Recent Advances in Deep Learning

Second-Order Heterogeneous Multi-Agent Target Tracking without Relative Velocities

Cristian F. Nino, Omkar Sudhir Patil, and Warren E. Dixon

System Dynamics and Objective

- Consider a network composed of N agents, each with dynamics:

$$\ddot{q}_i = f_i(q_i, \dot{q}_i) + u_i$$

- Consider a target with model:

$$\ddot{q}_0 = g(q_0, \dot{q}_0)$$

- Each agent is capable of *only* measuring relative positions:

$$d_{ij} \triangleq q_j - q_i \qquad \text{and} \qquad e_i \triangleq q_0 - q_i$$

- Objective is to regulate all agents to the target, i.e.

$$\lim_{t \to \infty} \|e_i\| = 0$$

- Define the relative position error

$$\eta_i \triangleq \sum_{j \in \mathcal{N}_i} d_{ij} + b_i e_i \Rightarrow \eta = \mathcal{H} e$$

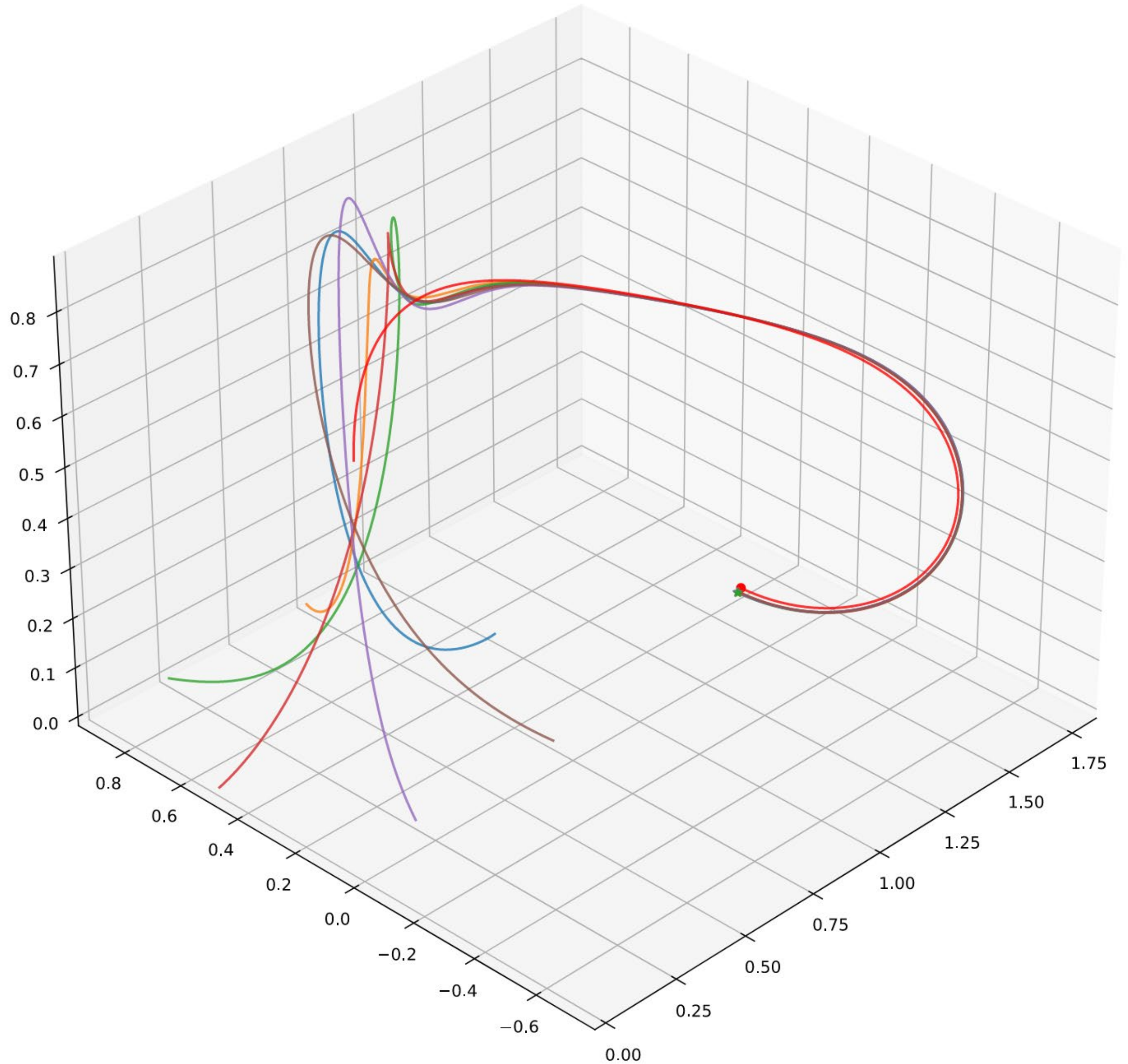
- Define the relative velocity error

$$\zeta_i \triangleq \dot{\eta}_i \Rightarrow \zeta = \mathcal{H} \dot{e}$$

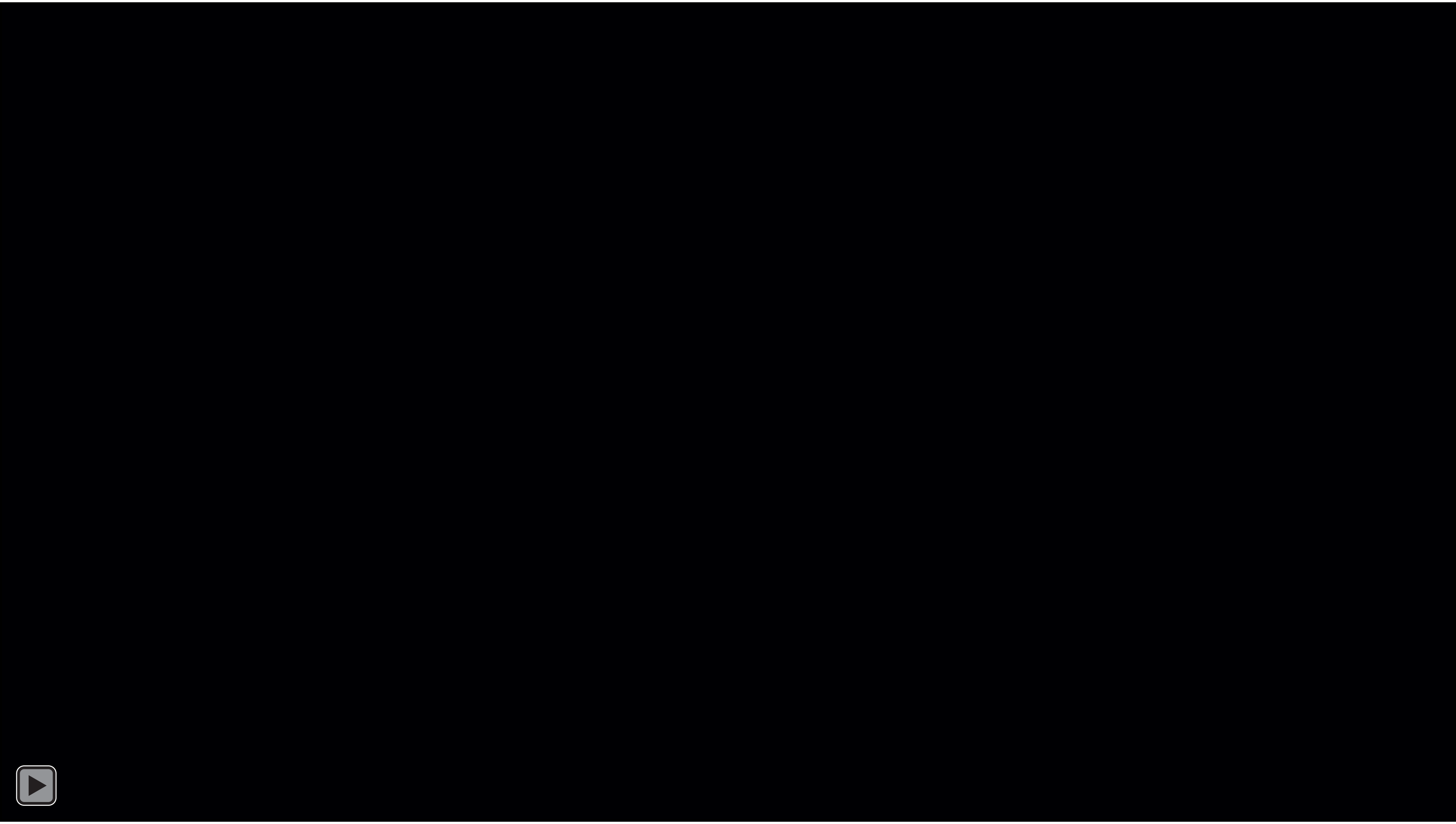
- $\mathcal{H}$ is a matrix which encodes the structure of the network

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

Simulation Results



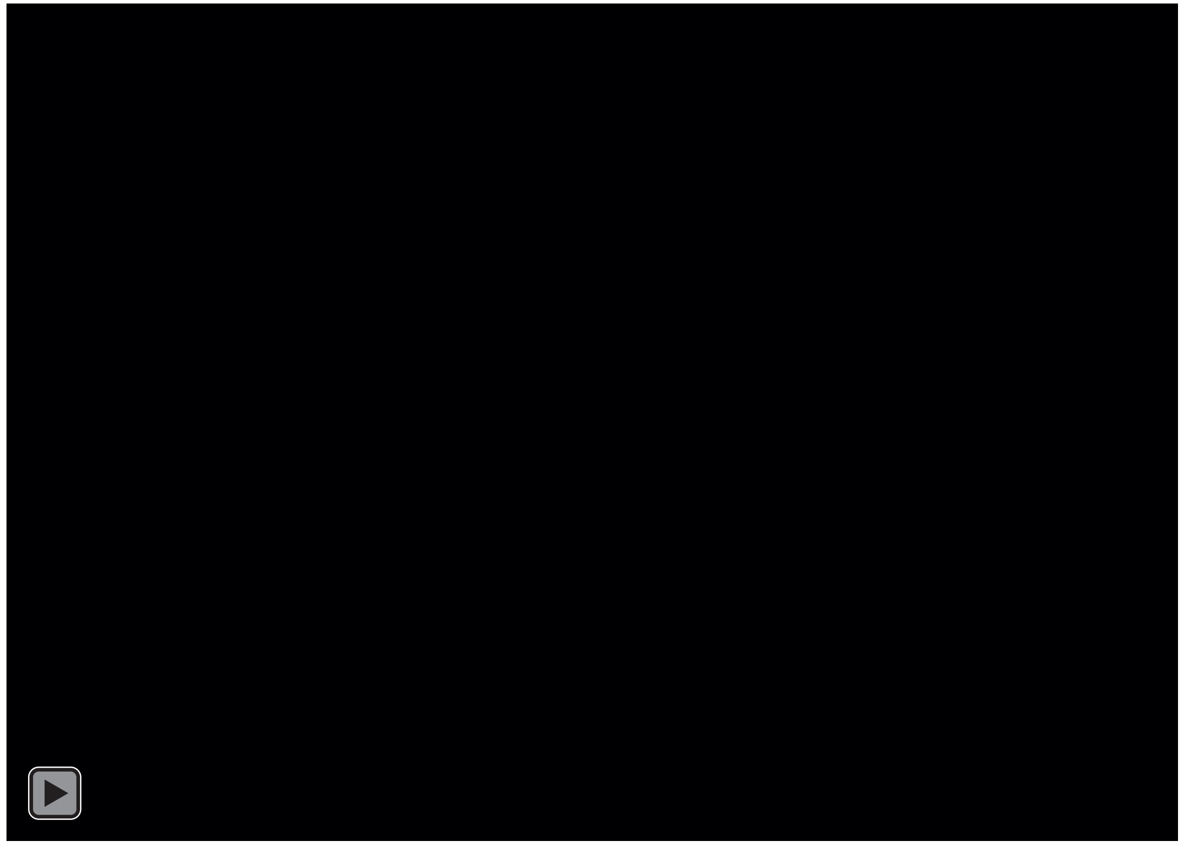
Why Deep Learing?



