Blending Neural Control Density Functions for Stabilization and Safety

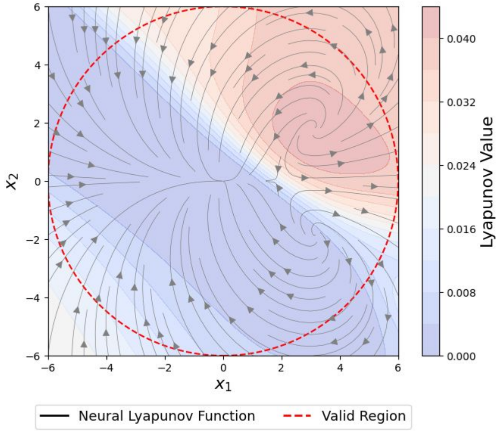
Sahil Chaudhary*, Chaitanya Murti*, and Chiranjib Bhattacharyya

sahilc@iisc.ac.in

When do Neural Lyapunov Functions Fail?

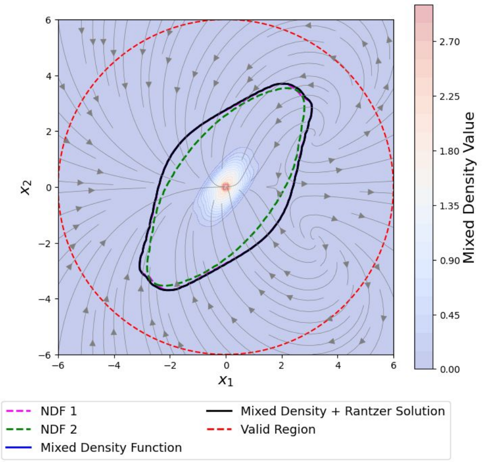
Neural networks for stability and control: powerful new tool with a variety of advantages
But Lyapunov function based methods have crucial disadvantages!

Consider the following system with four equilibria $\dot{x}_1 = -2x_1 + x_1^2 - x_2^2 \quad \dot{x}_2 = -6x_2 + 2x_1x_2$



Neural Lyapunov Control fails here **(left)**

(Spoiler) But our Neural Control Density Functions synthesize a working controller! **(right)**



Challenges with Lyapunov-based methods

Lyapunov functions are a cornerstone of control theory, **but have major drawbacks**

- Fail to certify stability when **multiple equilibria** are in the region of interest **[Rantzer et al, 2001]**

- Cannot be used to **blend controllers - finding a convex combination of controllers with state dependent weights - with a stability guarantee** - set of controller/certificate pairs is in general **not even connected**! **[Prieur and Praly, 1999]**

Density Functions offer a crucial alternative to Lyapunov Functions

Density functions **[Rantzer, 2001]** **offer a *dual* to Lyapunov Functions**

Key Result [Rantzer, 2001]:

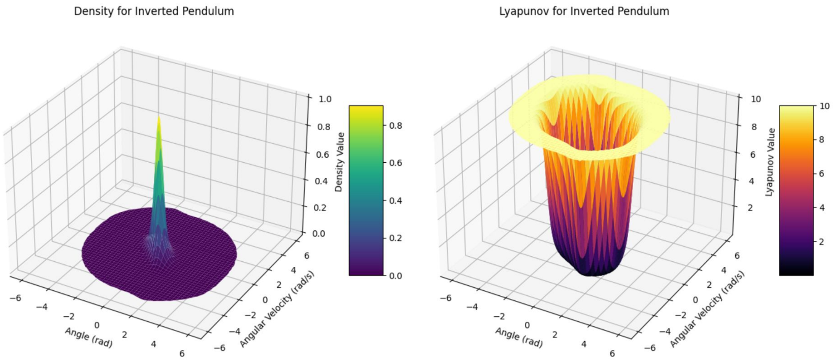
A system ***ẋ = f(x)*** is ***almost-everywhere* stable** ⇐ there exists a **density function** ρ such that

1. **ρ> 0**
2. **∫ρ*(x)f(x)/*|*x*|*dx* < ∞**
3. **div[ρ*f*](*x*) > 0.**

Almost everywhere stability ⇒ can certify stability of equilibria even when unstable equilibria are also present!

