Guaranteed and Safe Learning Methods

Wanjiku A. Makumi, Hannah M. Sweatland, Emily J. Griffis, and Warren E. Dixon

Lyapunov-based Adaptive Deep Learning for Approximate Dynamic Programming

Wanjiku A. Makumi, Omkar Sudhir Patil, Warren E. Dixon, “Lyapunov-based Adaptive Deep Learning for Approximate Dynamic Programming”, *Automatica*, under review.

Approximate Optimal Control

- Approximate dynamic programming (ADP)
  - Optimal control & adaptive control
- Hamilton-Jacobi-Bellman (HJB) equation
  - Optimal value function
  - Unknown for nonlinear systems
- Reinforcement learning-based actor-critic framework
  - Neural networks (NNs)
    - Actor: learns control policy approximation
    - Critic: learns value function approximation
- Model-based method
    - Model knowledge required

Problem Formulation

Control affine dynamic system:

$$\dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)u(t)$$

Control objective: Design a controller u which minimizes

$$J(x,u) = \int_{t_0}^{\infty} Q(x) + u^T R u$$

Optimal value function (cost-to-go)

$$V^*(x,u) = \int_{t}^{\infty} Q(x) + u^T R u$$

Optimal control policy

$$u^*(x) = -\frac{1}{2} R^{-1} G(x)^T \nabla V^*(x)^T$$

Hamilton-Jacobi-Bellman equation

$$0 = \nabla V^*(x)\big(f(x,\theta) + g(x)u^*(x)\big) + Q(x) + {u^*}^T R u^*$$

UF UNIVERSITY of FLORIDA Duke UNIVERSITY TEXAS The University of Texas at Austin UC SANTA CRUZ

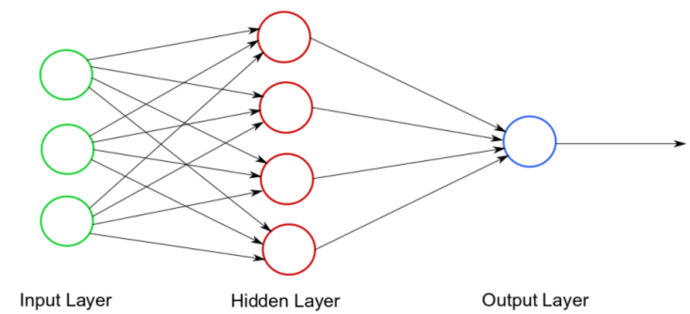
Actor-Critic Neural Networks

NN Optimal Value Function and NN Optimal Control Policy

$$V^*(x) = \boldsymbol{W}^{\boldsymbol{T}}\sigma(x) + \varepsilon(x) \qquad u^*(x) = -\frac{1}{2}R^{-1}g(x)^T\left(\nabla_x\sigma(x)^T\boldsymbol{W} + \nabla_x\varepsilon(x)^T\right)$$

$\widehat{\boldsymbol{W}}_{\boldsymbol{c}}$: Critic weight estimate

$\widehat{\boldsymbol{W}}_{\boldsymbol{a}}$: Actor weight estimate



Optimal Value Function and Optimal Control Policy Approximation

$$\hat{V}\left(x, \widehat{W}_c\right) = \widehat{\boldsymbol{W}}_{\boldsymbol{c}}{}^{\boldsymbol{T}}\sigma(x) \qquad \hat{u}\left(x, \widehat{W}_a\right) = -\frac{1}{2}R^{-1}g(x)^T\left(\nabla_x\sigma(x)^T\widehat{\boldsymbol{W}}_{\boldsymbol{a}}\right)$$

Bellman Error

Hamilton-Jacobi-Bellman Equation

$$0 = \nabla V^*(x)\big(f(x,\theta) + g(x)u^*(x)\big) + Q(x) + u^*(x)^T R u^*(x)$$

Bellman Error (BE)

$$\delta\left(x, \widehat{W}_c, \widehat{W}_a\right) =$$

$$\nabla\hat{V}\left(x, \widehat{W}_c\right)\left(\hat{f}_i(x,\theta) + g(x)\hat{u}\left(x, \widehat{W}_a\right)\right) + Q(x) + \hat{u}\left(x, \widehat{W}_a\right)^T R\hat{u}\left(x, \widehat{W}_a\right)$$

- Feedback to update the NN parameters
- Calculated along the state trajectory

BE Extrapolation

- User-defined, off-trajectory points
- Persistence of excitation (PE)
- Exploration vs exploitation

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Weight Update Laws

On-trajectory points

Off-trajectory points

Critic Weight Update Law

$$\dot{\hat{W}}_c(t) = -\eta_{c1}\Gamma\frac{\omega(t)}{\rho(t)}\delta(t) - \eta_{c2}\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i(t)}{\rho_i(t)}\delta_i(t)$$

Learning Gain Update Law

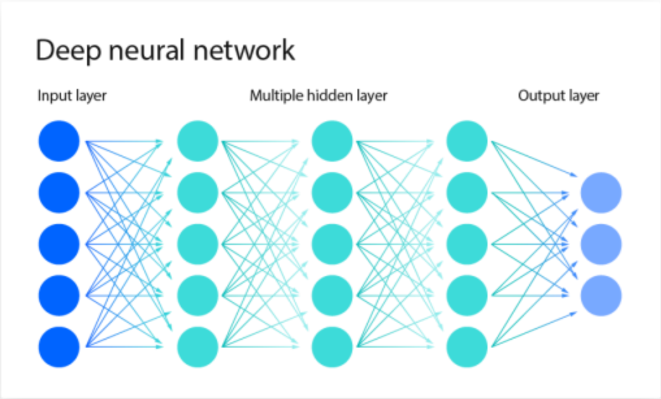
$$\dot{\Gamma}(t) = \left(\lambda\Gamma(t) - \frac{\eta_{c1}\Gamma(t)\omega(t)\omega(t)^T\Gamma(t)}{\rho(t)} - \eta_{c2}\Gamma(t)\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i(t)\omega_i^T(t)}{\rho_i(t)}\right)\Gamma(t)\right)\mathbf{1}_{\{\underline{\Gamma}\leq\|\Gamma\|\leq\overline{\Gamma}\}}$$