Approximate Optimal Indirect Herding with a Lyapunov-Based Deep Neural Network

Wanjiku A. Makumi, Jhyv N. Philor, Zachary I. Bell, and Warren E. Dixon

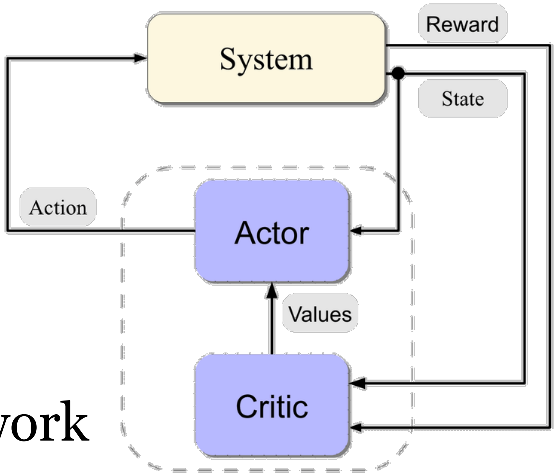
Simulations



Pursuing agent
Evading agent
Goal location

Approximate Dynamic Programming

- How do we optimally facilitate autonomous herding of an unknown while also providing real-time adaptation?
- Approximate dynamic programming (ADP)
  - Optimal control & adaptive control
- Hamilton-Jacobi-Bellman 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
- Integral Concurrent Learning (ICL)-based Deep Neural Network (DNN)
  - Unknown interaction dynamics between agents must be learned in real-time



Approximate Optimal Control

Control objective: Design a controller μ which minimizes the cost function

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

Optimal value function (cost-to-go)

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

Optimal control policy

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

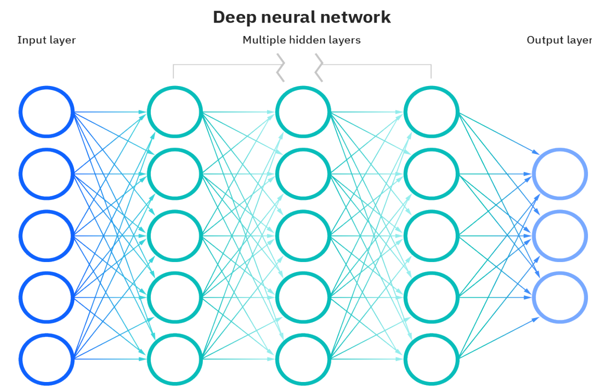
Hamilton-Jacobi-Bellman equation

$$0 = \nabla V^*(x)\big(F(x,\theta) + G(x)\mu^*(x)\big) + Q(x) + P(x) + \mu^{*T} R\mu^*$$

Replace optimal values in dynamics and HJB equation with estimates

$$\theta, V^*, \nabla V^*, \mu^* \longrightarrow \hat{\theta}, \hat{V}, \nabla\hat{V}, \hat{\mu}$$

Multi-timescale ICL-Lb-DNN



DNN agent dynamics representation

$$\dot{\breve{x}}(t) = \phi\big(\Phi(x)\big)\theta + \breve{G}\big(x(t), u(t)\big) + \varepsilon\big(x(t)\big)$$

$$\hat{\dot{\breve{x}}}_i(t) = \phi\left(\hat{\Phi}_i(x)\right)\hat{\theta} + \breve{G}\big(x(t), u(t)\big)$$

Output-layer weight updates

$$\dot{\hat{\theta}} = \Gamma_\theta \phi\left(\hat{\Phi}_i(x_j)\right)\tilde{x}^T + k_\theta \Gamma_\theta \sum_{j=1}^{M} \phi\left(\hat{\Phi}_i(x_j)\right)\left(\dot{x}_j - g_j(x_j)u_j\right) - \hat{\theta}^T \phi\left(\hat{\Phi}_i(x_j)\right)$$

- Online
- Real-time
- Adaptive
- ICL-based update law

Inner-layer feature updates

$$\mathcal{L}_{i+1}(t) = \frac{1}{M}\sum_{j=1}^{M} \left\| \dot{x}_j - g_j(x_j)u_j - \hat{\theta}^T \phi\left(\hat{\Phi}_i(x_j)\right) \right\|^2$$

