Development of a Deep Recurrent Neural Network Controller for Flight Applications

American Control Conference (ACC)
May 26, 2017

Scott A. Nivison
Pramod P. Khargonekar

Department of Electrical and Computer Engineering
University of Florida

UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Outline

- Inspiration
- DLC Limitations and Research Goals
- Background – Direct Policy Search
- Flight Vehicle (Plant) Model
- Architecture - Deep Learning Flight Control
- Optimization - Deep Learning Controller
- Simulation Results
- Conclusions
- Future Work

Inspiration

Deep Learning (DL):

- Dominant performance in multiple machine learning areas:

  Speech, language, vision, face recognition, etc.
- Replaces time consuming typical hand-tuned machine learning methods

Key Breakthroughs in establishing Deep Learning:

- Optimization methods: Stochastic Gradient Descent, Hessian-free, Nestorov's momentum, etc.
- Deep recurrent neural network (RNN) architectures: deep stacked RNN, deep bidirectional RNN, etc.
- RNN Modules: Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM)
- Automatic gradient computation
- Parallel computing

Connecting Deep Learning to Control:

- Guided Policy Search – uses trajectory optimization to assist policy learning (S. Levine and V. Koltun, 2013)
- Hessian-free Optimization– deep RNN learning control with uncertainties (I. Sutskever, 2013)

DLC Limitations and Research Goals

Policy Search/Deep Learning (DL) Control Limitations:

- Large number of computations at each time step
- No standard training procedures for control tasks
- Few analysis tools for deep neural network architectures
- Non-convex form of optimization provides few guarantees
- Lack of research in optimization with regards to robustness
- Most RL approaches use the combination of modeled dynamics and real-life trial to tune policy (not useful for flight control)

Research Goals

Develop a robust deep recurrent neural network (RNN) controller with gated recurrent unit (GRU) modules for a highly agile high-speed flight vehicle that is trained using a set of sample trajectories that contain disturbances, aerodynamic uncertainties, and significant control attenuation/amplification in the plant dynamics during optimization.

Direct Policy Search Background

Reinforcement Learning (RL):

Develop methods to sufficiently train an agent by maximizing a cost function through repeated interactions with its environment.

Markovian Dynamics

$$x_{t+1} = f(x_t, u_t) + w, \qquad x_0 \sim p(x_0)$$

Find Parametrized Policy (π_Θ^*) for the finite horizon problem:

$$\pi_\Theta^* = argmax_{\pi_\Theta} J_\Theta = argmax_\pi \Sigma_{t=1}^{t_f} \gamma^t E[r(x_t, u_t)|\pi_\Theta], \qquad \gamma \in [0,1]$$

Certainty-Equivalence (CE) Assumption:

The optimal policy for the learned model corresponds to the optimal policy for the true dynamics

How to use models for long-term predictions:

1. Stochastic inference (i.e. trajectory sampling)
2. Deterministic approximate inference

*M. P. Deisenroth, et al., "A survey on policy search for robotics," Foundations and Trends in Robotics, vol. 2, 2013.

Direct Policy Search Background

Stochastic Sampling:

Expected long-term reward:

$$J_\Theta = argmax_\pi \Sigma_{t=1}^{t_f} \gamma^t \mathrm{E}[r(x_t, u_t)|\pi_\Theta]$$

Approximation of J_Θ:

$$\widetilde{J_\Theta} = \frac{1}{N} \Sigma_{i=1}^{N} \Sigma_{t=1}^{t_f} \gamma^t r(x_t^i) \qquad \lim_{N \to \infty} \widetilde{J_\Theta} = J_\Theta$$

Deterministic Approximations:

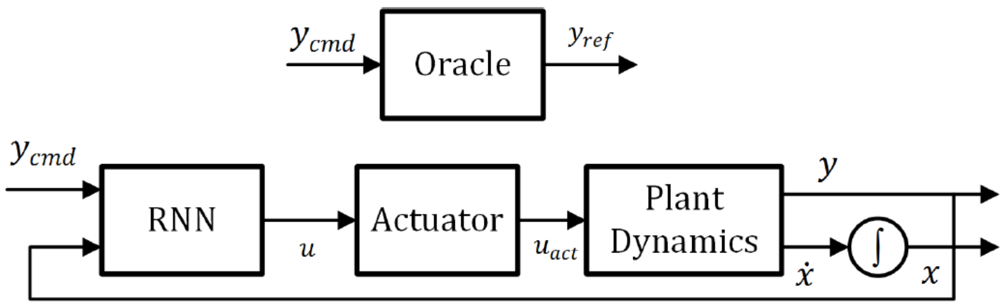
Approximation of $p(x_t)$: (e.g. Unscented Transformation or Moment Matching)

$$E[r(x_t, u_t)|\pi_\Theta] = \int r(x_t) p(x_t) dx_t$$

$$p(x_t) \cong \mathrm{N}(x_t | \mu_t^x, \Sigma_t^x)$$

*M. P. Deisenroth, et al., "A survey on policy search for robotics," Foundations and Trends in Robotics, vol. 2, 2013.