Actor Weight Update Law

$$\dot{\hat{W}}_a(t) = -\eta_{c1}\left(\hat{W}_a(t) - \hat{W}_c(t)\right) - \eta_{a2}\hat{W}_a(t) + \frac{\eta_{c1}G_\sigma^T(t)\hat{W}_a(t)\omega(t)^T}{4\rho(t)}\hat{W}_c(t) + \left(\frac{\eta_{c2}}{N}\sum_{i=1}^{N}\frac{G_{i\sigma}^T\hat{W}_a(t)\omega_i(t)}{4\rho_i(t)}\right)\hat{W}_c(t)$$

All-Layer Adaptive DNN

- Multi-timescale DNNs
  - Not updated via adaptive update laws
  - No guarantees on the identification of inner-layer weights
- Recent results update all weights
  - Lack of parameter convergence
- All-layer adaptive DNN update laws for ADP



DNN and RISE-Based Dynamics Observer

DNN representation

$$f(x) = \Phi(x, \theta^*) + \varepsilon(x)$$

DNN estimate

$$\Phi(x, \hat{\theta})$$

- Absence of state-derivative information
- Integrals do not help identify inner-layer weights
- Robust integral of the sign of the error (RISE)-based dynamics observer

RISE-Observer Design

$$\dot{\hat{x}} = \hat{f} + gu + \alpha_1 \tilde{x}$$

$$\dot{\hat{f}} = \tilde{x} + k_f\left(\dot{\tilde{x}} + \alpha_1 \tilde{x}\right) + \beta_f sgn(\tilde{x})$$

Observer Errors

$$\tilde{x} = x - \hat{x}$$

$$\tilde{f} = f - \hat{f}$$

Closed-Loop Observer Error System

$$\dot{\tilde{x}} = \tilde{f} - \alpha_2 \tilde{r}$$

$$\dot{\tilde{f}} = \dot{f} - k_f \tilde{f} - \tilde{r}$$

UF UNIVERSITY of FLORIDA | Duke UNIVERSITY | TEXAS The University of Texas at Austin | UC SANTA CRUZ

Adaptive Update Laws

Identification Error

$$E = \hat{f} - \Phi\left(x, \hat{\theta}\right)$$

Adaptive Update Law

$$\dot{\hat{\theta}} = \Gamma_\theta \Phi'\left(x, \hat{\theta}\right) E$$

Gain Matrix Update Law

$$\frac{d}{dt}\Gamma_\theta^{-1} = -\beta(t)\Gamma_\theta^{-1} + \Phi'^{\top}\left(X, \hat{\theta}\right) \Phi'\left(X, \hat{\theta}\right)$$

Bounded-Gain Time-Varying Forgetting Factor

$$\beta(t) = \beta_0\left(1 - \frac{\lambda_{\max}\{\Gamma_\theta\}}{\kappa_0}\right) \geq \beta_1 \in \mathbb{R}_{\geq 0}$$

If $\Phi'\left(X, \hat{\theta}\right)$ satisfies PE condition, then $\beta_1 > 0$.

Stability Analysis

Candidate Lyapunov Function

$$V_\theta(z_\theta) = \frac{1}{2}\tilde{x}^T\tilde{x} + \frac{1}{2}\tilde{f}^T\tilde{f} + \frac{1}{2}\tilde{\theta}^T\Gamma_\theta^{-1}(t)\tilde{\theta} + P$$

Theorem 1

The estimation errors are UUB such that $\|z_\theta\| \leq$

$$\sqrt{\frac{\lambda_2}{\lambda_1}\|z_\theta(0)\|^2 e^{-\frac{\lambda_3}{\lambda_2}t} + \frac{\lambda_2 C}{\lambda_1\lambda_3}\left(1 - e^{-\frac{\lambda_3}{\lambda_2}t}\right)}$$

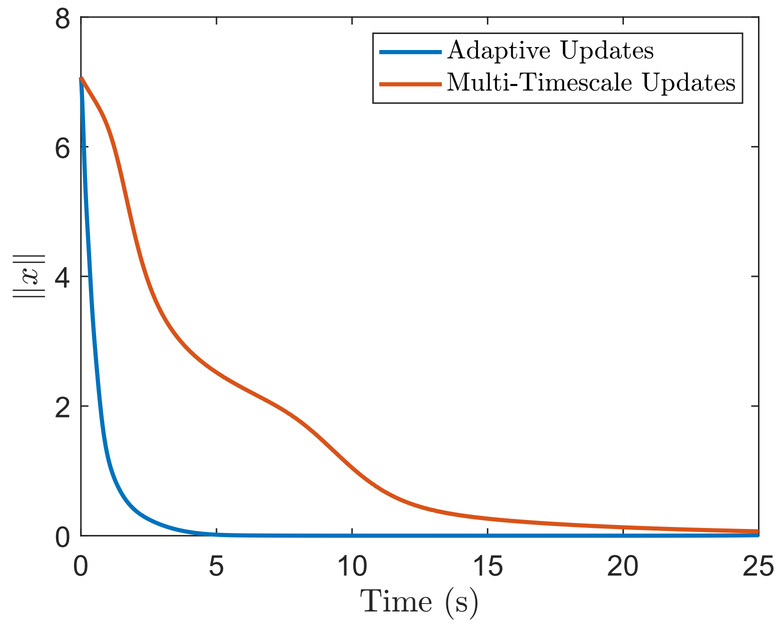
Candidate Lyapunov Function

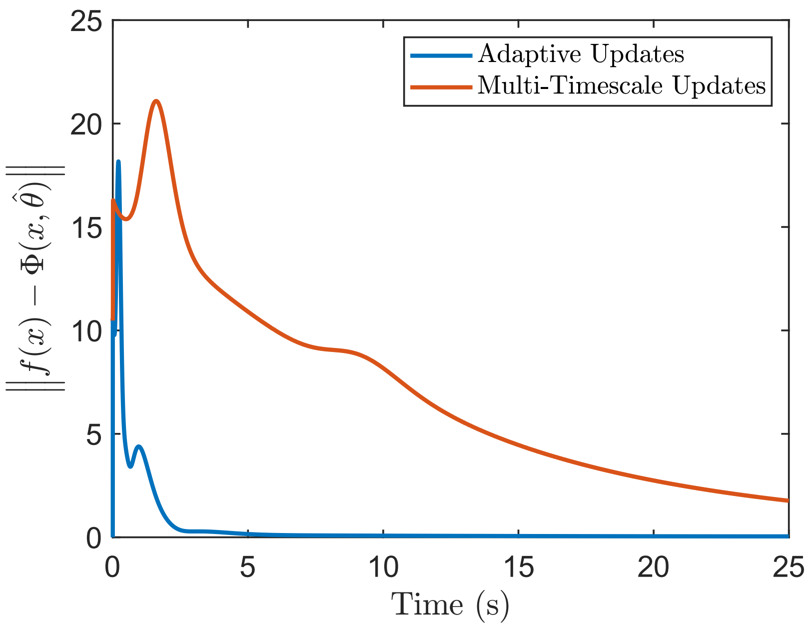
$$V_L(z,t) = V^*(x) + \frac{1}{2}\tilde{W}_c^T \Gamma^{-1}(t)\tilde{W}_c + \frac{1}{2}\tilde{W}_a^T \tilde{W}_a$$