- Concurrent to real-time
- Batch updates
- Optimization
- Loss function

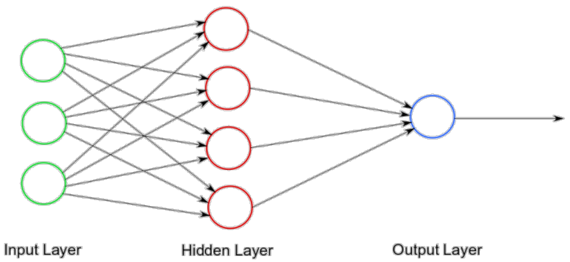
Actor-Critic Neural Networks

NN Optimal Value Function and NN Optimal Control Policy

$$V^*(x) = \boldsymbol{W}^{T}\sigma(x) + \varepsilon(x)$$

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

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

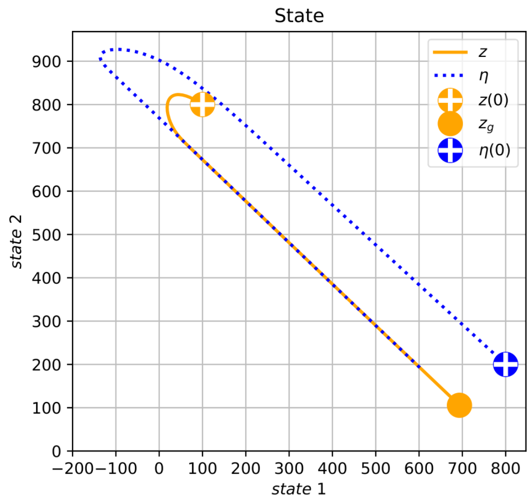


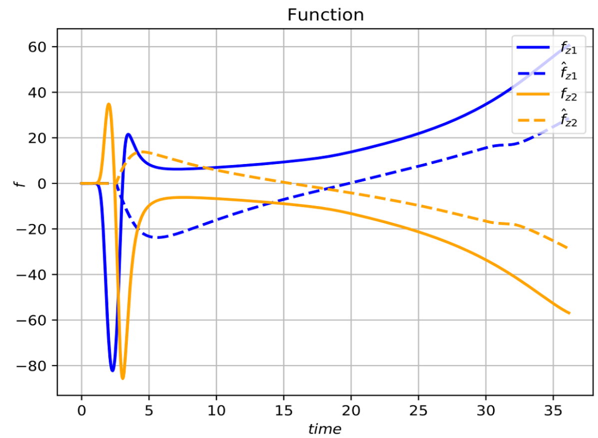
Optimal Value Function and Optimal Control Policy Approximation

$$\hat{V}\left(x, \widehat{W}_c\right) = \widehat{\boldsymbol{W}}_{\boldsymbol{c}}^{\boldsymbol{T}}\sigma(x)$$

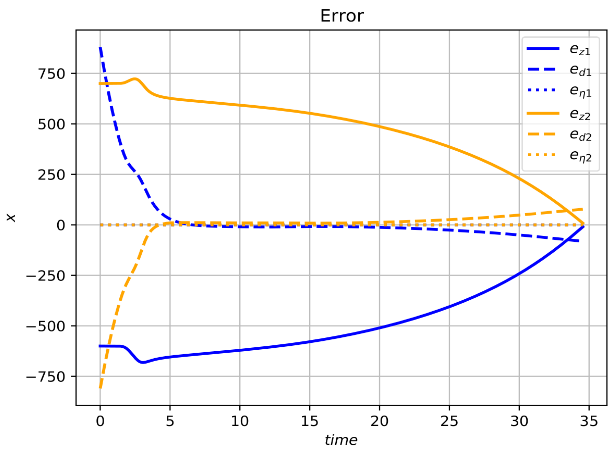
$$\hat{u}\left(x, \widehat{W}_a\right) = -\frac{1}{2}R^{-1}g(x)^T\left(\nabla_x\sigma(x)^T\widehat{\boldsymbol{W}}_{\boldsymbol{a}}\right)$$

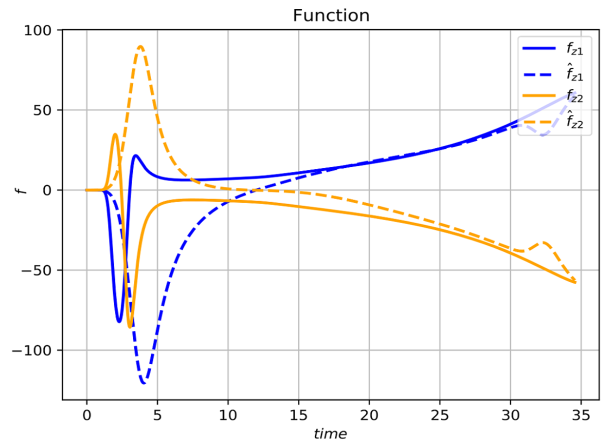
Simulation Results





Single-layer neural network





Deep neural network

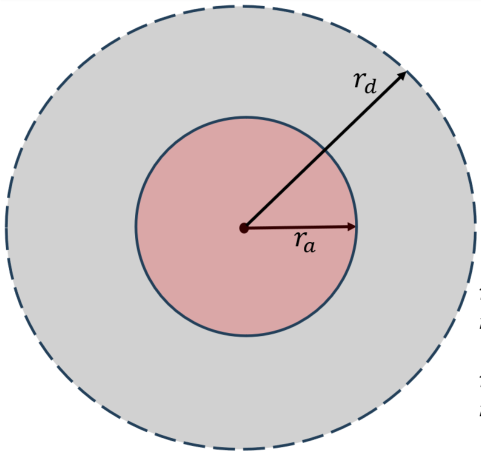
UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Obstacle Avoidance Extension

- Avoidance region dynamics

$$\dot{z}_i = s_i(H, \eta, z_i) b_i(z_i)$$

- $\boldsymbol{s_i}$- *scheduling function*
- $\boldsymbol{H}$ - *herding agent state*
- $\boldsymbol{\eta}$ - *evading agent state*
- $\boldsymbol{b_i}$ - *drift dynamics*



Herder dynamics

$$\dot{H} = f(H, \eta) g(H) u$$

Evader dynamics

$$\dot{\eta} = w(H, \eta) + \sum_{i=1}^{M} a_i(z_i, \eta)$$

$$\zeta = \left[x^T, z_1^T \cdots, z_M^T \right]^T$$

$$F(\zeta) = \begin{bmatrix} Existing\ F \\ s_1^* b_1(z_1) \\ \vdots \\ s_M^* b_M(z_M) \end{bmatrix}$$

$$G(\zeta) = \begin{bmatrix} Existing\ G \\ 0_{Mn \times m_H} \quad 0_{Mn \times n} \end{bmatrix}^T$$