Deep Learning based Flight Control



Deep Learning Training Architecture

Generalized Form of the Discrete Plant Dynamics

$$
\begin{aligned}
x_{t+1} &= f(x_t, (\lambda_u(u_t^{act} + \rho_u)) + d_u, \rho_\alpha, \rho_q) + \zeta_p \\
y_t &= f(x_t, (\lambda_u(u_t^{act} + \rho_u)) + d_u, \rho_\alpha, \rho_q) + \zeta_p
\end{aligned}
$$

x_t are the states of the plant
u_t^{act} is the actuator output
y_t is the output vector
ζ_p is the plant noise
λ_u is the control effectiveness
d_u is an input disturbance
$\rho_u, \rho_\alpha, \rho_q$ are uncertainty parameters

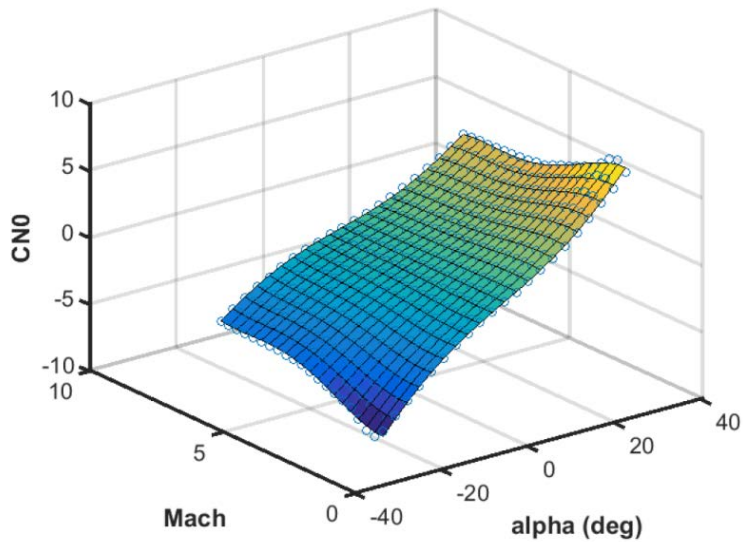
Flight Vehicle Model

Longitudinal Rigid Body Dynamics:

$$
\begin{aligned}
\dot{V}_T &= \frac{1}{m}(Tcos(\alpha) - D) - gsin(\theta - \alpha) \\
\dot{\alpha} &= \frac{1}{mV_T}(-Tsin(\alpha) - L) + q + \frac{g}{V_T}cos(\theta - \alpha) \\
\dot{\Theta} &= q \\
\dot{q} &= \frac{M}{I_{YY}} \\
\dot{h} &= V_T sin(\theta - \alpha)
\end{aligned}
$$

$$x = [V_T,\, \alpha,\, \Theta,\, q,\, h]$$

$$y = [\alpha,\, q,\, A_z,\, \bar{q}]$$



Force and Moment Equations:

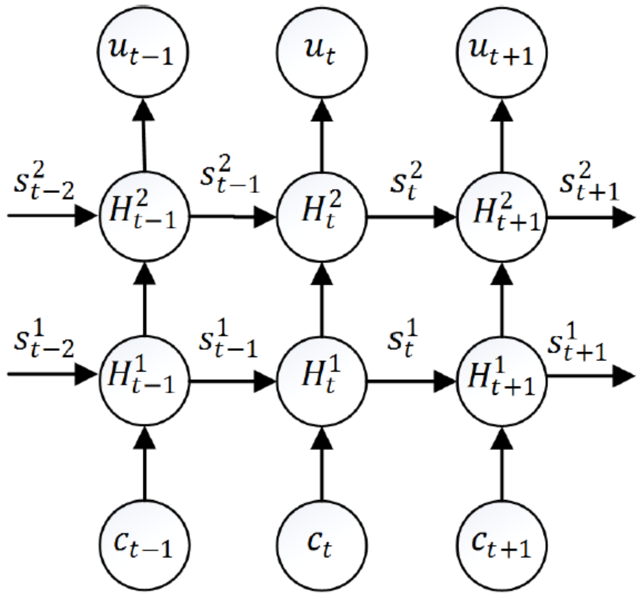
$$
\begin{aligned}
A &\approx \frac{1}{2}\rho V_T^2 S C_A & L &= Ncos(\alpha) - Asin(\alpha) \\
N &\approx \frac{1}{2}\rho V_T^2 S C_N & D &= Nsin(\alpha) + Acos(\alpha) \\
M &\approx \frac{1}{2}\rho V_T^2 S c_{ref} C_m
\end{aligned}
$$