Theorem 2

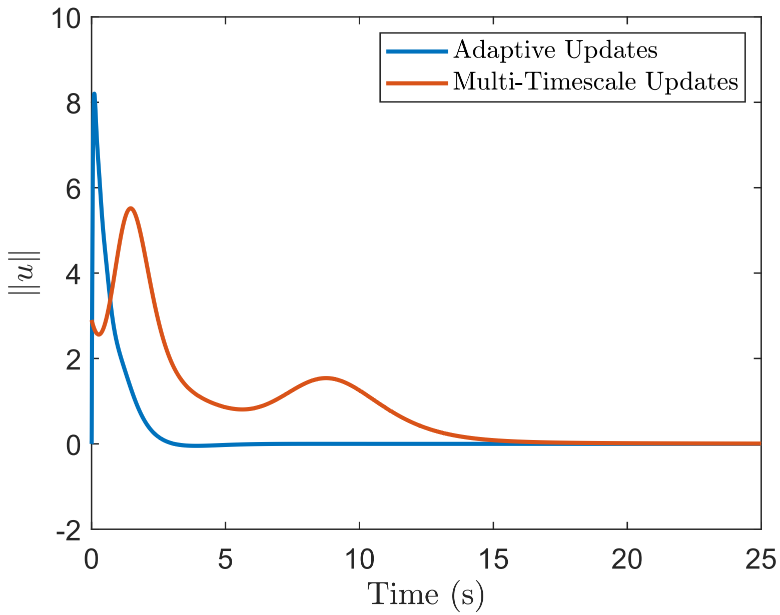
The state x, critic weight estimate error $\tilde{W}_c$, and actor weight estimate error $\tilde{W}_a$ are UUB. Hence, the control policy u converges to a neighborhood of the optimal control policy u^*.

Simulation Results





Simulation Results



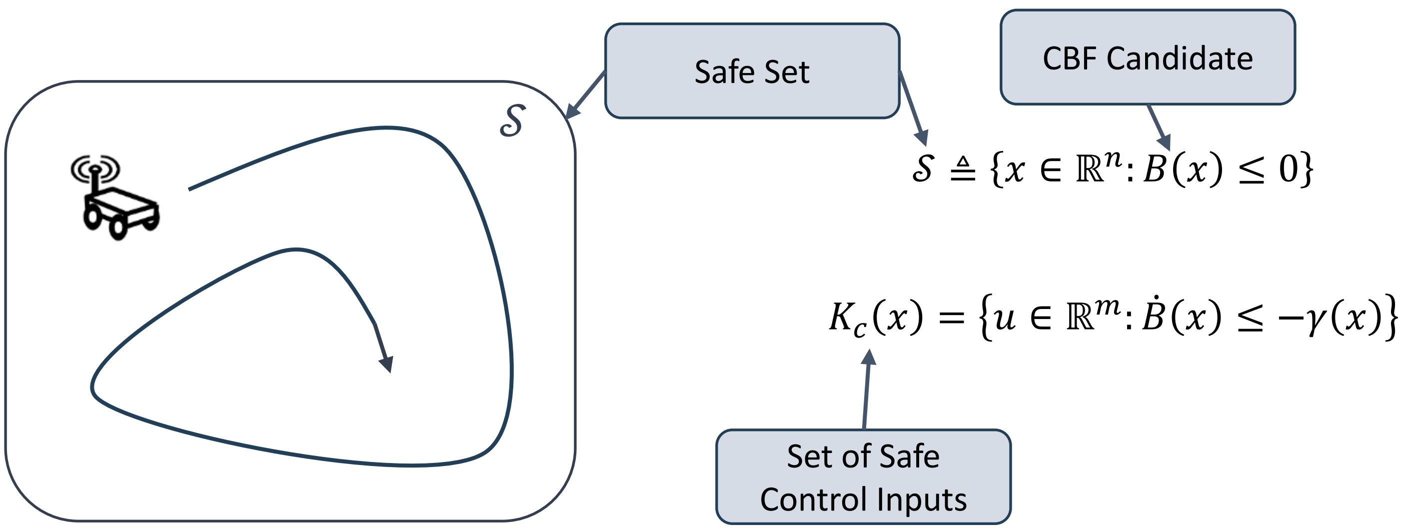
Controller	Multi-timescale	Adaptive	% Decrease
$\|x\|_{RMS}$	2.265	0.776	65.73
$\|u\|_{RMS}$	1.525	1.040	31.82
$\|f - \Phi(x, \hat{\theta})\|_{RMS}$	8.732	1.836	78.97

Adaptive Deep Neural Network-Based Control Barrier Functions

Hannah M. Sweatland, Omkar Sudhir Patil, and Warren E. Dixon, "Adaptive Deep Neural Network-Based Control Barrier Functions", *IEEE Control Systems Letters*, Under Review.