Additions from the no-obstacle problem

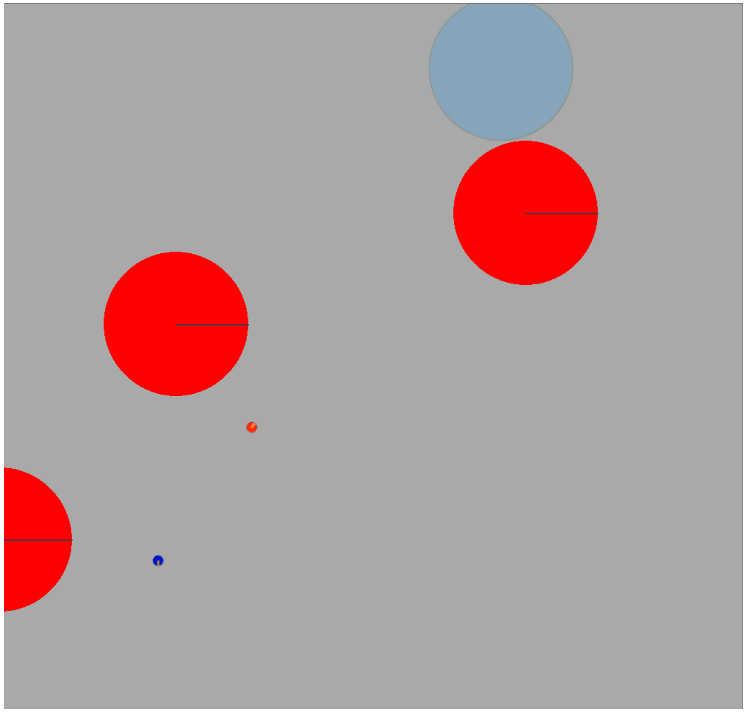
P. Deptula, H.-Y. Chen, R. Licitra, J. Rosenfeld, and W. E. Dixon, "Approximate Optimal Motion Planning to Avoid Unknown Moving Avoidance Regions," *IEEE Transactions on Robotics*, Vol. 36, No. 2, pp. 414-430 (2020).

Obstacle Avoidance Cost Function

- Cost function

$$J(\zeta,\mu)=\int_0^\infty\left(\sum_{i=1}^{M}s_iQ_z\big(z_i(\tau)\big)+Q_x\big(x(\tau)\big)+\Psi\big(\mu(\tau)\big)+\sum_{i=1}^{M}P_a(H,z_i)\right)d\tau$$

- $\boldsymbol{Q_x}$ *- penalty on the error states*
- $\boldsymbol{Q_z}$ *- penalty on sensed obstacles*
- $\boldsymbol{\Psi}$ *- penalty on control inputs*
- $\boldsymbol{P_a}$ *- penalty on avoidance regions*



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

Lyapunov-Based Dropout Deep Neural Network Controller

Saiedeh Akbari, Emily J. Griffis, Omkar Sudhir Patil, and Warren E. Dixon

Challenges of DNNs

Overfitting

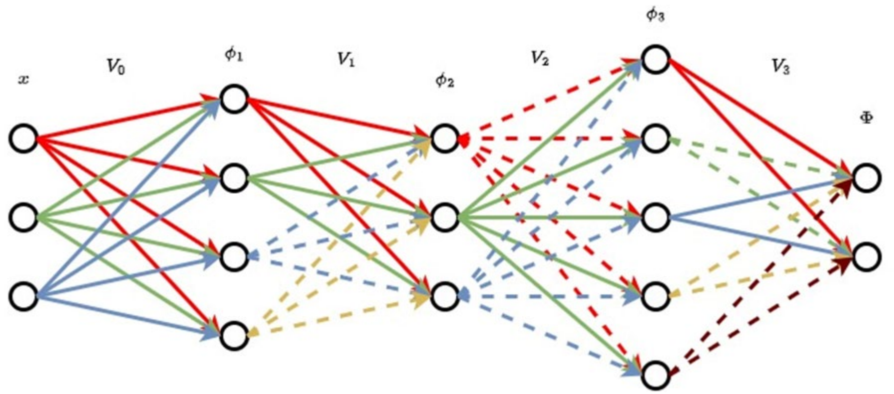
- Significantly degraded performance

Co-Adaptation

- Multiple neurons/layers become overly reliant on each other
- Decrease in generalization and DNN performance

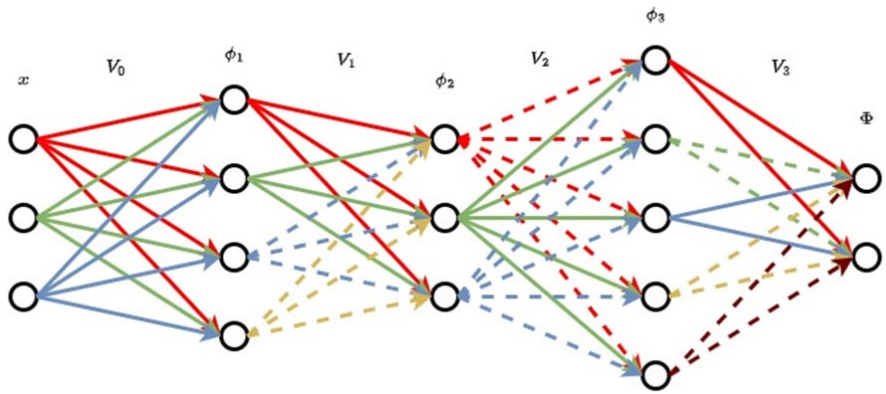
Dropout DNN (DDNN) Architecture

- Stochastically dropping out neurons during training
- Setting the activation of certain individual weights to zero
  - Induces sparse representation in the network
  - Reduces co-dependency in neurons
- Can be viewed as training of ensembles of multiple DNNs with similar width that are trained independently