Aerodynamic Coefficients (Longitudinal):

$$
\begin{aligned}
C_A &= C_{A_{ALT}}(h,\, M) + C_{A_{AB}}(\alpha,\, M) \\
&\quad + \sum_{i=1}^{4} C_{A_{\delta_i}}(\alpha,\, M,\, \delta_i) \\
C_N &= C_{N_0}(\alpha,\, M) + \sum_{i=1}^{4} C_{N_{\delta_i}}(\alpha,\, M,\, \delta_i) \\
C_m &= C_{m_0}(\alpha,\, M) + \sum_{i=1}^{4} C_{m_{\delta_i}}(\alpha,\, M,\, \delta_i) \\
&\quad + C_{m_q}(\alpha,\, M,\, q) + q\rho_q + \alpha\rho_\alpha
\end{aligned}
$$

Deep Learning Controller - Architecture

Stacked Recurrent Neural Network (S-RNN)



Algorithm 1 Gated Recurrent Units (GRU)

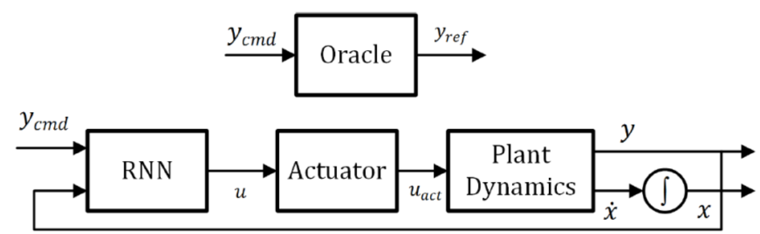
1: $z = \sigma(c_t U^u + s_{t-1} W^u + b_1)$
2: $r = \sigma(c_t U^r + s_{t-1} W^r + b_2)$
3: $h = tanh(c_t U^h + (s_{t-1} * r) W^h + b_3)$
4: $s_t = (1 - z) * h + z * s_{t-1}$
5: $*$ represents element-wise multiplication

$$u_t = s_t V + c$$

Θ_i matrix of parameters for each GRU module
$\Theta_i = [U^u, W^u, U^r, W^r, U^h, W^h, b_1, b_2, b_3]$
Θ total parameters of the controller
$\Theta = [\Theta_1, \Theta_2, \dots, \Theta_L, V, c]$
L total layers of GRU modules

Controller Input Vector:

$c_t = [e_i, \alpha, q, \bar{q}]$
$e = y_{sel} - y_{cmd}$
$e_i = \int_0^{t_f} e \, dt$



UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Deep Learning Controller - Optimization

Estimated Expected Long-Term Reward:

$$\tilde{J}(\Theta) = \frac{1}{N}\Sigma_{i=1}^{N} J_i(\Theta)$$

Cost function for each trajectory, i:

$$J_i(\Theta) = \Sigma_{t=0}^{t_f} \gamma^t \chi(x_t, u_t)$$

$label = P$: only uncertainties
$label = R$: uncertainties, noise, and disturbances

Instantaneous Measurement:

$$\chi(x_t, u_t) = \begin{cases} k_1 e_t^2 + k_2 f_{\dot{u}}^2 & if\ label = P \\ k_3 f_e^2 + k_4 f_{\dot{u}}^2 & if\ label = R \end{cases}$$

$$f_e = \max(|e_t| - b_e, 0)$$
$$f_{\dot{u}} = \max(|\dot{u}_t| - b_{\dot{u}}, 0)$$