Control Barrier Functions (CBFs)

- One way of guaranteeing the safety of a system is through forward invariance
- Trajectories that start within some forward invariant safe set will never reach an unsafe region
- Control barrier functions (CBFs) convert state constraints into constraints on the control input

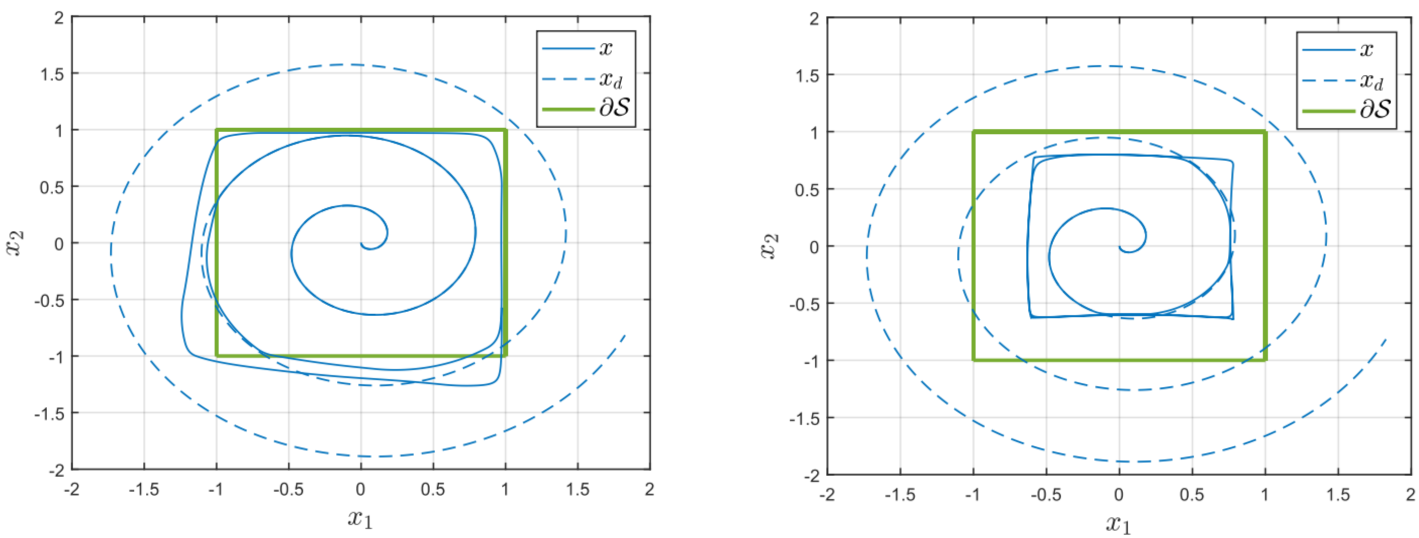


Control Barrier Functions

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

The function $f(x)$ is unknown but continuously differentiable

$$K_c = \{u \in \mathbb{R}^m : \nabla B^\top(x)(f(x) + g(x)u) \leq -\gamma(x)\}$$



A. Isaly, O. S. Patil, H. M. Sweatland, R. G. Sanfelice and W. E. Dixon, "Adaptive Safety with a RISE-Based Disturbance Observer," in *IEEE Transactions on Automatic Control*, 2024.

Adaptive Deep Neural Networks (DNNs)

- Motivation exists to estimate the unknown dynamics using a DNN with weights that update in real time

$$f(x) = \Phi(x, \theta^*) + \varepsilon$$

DNNs can approximate functions on a compact set $\Omega \supseteq \mathcal{S}$

- Recent works develop Lyapunov-based (Lb) weight adaptation laws for fully-connected DNNs, ResNets, LSTMs, PINNs, and Dropout DNNs, all of which are based on tracking error feedback

Jacobian $\Phi'^{\top}(x, \hat{\theta}) = \frac{\partial \hat{\Phi}}{\partial \hat{\theta}}$

$$\dot{\hat{\theta}} = \Gamma\left(-k_{\theta}\hat{\theta} + \Phi'^{\top}(x, \hat{\theta})e\right)$$

Tracking Error

- Because safety does not require tracking error convergence, weight adaptation laws should not be based on the tracking error

State-Derivative Observer

- A least squares weight adaptation law adaptively identifies the system dynamics based on an identification error
- Least squares-based real-time identification is challenging for continuous-time systems because it requires state-derivative information which is often unknown or noisy
- We develop a high-gain state-derivative estimator to quantify the identification error

$$\dot{\hat{x}} = \hat{f} + g(x)u + k_x\tilde{x},$$

$$\dot{\hat{f}} = k_f\left(\dot{\tilde{x}} + k_x\tilde{x}\right) + \tilde{x}$$

DNN Adaptation Law

- The DNN adaptation law is defined as

$$\dot{\hat{\theta}} = \mathrm{proj}\left(\Gamma\left(-k_\theta \hat{\theta} + \Phi'^{\top}(x,\hat{\theta})\left(\hat{f} - \Phi(x,\hat{\theta})\right)\right)\right)$$

The projection operator ensures $\hat{\theta}(t) \in \mathcal{B} \triangleq \{\theta \in \mathbb{R}^p : \|\theta - \theta^*\| \leq \Xi\}$

- The term $\Gamma \in \mathbb{R}^{p\times p}$ denotes a symmetric positive-definite time-varying least squares adaptation gain matrix that is a solution to

$$\frac{d}{dt}\Gamma^{-1} = -\beta(t)\Gamma^{-1} + \Phi'^{\top}(x,\hat{\theta})\Phi'(x,\hat{\theta}),$$

where the bounded-gain time-varying forgetting factor $\beta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is defined as

$$\beta(t) \triangleq \beta_0\left(1 - \frac{\lambda_{\max}\{\Gamma\}}{\kappa_0}\right) \geq \beta_1 \in \mathbb{R}_{\geq 0}$$

Stability Analysis

Theorem 1: The parameter estimation error is bounded such that

$$\left\|\tilde{\theta}(t)\right\| \leq \tilde{\theta}_{UB}(t) \triangleq \sqrt{\frac{\lambda_2}{\lambda_1}\|z(t_0)\|^2 e^{-\frac{\lambda_3}{\lambda_2}t} + \frac{\lambda_2 C}{\lambda_1}\left(1 - e^{-\frac{\lambda_3}{\lambda_2}t}\right)}$$

where $\lambda_1 \triangleq \min\left\{\frac{1}{2}, \frac{1}{2\kappa_0}\right\}$, $\lambda_2 \triangleq \min\left\{\frac{1}{2}, \frac{1}{2\kappa_1}\right\}$, $\lambda_3 \triangleq \min\left\{k_x, k_f - \frac{\bar{\dot{f}} + c_2}{2}, \frac{\beta_1}{2\kappa_0} + \frac{k_\theta}{2} - c_2\right\}$, and $C \triangleq \frac{\bar{\dot{f}} + c_2 c_1^2 + k_\theta \bar{\theta}^2}{2}$, provided $\lambda_3 > 0$.

