step adaptation on this task with the meta-parameter as the initialization as,

$$\theta^{k+1}_{n+1} \leftarrow \theta^k_{n+1} - \alpha\nabla\hat{L}(\theta^k_{n+1}, S^{\text{tr}}_{n+1}), \tag{9}$$

with $\theta^0_{n+1} = \theta_{\text{mnlf}}$. A set of figures illustrating the transition from a meta-NLF into a task-adapted NLF is shown in Figure 1. These figures and the simulation results of other experiments communicate a central message that a well-trained

meta-NLF tries to find an optimal initialization parameter θ_{mnlf} from which less conservative task-adapted NLFs can be computed in k steps. However, the meta-NLF itself doesn't aim to satisfy the Lyapunov conditions strictly, thus being undeployable for any test-time system without needing further test-time gradient updates.

Experiments

We demonstrate the performance of meta-NLF with other standard stability assessment methods on various closed-loop dynamical systems following definition (1). Each system is already equipped with a controller, such as an LQR controller or a power system-specific droop controller (Yu et al. 2015). Next, for parametric uncertainty, we assume a few system parameters to take random values in a pre-specified range. For example, the length of an inverted pendulum randomly taking a value in $[0.5, 1.0]$ can cause parametric uncertainty. To make our setup as realistic as real-world's, we assume the following:

- The meta-distribution function P_t that governs the uncertainty of parameter vector ϑ is unknown.
- The parametric uncertainty is experienced only through a gap between the deterministic parameter vectors ϑ_0 and ϑ_{n+1} that instantiate the nominal and test-time systems respectively.

Baseline Methods

The baseline methods include three non-adaptive Lyapunov functions, namely the Sum of Squares-based Lyapunov Function (SOS-LF), the Quadratic Lyapunov Function (QLF), and the standard Neural Lyapunov Function (NLF). In several numerical experiments, we have observed that when there is a substantial gap between ϑ_0 and ϑ_{n+1}, each non-adaptive method that's been trained on the nominal system results in a very conservative ROA estimate when directly evaluated (without any further gradient updates) on the test-time system. Furthermore, this approach often doesn't yield an ROA at all. Hence, for a meaningful performance comparison purpose, we have given the non-adaptive methods a significant advantage by making the test-time system available to them. We change the notations of the baselines to **SOS-LF(TS)**, **QLF(TS)**, and **NLF(TS)** to indicate their access to the test-time system. Lastly, an adaptive baseline method is created by training an NLF on the nominal system and then naively transferring (with a few gradient updates) to the test-time system. We call this the transfer learning-based NLF **(T-NLF)**. To summarize, each method experiences the nominal and test-time systems as per the following setting:

- QLF(TS), SOS-LF(TS), and NLF(TS) never see the nominal system but only gain explicit access to the test-time system
- T-NLF gets full access to the nominal system for neural network training. Then, it gets little access to the test-time system where a very small dataset (e.g. 50 training samples) is available for quick fine-tuning (e.g. 10 gradient step updates).
- In meta-NLF, the nominal system is always available. A set of similar system instances are created around the nominal one for meta-training. A fully meta-trained NN is then given access to the test-time system in the same manner as T-NLF.

Selection of a Valid Region *D*

During meta-training, it's important to select a valid region D in which a meta-NLF corresponding to each training system will stay valid. Due to the lack of scalability to larger systems, unlike the original NLF approach (Chang, Roohi, and Gao 2019), we don't use an SMT solver to select D. Instead, we follow an alternative procedure to create an appropriate D. During the meta-training, we start with a choice of $D := \{x : \|x\|_2 \leq d\}$ to train V_θ, where the radius d is a tunable hyperparameter. Then, after a full course of meta-training, we check the tightened conditions for each task-adapted NLF $V_{\theta_i}, i \in \{1, \ldots, n\}$ on a fine-grained rectangular grid in the same D. If this D contains regions where any V_{θ_i} violates at least one of the tightened Lyapunov conditions, then we shrink D by reducing radius d and repeat the whole procedure of meta-training. We continue this process until all V_{θ_i}s are completely valid in our choice of D. Finally, if no such D exists, the algorithm is concluded to fail to obtain a suitable meta-NLF. For T-NLF and NLF(TS), we follow the same series of steps to compute an appropriate D. We admit that our procedure of selecting a D is computationally expensive as it involves validating each task-adapted NLFs after the convergence of meta-training. Designing an algorithm that more efficiently selects a D is a part of our future work.