Time-varying funnel:

$$\beta_{\dot{u}}(t) = \{x \in \mathbb{R}^n | U(x,t) \le b_{\dot{u}}(t)\}$$
$$\beta_e(t) = \{x \in \mathbb{R}^n | E(x,t) \le b_e(t)\}$$

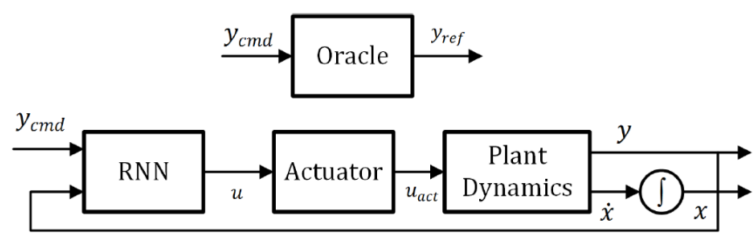
TABLE I

RANGE OF INITIAL CONDITIONS AND UNCERTAINTY VARIABLES

	MIN	MAX
α_0[deg]	-25	25
q_0[deg/sec]	-75	75
$Mach_0$	1.0	2.0
alt_0[km]	7	14

	MIN	MAX
R_α	-0.5	0.1
R_q	-7	5
R_u	-5	5
Λ_u	0.25	3.0

$[R_\alpha, R_q, R_u, \Lambda_u] \sim Uniform[min, max]$
$[\rho_\alpha^i, \rho_q^i, \rho_u^i, \lambda_u^i]$ are uncertainties (ith trajectory)
N is the number of sampled trajectories
t_f is the duration of each sampled trajectory
k_1, k_2, k_3, k_4 are positive constant gains
$b_e, b_{\dot{u}}$ are static constant bounds of the funnel
$e_t = y_{sel} - y_{ref}$



UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

DLC – Incremental Training

Algorithm 2 Incremental Training Procedure

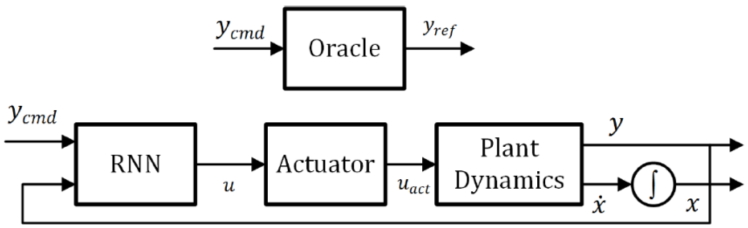
1: Randomly initialize controller parameters (Θ)
2: STEP 1: Optimize Θ for RNN/GRU using cost function (21) with linear dynamics and labels=P
3: STEP 2: Re-optimize Θ using nonlinear dynamics, labels=P, with small uncertainties in aerodynamics
4: STEP 3: Re-optimize Θ using nonlinear dynamics, labels=(R,P), with uncertainties, disturbances, and noise

Optimization Specifications:

RNN/GRU, RNN, TD-FNN
L-BFGS (Quasi-Newton) Optimization

N = 2,970 Sample Trajectories
540 Performance Trajectories
2,430 Robust Trajectories

3.1 GHz PC with 256 GB RAM and 10 cores
Parallel Processing
~40 hours for 1,000 sample trajectories for optimization



DLC - Results

Flight Condition:

Mach	α (deg)	q (deg/sec)	Altitude (km)
1.7	0	0	14

Initial Conditions:

	MIN	MAX
α_0(deg)	-30	30
q_0(deg/sec)	-100	100
R_α	-0.5	0.1
R_q	-7	5
R_u	-5	5
Λ_u	0.25	3.0

Augmented polynomial short period model:

$$
\begin{aligned}
\dot{e}_I &= \alpha - \alpha_{cmd} \\
\dot{\alpha} &= f(\alpha, q, \delta_e, \rho_\alpha, \rho_q, \rho_u, \lambda_u) \\
\dot{q} &= f(\alpha, q, \delta_e, \rho_\alpha, \rho_q, \rho_u, \lambda_u)
\end{aligned}
$$

DLC - Results

Flight Condition:

Mach	α (deg)	q (deg/sec)	Altitude (km)
1.7	0	0	14

Initial Conditions:

	MIN	MAX
α_0(deg)	-30	30
q_0(deg/sec)	-100	100
R_α	-0.5	0.1
R_q	-7	5
R_u	-5	5
Λ_u	0.25	3.0