Dropout Deep Neural Network

Dropout DNN (DDNN) Architecture



$$\Phi\left(x, R_i, \theta\right) = \left(R_{i,k} V_k\right)^\top \phi_k \circ \cdots \circ \left(R_{i,1} V_1\right)^\top \phi_1 \circ \left(R_{i,0} V_0\right)^\top x$$

Randomization matrices

$R_{i,j}$ is a diagonal matrix

A user-selected number of ones on the diagonal

The placement of ones on the diagonal changes every δt seconds

Dropout Deep Neural Network

$$u(t) \triangleq \dot{x}_d - \widehat{\Phi} - k_e e - k_s \operatorname{sgn}(e)$$

Lb-DDNN Weight Adaptation Law

$$\dot{\hat{\theta}} \triangleq \operatorname{proj}\left(\Gamma_\theta \widehat{\Phi}'^\top e\right)$$

$$\widehat{\Phi}' \triangleq \frac{\partial \Phi\left(x, R_i, \hat{\theta}\right)}{\partial \hat{\theta}}$$

$$\widehat{\Phi}'_0 \triangleq \left(\prod_{l=1}^{\overset{\curvearrowleft}{k}} \left(R_{i,l}\widehat{V}_l\right)^\top \widehat{\phi}'_l\right)\left(\left(x^\top R_{i,0}\right) \otimes I_{L_1}\right)$$

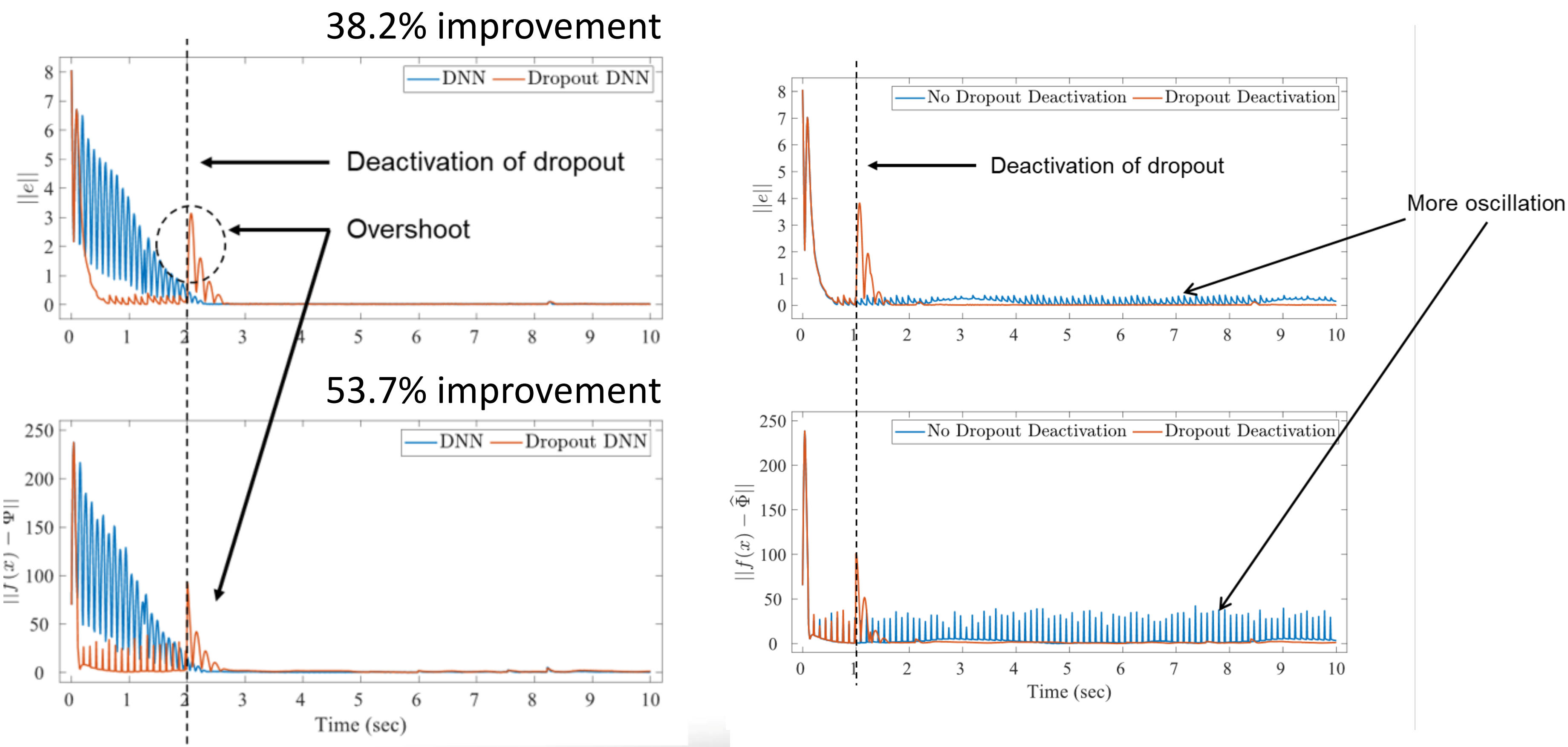
$$\widehat{\Phi}'_j \triangleq \left(\prod_{l=j+1}^{\overset{\curvearrowleft}{k}} \left(R_{i,l}\widehat{V}_l\right)^\top \widehat{\phi}'_l\right)\left(\left(\widehat{\phi}_j^\top R_{i,j}\right) \otimes I_{L_{j+1}}\right)$$

Dropout Deep Neural Network

- Three-dimensional nonlinear system:

$$f = \begin{bmatrix} x_1 x_2^2 \tanh(x_2) + \sin(x_1)^2 \\ \cos(x_1 + x_2 + x_3)^3 - \exp(x_2)^2 + x_1 x_2 \\ x_3^2 \log(1 + \mathrm{abs}(x_1 - x_2)) \end{bmatrix}$$