Why an Adaptive Lyapunov Function is Needed?

From a deployment standpoint, among all the above methods (including meta-NLF), QLF(TS) can be computed in the least time owing to the linearization-based simplification of the dynamics. SOS-LF(TS) comes next in terms of the computational time required to generate a Lyapunov function. However, both QLF(TS) and SOS-LF(TS) are typically conservative in computing the ROAs, and the latter needs the dynamics to be polynomial, which is a stringent requirement. While NLF(TS) overcomes the conservative performance issue, it requires a lot of gradient descent steps and a large number of training samples from the test-time system. Given the safety-critical nature of many applications, relying on NLF(TS) will delay the stability assessment process which in turn might lead to a catastrophic outcome, e.g., a city-wide blackout in the context of power systems. Hence, an adaptive Lyapunov function is needed, which doesn't compromise on the performance side and swiftly adapts to any arbitrary test-time system. We have also employed T-NLF for this purpose only to observe the simulation outcomes contrary to our expectations *when a few training samples from the test-time system are available*. For significantly mismatched nominal and test-time systems, T-NLF's ROA falls behind the ROAs of QLF(TS) and SOS-LF(TS). We recall that the first step of T-NLF computation is training an NLF on the nominal system, which necessitates a huge amount of training data and time. However, T-NLF doesn't give a significant benefit in return. Finally, we argue that meta-NLF, by virtue of

a systematic meta-training, will facilitate fast-adaptive and less conservative stability assessment on any test-time system. Note that to make a fair comparison with meta-NLF, we don't use the SMT solver-based numerical checker in either T-NLF or NLF(TS).

In the end, we evaluate meta-NLF from two perspectives: i) Can meta-NLF produce a comparable or larger ROA than non-adaptive methods that are explicitly computed on a test-time system? (ii) Can meta-NLF adapt to a test-time system better than T-NLF and fetch a less conservative ROA?

Simulations

We compare meta-NLF's performance with other baselines by matching their ROAs on a test-time system. In each test case, the trained meta-NLF updates into a task-adapted NLF upon facing a test-time system. First, we consider the inverted pendulum environment, where the system parameters are the pendulum's length (l) and mass (m), acceleration due to gravity (g), and coefficient of friction (b). The system states are angular displacement from the upright position (θ) and angular velocity ($\dot{\theta}$). The standard LQR controller closes the control loop so that the system turns ready for stability assessment. Next, we build a test case on this system by assuming l to be stochastic and create the nominal and test-time systems by setting $\vartheta_0 = (l, m, g, b)_0 = (0.5, 0.15, 9, 81, 0.1)$ and $\vartheta_{n+1} = (1.2, 0.15, 9.81, 0.1)$. In Figure 2a, we overlay the phase portrait of the system with task-adapted NLF's ROA to notice the latter capturing a region where the system tends to gravitate towards the origin. This testifies to the accuracy of our meta-learning-based stability assessment, which we again cross-check through a time domain simulation in Figure 2b. Here, the system converges to the origin from random starting points inside the task-adapted NLF's ROA. Next, we move to Figure 2c, where the task-adapted NLF is comparable with QLF(TS) and SOS-LF(TS) but significantly falls behind NLF(TS) in terms of ROA area. T-NLF fails to capture an ROA in this case which suggests the superior adaptation of meta-NLF over the former method. We remark that for small dimensional systems with a few stochastic parameters (e.g., this setting where l is stochastic), meta-NLF's full potential might not be quite evident. We perform additional experiments (see the Appendix) on the same system by assuming more parameters to be stochastic and find results where the task-adapted NLF convincingly beats SOS-LF(TS) and QLF(TS) in performance. In all our experiments, we observe the NLF(TS) to generate the largest ROA among all methods. This happens because NLF(TS) is a Lyapunov function that exclusively trains to assess the stability of the test-time system instead of adapting to it.