$$z \triangleq \left[\tilde{x}^\top, \tilde{f}^\top, \tilde{\theta}^\top\right]^\top$$

- Because $\tilde{\theta}_{UB}(t)$ may initially be more conservative than Ξ, we define the auxiliary function χ_θ

$$\left\|\tilde{\theta}(t)\right\| \leq \chi_\theta \triangleq \min\left\{\Xi, \sqrt{\frac{\lambda_2}{\lambda_1}\left(\Xi^2 + 4\bar{f}^2\right)e^{-\frac{\lambda_3}{\lambda_2}t} + \frac{\lambda_2 C}{\lambda_1}\left(1 - e^{-\frac{\lambda_3}{\lambda_2}t}\right)}\right\}$$

$$\hat{\theta}(t_0) \in \mathcal{B},\ \hat{f}(t_0) \leq \bar{f}$$

Safe Control Inputs

- A new set of safe control inputs can be found that is composed of only known terms
- Begin with the original

$$K_c = \{u \in \mathbb{R}^m : \nabla B^\top(x)(f(x) + g(x)u) \leq -\gamma(x)\}$$

- Substitute in DNN estimate of $f(x)$, the Taylor series approximation of $\Phi(x, \theta^*)$, and χ_θ to yield

$$K_d(x) \triangleq \left\{u \in \mathbb{R}^m : \|\nabla B^\top(x)\Phi'\|(\chi_\theta + \bar{\Delta}) + \nabla B^\top(x)\left(\Phi\left(x, \hat{\theta}\right) + g(x)u\right) \leq -\gamma(x)\right\}$$

Definition 2: A continuously differentiable CBF candidate $B: \mathbb{R}^n \to \mathbb{R}^d$ defining the set $\mathcal{S} \subseteq \Omega$ is an *adaptive DNN CBF (aDCBF)* for the dynamic system and the safe set $\mathcal{S}$ on a set $\mathcal{O} \subset \mathbb{R}^n$ with respect to a function $\gamma: \mathbb{R}^n \to \mathbb{R}^d$ if there exists a neighborhood of the boundary of $\mathcal{S}$ such that $\mathcal{N}(\partial\mathcal{S}) \subset \mathcal{O}$, 2) for each $i \in [d]$, $\gamma_i \geq 0$ for all $x \in \mathcal{N}(M_i) \backslash \mathcal{S}_i$, and 3) the set

$$K_d(x) \triangleq \left\{u \in \mathbb{R}^m : \|\nabla B^\top(x)\Phi'\|(\chi_\theta + \bar{\Delta}) + \nabla B^\top(x)\left(\Phi\left(x, \hat{\theta}\right) + g(x)u\right) \leq -\gamma(x)\right\}$$

is nonempty for all $x \in \mathcal{O}$.

Forward Invariance

- An optimization-based control law $\kappa^*: \mathbb{R}^n \to \mathcal{U}$ is used to make a selection of K_d and is defined as

$$\kappa^*(x) \triangleq \mathrm{argmin}_{u \in \mathcal{U}} Q(x, u)$$

$$\text{s.t. } \|\nabla B^\top(x)\Phi'\|(\chi_\theta + c_1) + \nabla B^\top(x)\left(\widehat{\Phi}(x, \theta) + g(x)u\right) \leq -\gamma(x)$$

Main Result

Theorem 2: Suppose $B: \mathbb{R}^n \times \mathbb{R}^p$ is an aDCBF defining a safe set $\mathcal{S} \subseteq \Omega$ for the closed-loop system. Let $\hat{x}$, $\hat{f}$, and $\hat{\theta}$ update according to the developed state-derivative estimator and adaptive update law, respectively, and let $\hat{x}(t_0) = x(t_0)$, $\|\hat{f}\| \leq \bar{f}$, $z(t_0) \in \mathcal{D}$, and $\hat{\theta}(t_0) \in \mathcal{B}$. If κ^* is continuous, then the set $\mathcal{S}$ is forward invariant, provided $\lambda_3 > 0$.

Adaptive Cruise Control

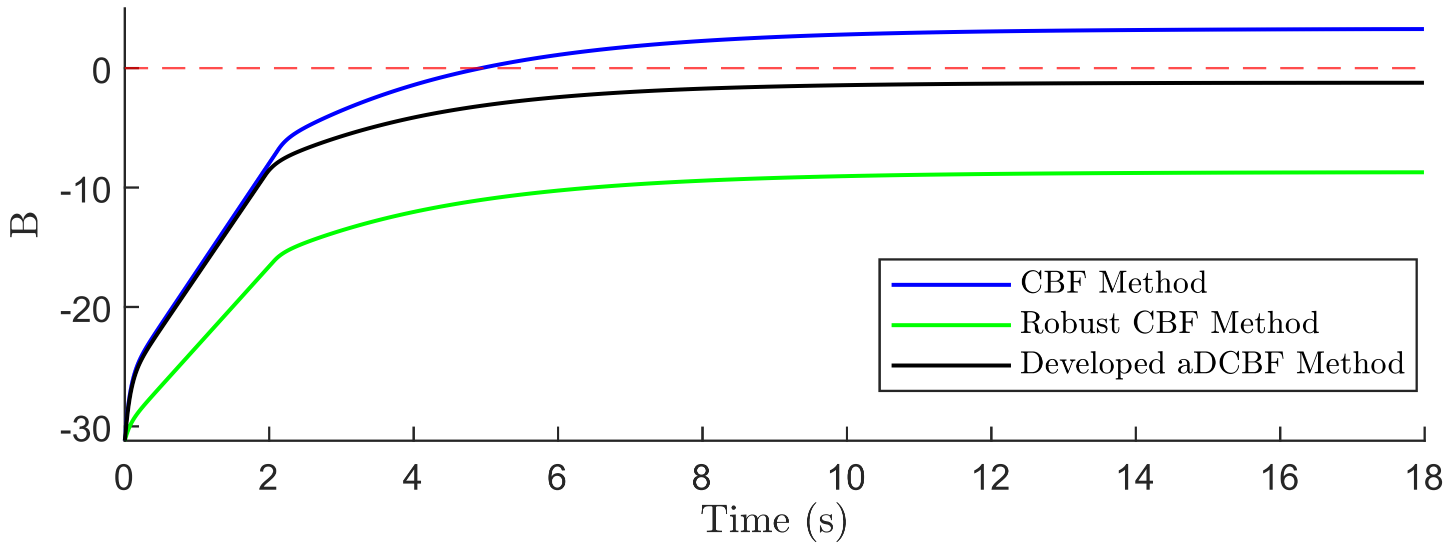
$$\dot{v} = -\frac{1}{m}F_r + \delta(v) + \frac{1}{m}u$$