- Desired trajectory: $x_d(t) = [\sin(2t), -\cos(t), \sin(3t) + \cos(-2t)]^\top$



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

Lyapunov-Based Long-Short Term Memory (Lb-LSTM) Neural Network-Based Adaptive Observer

Emily J. Griffis, Omkar Sudhir Patil, Rebecca G. Hart, and Warren E. Dixon

System Dynamics and Objective

- Consider a second order nonlinear system

x_1 known
x_2 unknown

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = g(x,u)$$

- Design estimation error

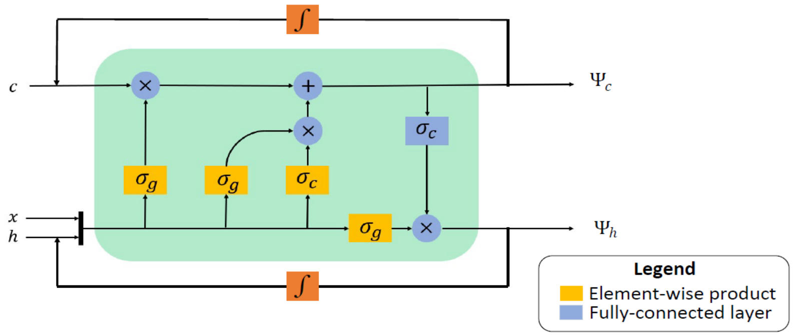
$$\tilde{x}_1 \triangleq x_1 - \hat{x}_1$$
$$r \triangleq \dot{\tilde{x}}_1 + \alpha\tilde{x}_1 + \eta$$

Use LSTM to adaptively estimate system dynamics

- Dynamic filter

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

LSTM Model



Gate Outputs	Cell State and Hidden State Dynamics

$$f(z, W_f) = \sigma_g \circ W_z^\top z$$

$$o(z, W_o) = \sigma_g \circ W_o^\top z$$

$$i(z, W_i) = \sigma_g \circ W_i^\top z$$

$$c^*(z, W_c) = \sigma_c \circ W_c^\top z$$

$z \triangleq [x^\top h^\top]^\top$ for some input x

$$\dot{c} = -b_c c + b_c \Psi_c(x, c, h, \theta)$$

$$\dot{h} = -b_h h + b_h \Psi_h(x, c, h, \theta, W_o)$$

$$\Psi_c(x, c, h, \theta) = f(z, W_f) \odot c + i(z, W_i) \odot c^*(z, W_c)$$

$$\Psi_h(x, c, h, \theta, W_o) = o(z, W_o) \odot (\sigma_c \circ \Psi_c(x, c, h, \theta))$$

Contribution

- Previous LSTM models use offline optimization techniques to train the LSTM weights.
  - No online learning of the LSTM
- An adaptive Lyapunov-based LSTM (Lb-LSTM) observer is developed to estimate unknown system states
  - A continuous-time Lb-LSTM NN is formulated
  - Stability-driven adaptation laws adjust the LSTM in real-time.

Architecture	$\|x_2 - \hat{x}_2\|$ **[deg/s]**	$\|\tilde{x}_1\|$ **[deg]**
RNN	0.3856	0.1065
LSTM	0.2270	0.0595
Percent Improvement	41.13%	44.09%

Lyapunov-Based Physics-Informed Long-Short Term Memory (LSTM) Neural Network-Based Adaptive Control

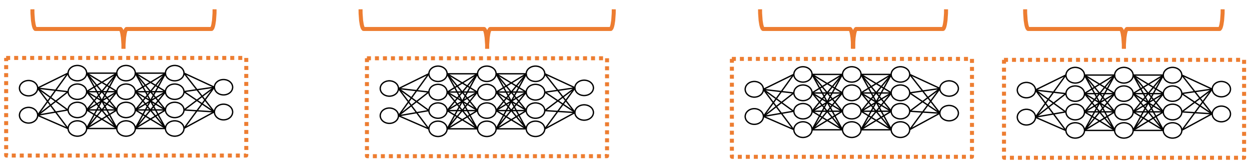
Rebecca G. Hart, Emily J. Griffis, Omkar Sudhir Patil, and Warren E. Dixon

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

Physics-Inspired Architecture

Physics-Inspired Motivation: To impose constraints derived from known physical laws on the learning algorithm to reduce the possible solution space and eliminate invalid solutions resulting from noisy data

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d(t) = \tau(t)$$



Physics-Inspired LSTM

Physics-Inspired LSTM Motivation: The presence of history or time-dependent dynamics in complex systems motivates the desire to capture these dependencies using a combination of a DNN and LSTM based approach

Example Systems: Smart Materials and systems that experience fluid structure interaction or electromagnetic effects.

$$\underbrace{M(q)\ddot{q}}_{\chi_M} + \underbrace{V_m(q,\dot{q})\dot{q}}_{\chi_V} + \underbrace{G(q)}_{\chi_G} + \underbrace{F(\dot{q})}_{\chi_F} + \tau_d(t) = \tau(t)$$