For our second and third systems, we take two power system examples with N microgrids that use droop control schemes to stabilize the evolution of phase angles. The state x captures modified phase angle $[\Delta\delta_1, \Delta\delta_2, \ldots, \Delta\delta_N]$, where $\Delta\delta_j = \delta_j - \delta_j^*$ denote the difference between the j-th phase angle and its setpoint. Among all parameters, the droop constants $(\mathrm{dc}_1, \ldots, \mathrm{dc}_N)$, which regulate the strength of subsystem-wise control loops, are chosen to be stochastic. We start off with a three-microgrid system with stochastic dc_1 and dc_2, and set $\vartheta_0 = (\mathrm{dc}_1, \mathrm{dc}_2, \mathrm{dc}_3)_0 = (2.0, 2.0, 2.0)$ and $\vartheta_{n+1} = (3.0, 4.3, 2.0)$. As illustrated in Figure 2d, we find the ROA of the task-adapted NLF (derived from meta-NLF) to be larger than that of QLF(TS) and SOS-LF(TS) but smaller than NLF(TS)'s ROA, while T-NLF doesn't succeed in sketching an ROA. Next, we consider a five-microgrid system and make all droop constants stochastic. In this case, the nominal and test-time parameters are $\vartheta_0 = (\mathrm{dc}_1, \ldots, \mathrm{dc}_5)_0 = (2.0, 2.0, 2.0, 2.0, 2.0)$ and $(\mathrm{dc}_1, \ldots, \mathrm{dc}_5)_{n+1} = (3.5, 3.5, 3.0, 4.0, 3.2)$. Figure 2e compares the ROAs of all methods, and we notice the updated task-adapted NLF leaving behind QLF(TS), SOS-LF(TS), and T-NLF in terms of performance. Specifically, the latter two methods completely fail to produce ROAs. Finally, we hypothesize the following based on the preceding simulation results: (i) despite QLF and SOS-LF being easier to compute on the test-time system, meta-NLF should be preferred over them in the interest of performance. (ii) T-NLF isn't a reliable adaptive method as its performance remains questionable when the parametric stochasticity tends to be high.

Our final system is the Caltech Ducted Fan in hover mode (Jadbabaie, Yu, and Hauser 1999). The state variables are x, y, θ that denote the horizontal and vertical positions and angular orientation of the center of the fan. The system parameters include mass (m), moment of inertia (J), gravitational constant (g), and coefficient of viscous friction (d). We assume m, r, d to be stochastic and fix $\vartheta_0 = (m, J, r, g, d)_0 = (11.2, 0.0462, 0.15, 0.28, 0.1)$ and $\vartheta_0 = (13.0, 0.0462, 0.165, 0.28, 0.15)$. Figure 2f compares all ROAs and conveys the message that the task-adapted NLF exceeds T-NLF and QLF(TS) in performance while SOS-LF(TS) method fails to converge. Similar to the previous experiments, we conclude here that meta-NLF outperforms T-NLF, QLF(TS), and SOS-LF(TS) while it stays competitive with NLF(TS).

Remark 1. For three-dimensional or larger systems taken in the experiments (in Figures 2d, 2e and 2f), we show two-dimensional projections of ROAs onto specific planes for visual comparison purposes.