- Deep ResNet with 2 hidden layers, a shortcut connection between each layer, and 6 neurons in each layer for a total of 122 weights
- Controller uses cost function

$$Q(x,u) = \|u - u_{nom}(x)\|^2$$

where

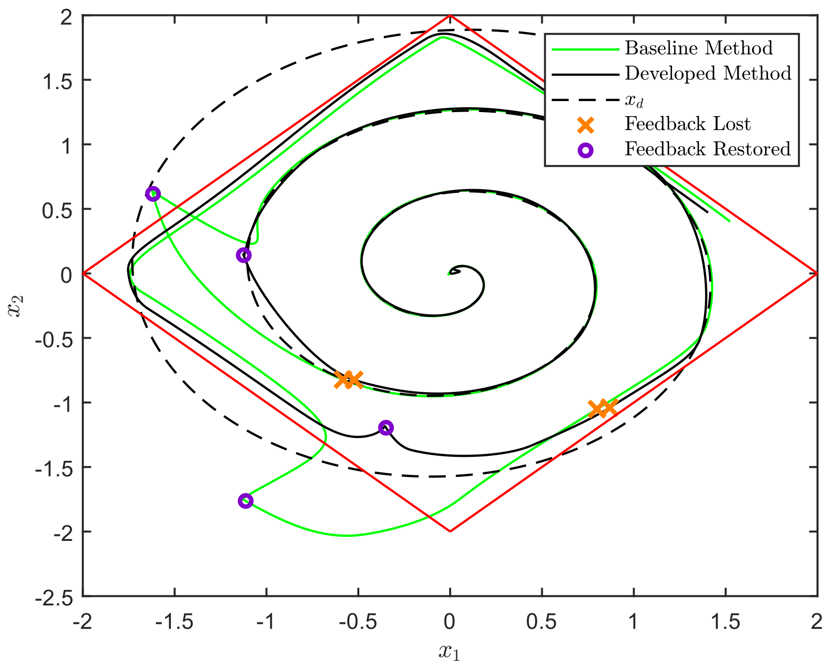
$$u_{nom} = -\Phi\left(v,\hat{\theta}\right) - mk_1(x - x_d)$$



Non-Polynomial Dynamics

$$f(x) = [x_2 \sin(x_1) \tanh^2(x_2), x_1 x_2 \cos(x_2) \operatorname{sech} x_2]^\top$$

- Deep ResNet with 3 hidden layers, a shortcut connection between each layer, and 5 neurons in each layer for a total of 174 weights
- Controller uses cost function
  $$Q(x,u) = \|u - u_{nom}(x)\|^2$$
  where
  $$u_{nom} = \dot{x}_d - \Phi(x, \hat{\theta}) - k_e(x - x_d)$$
- Baseline method uses
  $$u_{nom} = \dot{x}_d - \hat{f} - k_e(x - x_d)$$



Adaptive Output Feedback Control Using Lyapunov-Based Deep Recurrent Neural Networks (Lb-DRNNs)

Emily Griffis, Omkar Sudhir Patil, Wanjiku A. Makumi, and Warren E. Dixon, “Adaptive Output Feedback Control Using Lyapunov-Based Deep Recurrent Neural Networks (Lb-DRNNs)”, *IEEE Transactions on Automatic Control*, Under Review.

Contribution

- RNNs are a dynamic model → better suited for dynamical system identification and output feedback (OFB) control compared to feedforward NNs
- Previous deep RNN (DRNN)-based control results use offline optimization techniques to train the DRNN weights.
  - No online learning or adaptive control result for deep RNN architectures.
  - No OFB control result for DRNNs.
- Develop adaptive Lyapunov-based DRNN (Lb-DRNN) OFB controller.
  - A continuous-time Lb-DRNN is developed to adaptively estimate unknown system states in an observer design.
  - Lb-DRNN is implemented in controller to adaptive compensate for model uncertainties.
  - Stability-driven adaptation laws adjust the Lb-DRNN weights in real-time.

System Dynamics and Control Objective

Consider a second order nonlinear system

x_1 known
x_2 unknown

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = f(x) + g(x_1)u$$

Use DRNN to adaptively estimate system dynamics

Design estimation ($\tilde{x}_1$) and tracking (e_1) errors

$$\tilde{x}_1 \triangleq x_1 - \hat{x}_1 \qquad e_1 \triangleq x_1 - x_{d,1}$$
$$\xi \triangleq \dot{\tilde{x}}_1 + \alpha\tilde{x}_1 + \eta \qquad r \triangleq \dot{e}_1 + \alpha e_1 + \eta$$

Auxiliary errors ξ and r are unknown → can't be used in adaptation law design

Dynamic Filter

Auxiliary errors ξ and r are unknown →
Design dynamic filter to generate secondary errors that can be implemented in the adaptation law design

Dynamic filter:

$$\eta \triangleq p - (\alpha + k_r)\tilde{x}_1$$

$$\dot{p} \triangleq -(k_r + 2\alpha)p - \nu + \left((\alpha + k_r)^2 + 1\right)\tilde{x}_1 + \mathrm{e}_1$$

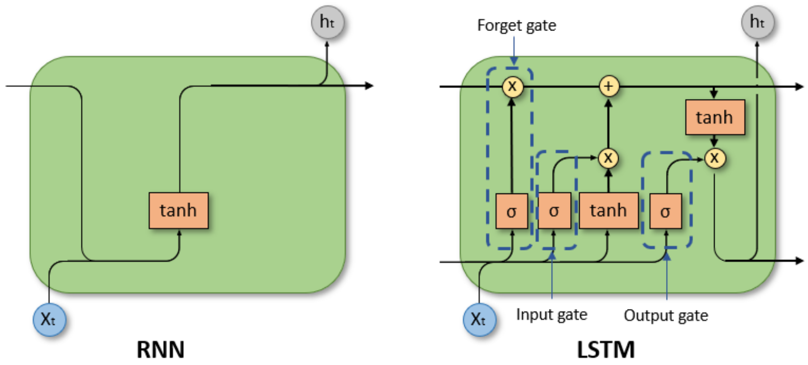
$$\dot{\nu} \triangleq p - \alpha\nu - (\alpha + k_r)\tilde{x}_1$$