Allows for **smooth blending of nonlinear controllers** **[Rantzer and Ceragioli, 2002]**

Density functions are maximized at equilibrium, as opposed to Lyapunov functions



Challenges with Neural Density Functions

Challenge - Implementing Density Functions with Neural Networks for stabilizing *Control-Affine Systems* of the form $\dot{x} = f(x) + g(x)u$

Existing Neural Control methods with Lyapunov functions: use one Neural Network for Certificates, one for Controller

Cannot apply to density functions directly: how to enforce integrability (criteria 2)?

Key challenge: how do we parameterize Neural Density Functions to satisfy criteria 1-3?

Safety - trajectories of system remain in a **safe set**/avoid **unsafe set** is crucial for modern control applications **(i.e. robotics/navigation)**

Key Challenge: how do we train Neural Density functions to stabilize and also satisfy safety constraints?

Our Contributions

Our Goal: Design Controllers for systems of the form $\dot{x} = f(x) + g(x)u$ using (control) density functions

Our Contributions

- A **novel exponential parameterization** to learn Neural Control Density Functions (NCDFs) which certifies almost everywhere stability.
- Characterize **almost everywhere region of attraction(RoA)** certified by density functions and **show the blended controllers using density functions always improves RoA**.
- Extend NCDFs to **synthesize safe and stable controllers**
- **Our paper is available at: https://openreview.net/pdf?id=lxBsOvMFXc**
  Our project website: https://sahil-chaudhary.github.io/NCDF_Project/index.html

Learning Neural Control Density Functions

Key Challenge: Learning neural density functions that satisfy integrability condition

For control, need to learn **2 networks: *ϱ(x)*** and ***ψ(x)*** . These are called **Neural Control Density Functions.**

We use the parameterization :

$$\rho(x) = \frac{a(x)}{e^{(\|x\|^2 + \|b(x)\|^2)}}$$
$$\psi(x) = \frac{c(x)}{e^{(\|x\|^2 + \|b(x)\|^2)}}$$

$$u(x) = \frac{\psi(x)}{\rho(x)} = \frac{c(x)}{a(x)}$$

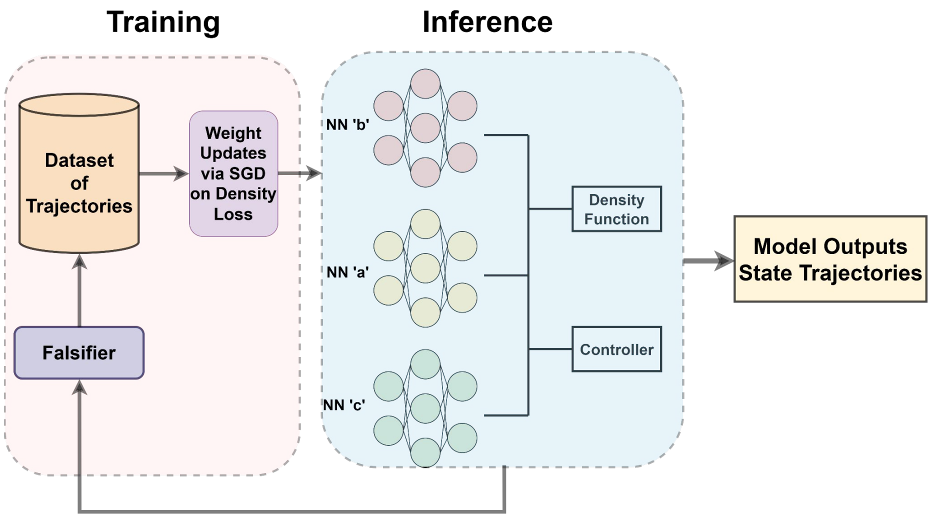
Our Result: If *a(x)* is Lipschitz, *ϱ(x)* with this parameterization satisfies integrability condition (criteria 2)

Training Neural Control Density Functions

Key Idea: create loss function - the Neural Control Density Loss - capturing criteria 1&3 for density functions **(assuming paramterization satisfies criteria 2):**

$$L(\theta) = \mathbb{E}[\max(0, -\rho_\theta(x)) + \max(0, -\nabla \cdot [\rho_\theta(x)(f(x) + g(x)u_\theta(x))])]$$

Learn using initial training data, and check against falsifier (i.e. DREAL) ⇒ generate new training samples if needed



Estimating Regions of Attraction

Regions of Attraction (ROA): largest *forward invariant* set of trajectories converting to the equilibrium $\Rightarrow$ crucial qualitative measure of controller quality

Well known that **sublevel sets of Lyapunov functions inner-approximate ROA**

Key Challenge: How do we characterize (a.e.) regions of attraction using density functions?