Conclusion

In this work, we presented a meta-learning-based Lyapunov function that adapts well to dynamical system instances under parametric shifts. To achieve this, meta-training is conducted using a set of dynamical system instances, where each instance represents a deterministic snapshot of a system exhibiting parametric stochasticity. Upon convergence of meta-training, the meta-function (which we term the meta-NLF) faces a new system instance and adapts to it quickly (in a few gradient steps). To the best of our knowledge, this is the first work that integrates meta-learning with NLFs to formulate adaptive stability certificates. We compared meta-NLF with three non-adaptive baselines and an adaptive baseline method on different benchmark dynamical systems to highlight the superior adaptive performance of our method. One limitation of our present work is we assume a small dataset from the test-time system to be available on which our meta-function performs a few gradient update steps. In future work,

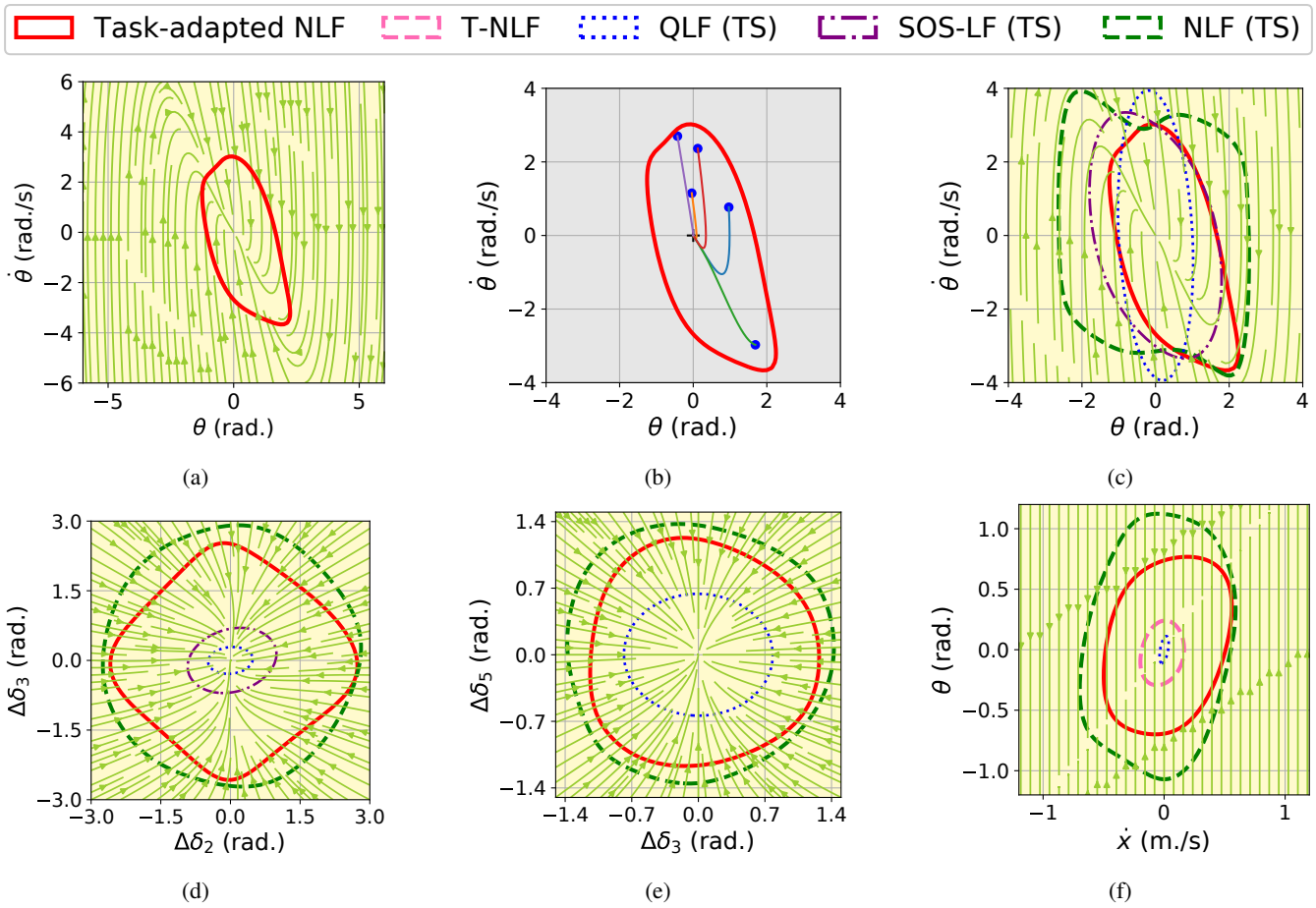

Figure 2: *ROA verification and Comparison plots.* In each figure, the phase portrait provides a sense of the system's transient behavior, which serves as a ground truth for understanding each stability assessment method's reliability and conservativeness. For a test case of an Inverted Pendulum with stochastic l, (a) verifies the ROA of the task-adapted NLF (computed from the meta-NLF on a test-time system) against the phase portrait; (b) validates task-adapted NLF's ROA through time domain simulation; (c) shows task-adapted NLF's performance with other baselines through an ROA comparison. Each subsequent plot concerns a specific test case and compares the ROAs of all methods on it. The test cases are, (d) a Three-microgrid system with stochastic droop constants $(\mathrm{dc}_1, \mathrm{dc}_2)$; (e) a Five-microgrid system with stochastic droop constants $(\mathrm{dc}_1, \ldots, \mathrm{dc}_5)$; (f) Caltech Fan with stochastic (m, r, d). We note that in Figure 2f, the system asymptotically converges to the origin even though the vector fields seem almost parallel to each other.

we will extend our method to a meta-reinforcement learning framework and try to learn a policy function that adapts to an arbitrary test-time environment and stays both optimal and stable (in the Lyapunov sense) in it. Formulations of meta-learning-based barrier functions and contraction metrics are two other exciting research areas that we plan to study.