Implementable Errors:

$$e_{es} = \tilde{x}_1 + \nu \rightarrow r = \dot{e}_{es} + \alpha e_{es}$$

$$e_{tr} = e_1 + \nu \rightarrow \xi = \dot{e}_{tr} + \alpha e_{tr}$$

Use deep RNN to estimate the unknown state



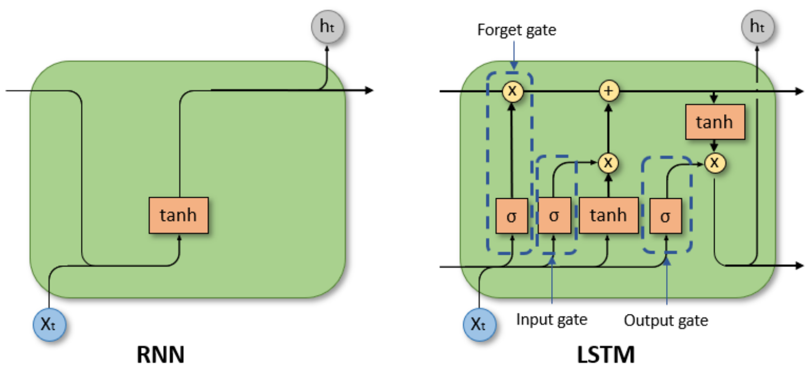
$$\dot{h} = -bh + W_1^\top \phi_k \circ W_0^\top y$$

ϕ: tanh activation function

$y = [h^\top, x^\top, 1]^\top$: concatenated input

OFB Control Design

Use deep RNN to estimate the unknown state



Make it deep!

$$\dot{h} = -bh + W_k^\top \phi_k \circ \cdots \circ W_1^\top \phi_1 \circ W_0^\top y$$

ϕ: tanh activation function

$y = [h^\top, x^\top, 1]^\top$: concatenated input

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Observer Development

Use deep RNN to estimate the unknown state

$$\dot{h} = -bh + W_k^{\top}\phi_k \circ \cdots \circ W_1^{\top}\phi_1 \circ W_0^{\top} y$$

$\dot{h} = -bh + \Phi(h, \theta) \rightarrow \theta$: DRNN weights

$$\dot{x}_2 = -bx_2 + \Phi(x, \theta) + g(x_1)u + \varepsilon(x)$$

$\rightarrow \varepsilon$: residual error

Observer Design

Using the adaptive DRNN term $-b\hat{x}_2 + \widehat{\Phi}(\hat{x}, \hat{\theta})$, the observer is designed as

$$\dot{\hat{x}}_1 \triangleq \hat{x}_2$$
$$\dot{\hat{x}}_2 \triangleq -b\hat{x}_2 + \widehat{\Phi}(\hat{x}, \hat{\theta}) + g(x_1)u + \beta_1 \mathrm{sgn}(e_{es}) + \chi$$

$\hat{\theta} = [vec(W_0)^{\top} \ldots vec(W_k)^{\top}]^{\top}$ → stacked representation of all weight estimates

Generates state estimate $\hat{x}_2$ to use in controller

The controller is designed as

$$u \triangleq g(x_1)^+\left[-\left(-bx_2 + \widehat{\Phi}\left(\hat{x}, \hat{\theta}\right)\right) - \beta_2 \operatorname{sgn}(e_{tr}) + \ddot{x}_{d,1} - (k_r + \alpha)\left(\dot{\hat{e}}_1 + \alpha \hat{e}_1\right) - \alpha^2 e_1 - \nu\right]$$

where $\hat{e}_1 \triangleq \hat{x}_1 - x_{d,1}$

How do we design weight estimates $\hat{\theta}$?

Weight Adaptation Law

The weight adaptation law is designed as

$$\dot{\hat{\theta}} \triangleq \Gamma \widehat{\Phi}'^{\top}(e_{es} + e_{tr})$$

Adaptation gain matrix

Jacobian $\widehat{\Phi}' = \frac{\partial \widehat{\Phi}}{\partial \hat{\theta}}$

Estimation and Tracking errors

Implementable errors (using dynamic filter):

$$e_{es} \triangleq \tilde{x}_1 + \nu$$
$$e_{tr} \triangleq e_1 + \nu$$

Main Stability Result

Theorem 1. The adaptive DRNN OFB controller and weight adaptation laws ensure asymptotic state estimation error and tracking error convergence in the sense that

$$\|x_2 - \hat{x}_2\| \to 0 \text{ as } t \to \infty$$
$$\|x_1 - x_{d,1}\| \to 0 \text{ as } t \to \infty$$

Simulation Parameters

- Comparative simulations were performed on a 6DOF unmanned underwater vehicle (UUV) system
  - DRNN OFB controller: 8 (tanh) layers with 8 neurons
  - Shallow RNN (SRNN) OFB controller: 2 (tanh) layers with 17 neurons
  - Central difference observer (no DNN in controller)
- Noise on position from uniform distribution *U(-0.001, 0.001)*
- 150 seconds with a step size of 0.001 seconds with initial condition

$$x_1 = \left[4\ [m], 0.5\ [m], 0[m], 0\ [rad], 0.2\ [rad], 0[rad]\right]^\top$$

- Helical Desired trajectory

$$x_{1,d} = \left[5\cos(0.1t)\ \ [m], 5\sin(0.1t)\ \ [m], 0.1t\ [m], 0\ [rad], 0\ [rad], -0.05t\ [rad]\right]^\top$$