Lyapunov functions: Superlevel Set of $\subseteq$ Region of Attraction

Our Result:
Under mild assumptions, *superlevel sets of density functions* inner-approximate the a.e. Regions of Attraction!

Theorem 3. *Suppose $D_\tau \Rightarrow \tau$-superlevel set of ρ.
Then, a.e. ROA $\supseteq D_\tau$ if $\nabla\rho(x)^\top f(x) \geq 0$ for $x \in \partial D_\tau$*

Blending with Density Functions

Well known that **Density Functions enable smooth blending of nonlinear controllers**

[Rantzer and Ceragioli, 2001]

Blended Controllers: Given *N* different density functions ϱ_i + controllers u_i, blending gives

$$\rho^*(x) = \sum_{i=1}^{N} \rho_i(x), \quad u^*(x) = \sum_{i=1}^{N} \frac{\rho_i(x) u_i(x)}{\rho^*(x)}$$

Key Challenge: How does blending impact the ROA?

Our Result: All Superlevel sets of $\varrho_i \subseteq$ Superlevel set of ϱ^*

Given: $D_i \Rightarrow$ ROA of u_i for each i, $D^\star \Rightarrow$ ROA of $u^\star$. Then,

Union of $D_i \subseteq D^\star$

Safe Stabilization

Safety is crucial to several modern control applications (i.e. robotics)

Key Challenge: Can we synthesize Stabilizing *and* Safe controllers using density functions?

Key Idea: Use Control Barrier Functions:

Given **safe set** $C := \{x : h(x) > 0\} \Rightarrow$ control u is safe if $L_f h(x) + L_g h(x) u(x) \geq -\alpha(h(x))$

Allows us to define new loss function:

$$L_{\text{safe}}(\theta) = L(\theta) + \mathbb{E}[\max(0, -[\rho(x)[L_f h(x) + \alpha(h(x))] + \psi(x) L_g h(x)])]$$

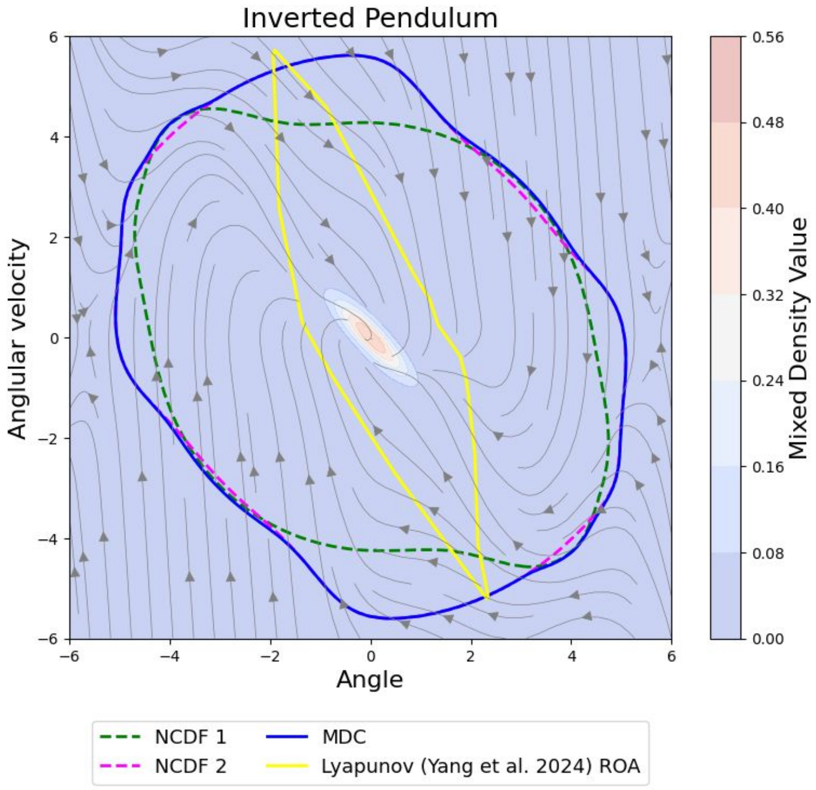
Experiments: NCDFs and blending achieve SOTA Regions of Attraction!