Acknowledgements

This work is supported in part by NSF ECCS-2038963, the U.S. Department of Energy (DoE) Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office (SETO) Award Number DEEE0009031, and Texas A&M Engineering Experiment Station (TEES) Smart Grid Center.

References

Achille, A.; Lam, M.; Tewari, R.; Ravichandran, A.; Maji, S.; Fowlkes, C. C.; Soatto, S.; and Perona, P. 2019. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6430–6439.

Bengio, Y.; Bengio, S.; and Cloutier, J. 1990. *Learning a synaptic learning rule*. Citeseer.

Berkenkamp, F.; Turchetta, M.; Schoellig, A.; and Krause, A. 2017. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30.

Boffi, N.; Tu, S.; Matni, N.; Slotine, J.-J.; and Sindhwani, V. 2021. Learning stability certificates from data. In *Conference on Robot Learning*, 1341–1350. PMLR.

Chang, Y.-C.; Roohi, N.; and Gao, S. 2019. Neural lyapunov control. *Advances in neural information processing systems*, 32.

Cheng, D.; Guo, L.; and Huang, J. 2003. On quadratic Lyapunov functions. *IEEE Transactions on Automatic Control*, 48(5): 885–890.

Chiang, H.-D. 1989. Study of the existence of energy functions for power systems with losses. *IEEE Transactions on Circuits and Systems*, 36(11): 1423–1429.

Choi, J.; Castaneda, F.; Tomlin, C. J.; and Sreenath, K. 2020. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*.

Dai, H.; Landry, B.; Yang, L.; Pavone, M.; and Tedrake, R. 2021. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*.

Dawson, C.; Gao, S.; and Fan, C. 2022. Safe Control with Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction methods. *arXiv preprint arXiv:2202.11762*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Henrion, D.; and Garulli, A. 2005. *Positive polynomials in control*, volume 312. Springer Science & Business Media.

Hochreiter, S.; Younger, A. S.; and Conwell, P. R. 2001. Learning to learn using gradient descent. In *International conference on artificial neural networks*, 87–94. Springer.

Huang, P.-S.; Wang, C.; Singh, R.; Yih, W.-t.; and He, X. 2018. Natural language to structured query generation via meta-learning. *arXiv preprint arXiv:1803.02400*.