Control Gains

$b = 1$

$k_r = 2$

$\alpha = 5$

$\beta_1 = \beta_2 = 0.001$

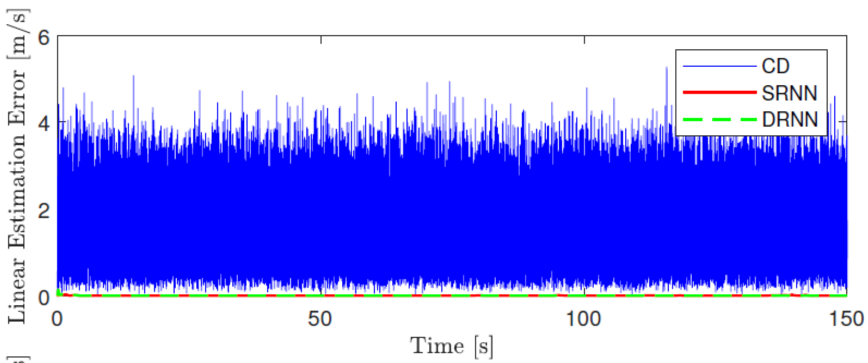
$\Gamma = 0.5 \cdot I$

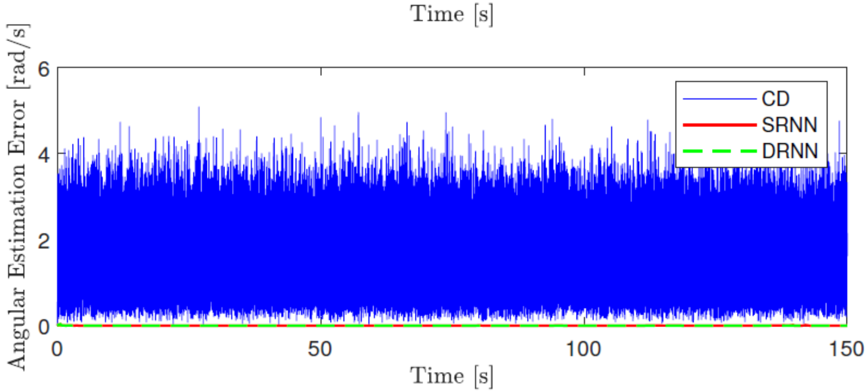
For a fair comparison, the same gains were used

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

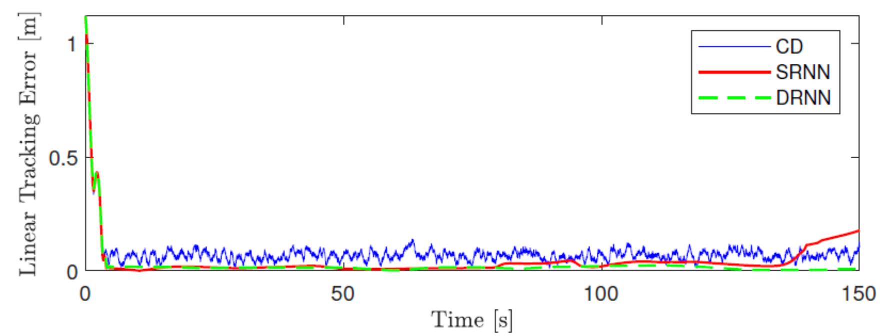
Simulation Results

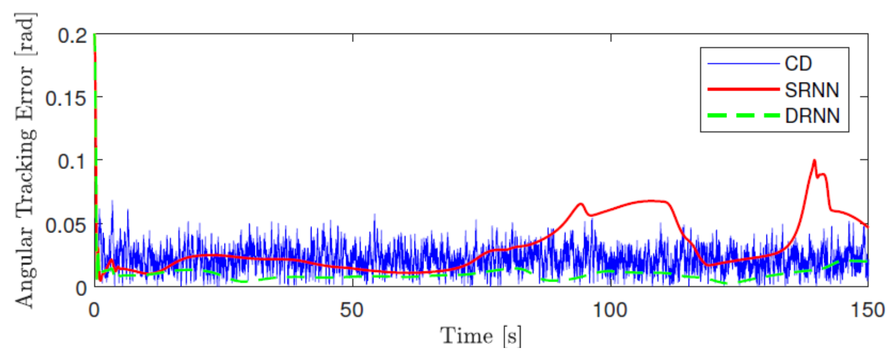
Velocity Estimation Error





Position Tracking Error





Simulation Results

Architecture		$\|e_1\|$	$\|\tilde{x}_2\|$	$\|\text{control input}\|$
SRNN	Linear	0.1215 [m]	0.0103 [m/s]	92.19 [N]
	Angular	0.0815 [rad]	0.0081 [rad/s]	12.72 [Nm]
CD	Linear	0.1115 [m]	1.7339 [m/s]	301.37 [N]
	Angular	0.0252 [rad]	1.7387 [rad/s]	336.43 [Nm]
DRNN	Linear	0.0875 [m]	0.0049 [m/s]	91.84 [N]
	Angular	0.0082 [rad]	0.0027 [rad/s]	12.68 [Nm]

CD sensitive to measurement noise – High estimation error and control effort!

Simulation Results

Architecture		$\|e_1\|$	$\|\tilde{x}_2\|$	$\|$control input$\|$
SRNN	Linear	0.1215 [m]	0.0103 [m/s]	92.19 [N]
	Angular	0.0815 [rad]	0.0081 [rad/s]	12.72 [Nm]
CD	Linear	0.1115 [m]	1.7339 [m/s]	301.37 [N]
	Angular	0.0252 [rad]	1.7387 [rad/s]	336.43 [Nm]
DRNN	Linear	0.0875 [m]	0.0049 [m/s]	91.84 [N]
	Angular	0.0082 [rad]	0.0027 [rad/s]	12.68 [Nm]

27.98% improvement in linear tracking error
89.94% improvement in angular tracking error

Simulation Results

Architecture		$\|e_1\|$	$\|\tilde{x}_2\|$	$\|$control input$\|$
SRNN	Linear	0.1215 [m]	0.0103 [m/s]	92.19 [N]
	Angular	0.0815 [rad]	0.0081 [rad/s]	12.72 [Nm]
CD	Linear	0.1115 [m]	1.7339 [m/s]	301.37 [N]
	Angular	0.0252 [rad]	1.7387 [rad/s]	336.43 [Nm]
DRNN	Linear	0.0875 [m]	0.0049 [m/s]	91.84 [N]
	Angular	0.0082 [rad]	0.0027 [rad/s]	12.68 [Nm]

27.98% improvement in linear tracking error
89.94% improvement in angular tracking error

52.43% improvement in linear estimation error
66.67% improvement in angular estimation error

With comparable control effort

Thank you

UF UNIVERSITY of FLORIDA

Duke UNIVERSITY

TEXAS The University of Texas at Austin

UC SANTA CRUZ