Consider the problem of
inverted pendulum:

$$\dot{\theta} = \dot{\theta}, \quad \ddot{\theta} = \frac{MgL\sin(\theta) - \mu\dot{\theta}}{ML^2} + \frac{u}{ML^2}$$

NCDFs achieve better (a.e.) ROAs than Neural Lyapunov Controllers

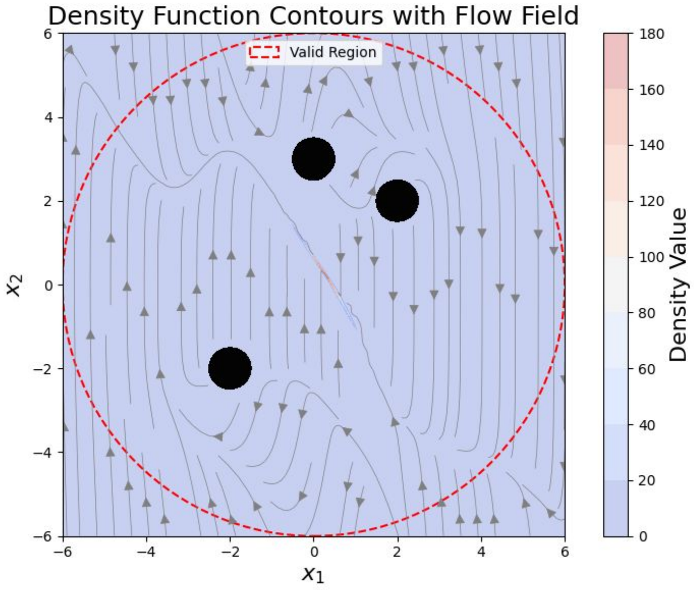
Blended controllers significantly improve a.e. ROA

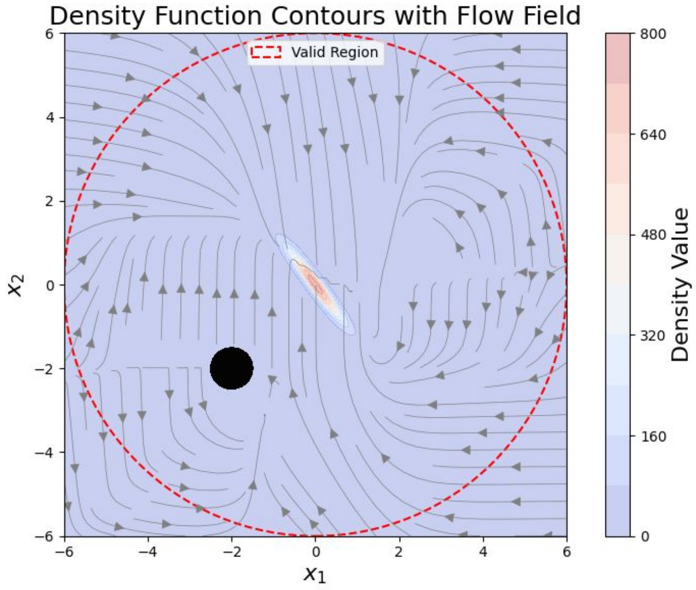


Experiments: NCDFs for **safe stabilization**

Goal: Avoid unsafe object while moving to the point of stability

Density functions offer a practical way to achieve this!





Conclusions and Takeaways

- Neural Lyapunov Functions are powerful tools for control, but have drawbacks:
  - Cannot handle multiple equilibria
  - Cannot be used to smoothly blend nonlinear controllers with stability certificate
- **Our solution: Neural Control Density Functions (NCDFs)**
- Propose a **novel exponential parameterization for NCDFs**, which are then trained with a falsifier in the loop

  Can also be extended to **stabilizing and safe controller synthesis.**
- **Superlevel sets of Density Functions inner-approximate almost-everywhere ROAs**, and the ROA of **blended controllers always subsumes those of individual controllers.**

Thanks for your time!

Our paper is available at:
https://openreview.net/pdf?id=IxBsOvMFXc

Our project website:
https://sahil-chaudhary.github.io/NCDF_Project/index.html