Huang, T.; Gao, S.; and Xie, L. 2021. A neural lyapunov approach to transient stability assessment of power electronics-interfaced networked microgrids. *IEEE Transactions on Smart Grid*, 13(1): 106–118.

Jadbabaie, A.; Yu, J.; and Hauser, J. 1999. Receding horizon control of the Caltech ducted fan: A control Lyapunov function approach. In *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328)*, volume 1, 51–56. IEEE.

Jarvis-Wloszek, Z.; Feeley, R.; Tan, W.; Sun, K.; and Packard, A. 2003. Some controls applications of sum of squares programming. In *IEEE Conference on Decision and Control (CDC)*, 4676–4681.

Jena, A.; Huang, T.; Sivaranjani, S.; Kalathil, D.; and Xie, L. 2022. Distributed Learning of Neural Lyapunov Functions for Large-Scale Networked Dissipative Systems. *arXiv preprint arXiv:2207.07731*.

Jha, S.; Tiwari, A.; Seshia, S. A.; Sahai, T.; and Shankar, N. 2017. Telex: Passive stl learning using only positive examples. In *International Conference on Runtime Verification*, 208–224. Springer.

Jin, W.; Wang, Z.; Yang, Z.; and Mou, S. 2020. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*.

Kapinski, J.; Deshmukh, J. V.; Sankaranarayanan, S.; and Aréchiga, N. 2014. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, 133–142.

Khalil, H. K. 2015. *Nonlinear control*, volume 406. Pearson New York.

Khansari-Zadeh, S. M.; and Billard, A. 2014. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6): 752–765.

Kolter, J. Z.; and Manek, G. 2019. Learning stable deep dynamics models. *Advances in neural information processing systems*, 32.

Mehrjou, A.; Ghavamzadeh, M.; and Schölkopf, B. 2020. Neural lyapunov redesign. *arXiv preprint arXiv:2006.03947*.

Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R. S.; Abbeel, P.; Levine, S.; and Finn, C. 2018. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*.

Parrilo, P. A. 2000. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology.

Ravanbakhsh, H.; and Sankaranarayanan, S. 2016. Robust controller synthesis of switched systems using counterexample guided framework. In *2016 international conference on embedded software (EMSOFT)*, 1–10. IEEE.

Richards, S. M.; Berkenkamp, F.; and Krause, A. 2018. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, 466–476. PMLR.

Schmidhuber, J.; Zhao, J.; and Wiering, M. 1997. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28(1): 105–130.

Shi, G.; Azizzadenesheli, K.; O'Connell, M.; Chung, S.-J.; and Yue, Y. 2021. Meta-adaptive nonlinear control: Theory and algorithms. *Advances in Neural Information Processing Systems*, 34: 10013–10025.

Taylor, A. J.; Dorobantu, V. D.; Le, H. M.; Yue, Y.; and Ames, A. D. 2019. Episodic learning with control lyapunov functions for uncertain robotic systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6878–6884. IEEE.

Thrun, S.; and Pratt, L. 1998. Learning to learn: Introduction and overview. In *Learning to learn*, 3–17. Springer.

Topcu, U.; Packard, A.; and Seiler, P. 2008. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44(10): 2669–2675.

Topcu, U.; Packard, A. K.; Seiler, P.; and Balas, G. J. 2009. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1): 137–142.

Umlauft, J.; Pöhler, L.; and Hirche, S. 2018. An uncertainty-based control Lyapunov approach for control-affine systems modeled by Gaussian process. *IEEE Control Systems Letters*, 2(3): 483–488.

Younger, A. S.; Hochreiter, S.; and Conwell, P. R. 2001. Meta-learning with backpropagation. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3. IEEE.

Yu, K.; Ai, Q.; Wang, S.; Ni, J.; and Lv, T. 2015. Analysis and optimization of droop controller for microgrid system based on small-signal dynamic model. *IEEE Transactions on Smart Grid*, 7(2): 695–705.