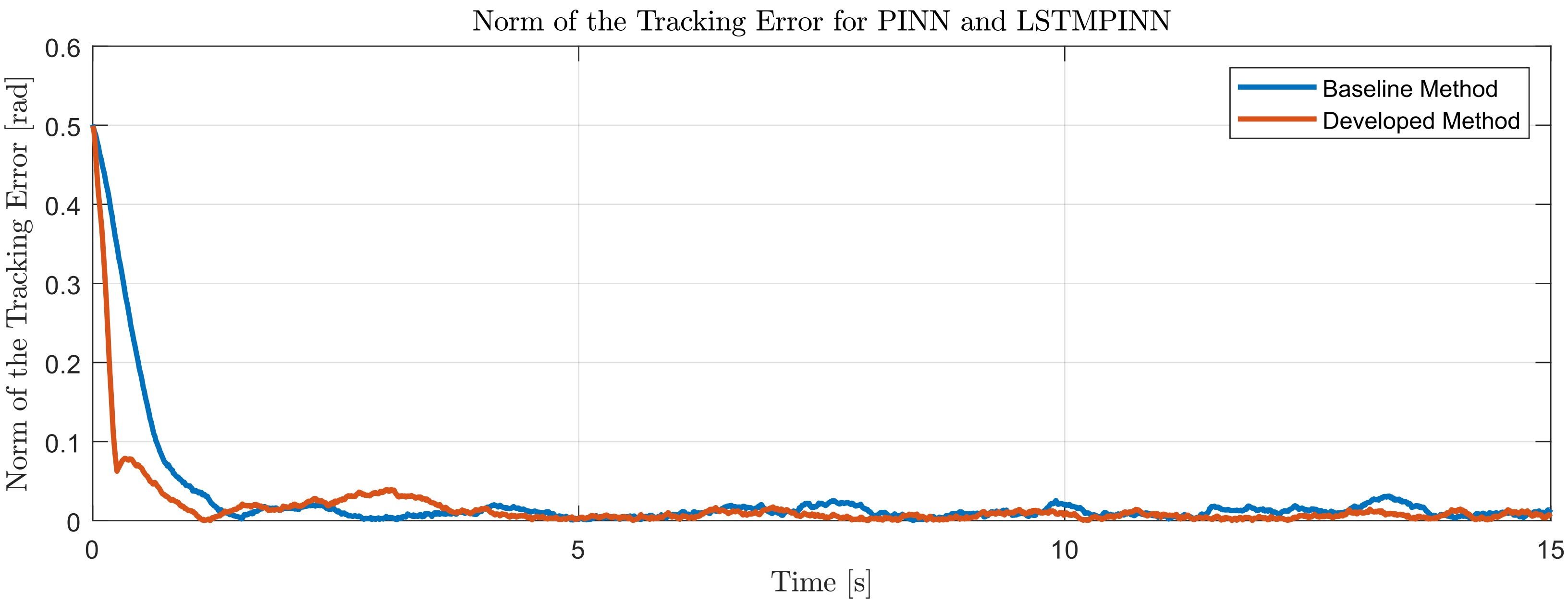
Combined DNN + LSTM architecture described as

$$\chi_i(x_i, \theta_i, c_i, h_i, \vartheta_i) \triangleq \underbrace{\Phi_i(x_i, \theta_i)}_{\text{DNN}} + \underbrace{\Xi_i(x_i, c_i, h_i, \vartheta_i)}_{\text{LSTM}}$$

Simulations

Simulations compared a DeLb-PINN controller to a baseline

The simulation results demonstrated a 33.76% improvement over the baseline method



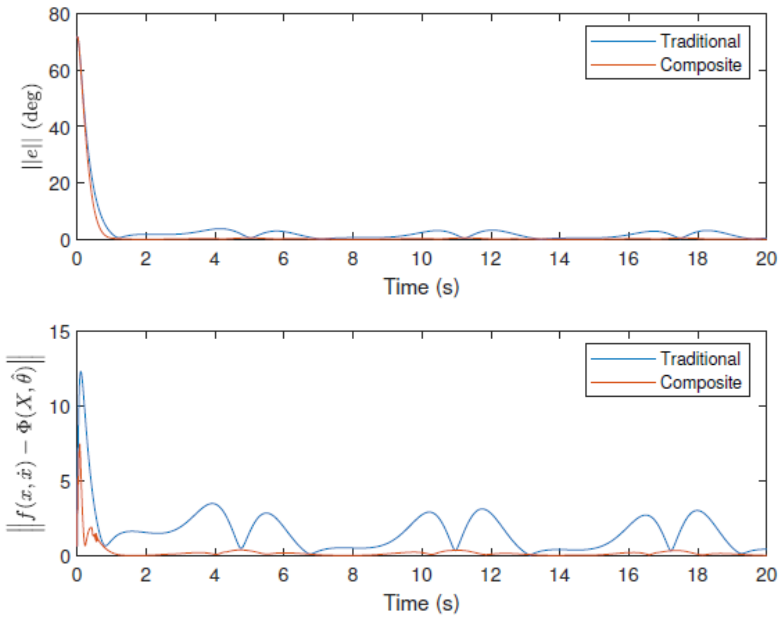
Composite Adaptive Lyapunov-Based Deep Neural Network Control

Omkar Sudhir Patil, Emily J. Griffis, Wanjiku A. Makumi, and Warren E. Dixon

Composite Adaptive DNN

- Emerging Lb-DNNs use adaptation laws where the adaptation is tracking error-based and only guarantees tracking performance
- No guarantees on weight estimation and function approximation
- Desirable to incorporate a prediction error of the dynamics in the adaptation law $\dot{\hat{\theta}} = \Gamma\left(-k_{\hat{\theta}}\,\hat{\theta} + \Phi'^{\top}\left(X, \hat{\theta}\right)(r + \alpha_3 E)\right)$
- We develop a new formulation of a prediction error motivated from traditional composite adaptive control, that ensures parameter convergence (function approximation convergence) provided a PE condition is satisfied

Simulation Results



- Fully-connected DNN with 5 layers and 5 neurons
- Reference Trajectory $x_d(t) = 0.25\exp(-\sin t)\,[\sin t\,;\cos t]$

	RMS $\|e\|$ (deg)	RMS $\|f - \Phi(X,\hat{\theta})\|$
Traditional	0.408	0.455
Composite	0.180	0.130

Acknowledgements

UF UNIVERSITY of FLORIDA

Duke UNIVERSITY

TEXAS The University of Texas at Austin

UC SANTA CRUZ