Augmented polynomial short period model:

$$\begin{aligned} \dot{e}_I &= \alpha - \alpha_{cmd} \\ \dot{\alpha} &= f(\alpha, q, \delta_e, \rho_\alpha, \rho_q, \rho_u, \lambda_u) \\ \dot{q} &= f(\alpha, q, \delta_e, \rho_\alpha, \rho_q, \rho_u, \lambda_u) \end{aligned}$$

Performance Metrics:

$$ATE = \frac{1}{N}\Sigma_{i=1}^{N}\Sigma_{t=1}^{t_f}|e_t|$$

$$ACR = \frac{1}{N}\Sigma_{i=1}^{N}\Sigma_{t=1}^{t_f}|\dot{u}_t|$$

$$CTE = \Sigma_{j=1}^{M} ATE$$

$$CCR = \Sigma_{j=1}^{M} ACR$$

TABLE II

CUMULATIVE ERROR (CTE), CONTROL RATE (CCR), AND FINAL COST

	CTE	CCR	Cost
3-Layer RNN/GRU	339.15	100.16	1.0438
2-Layer RNN	359.28	313.64	2.4970
GS	1000.21	1180.04	-

DLC Performance: 66% reduction in CTE, 91.5% reduction in CCR

N number of trajectories for each analysis model
t_f is the duration of each sampled trajectory
CTE is the cumulative tracking error
CCR is the cumulative control rate

DLC - Results

Flight Condition:

Mach	α (deg)	q (deg/sec)	Altitude (km)
1.7	0	0	14

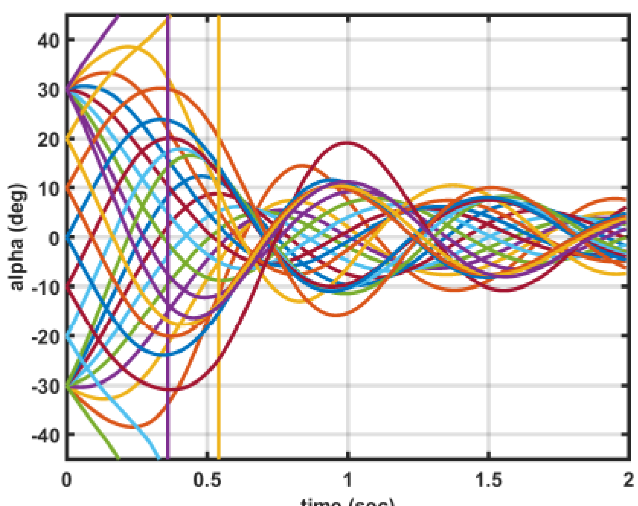
Initial Conditions:

	MIN	MAX
α_0(deg)	-30	30
q_0(deg/sec)	-100	100
R_α	-0.5	0.1
R_q	-7	5
R_u	-5	5
Λ_u	0.25	3.0

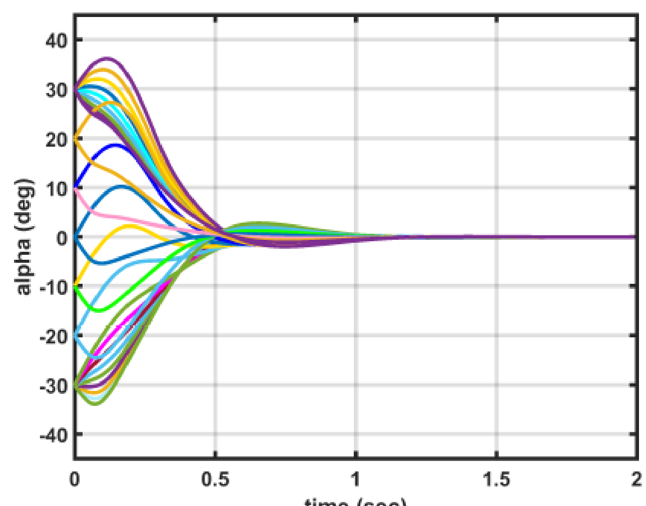
Augmented polynomial short period model:

$$
\begin{aligned}
\dot{e}_I &= \alpha - \alpha_{cmd} \\
\dot{\alpha} &= f(\alpha,\, q,\, \delta_e,\, \rho_\alpha,\, \rho_q,\, \rho_u,\, \lambda_u) \\
\dot{q} &= f(\alpha,\, q,\, \delta_e,\, \rho_\alpha,\, \rho_q,\, \rho_u,\, \lambda_u)
\end{aligned}
$$

Gain Scheduled Controller



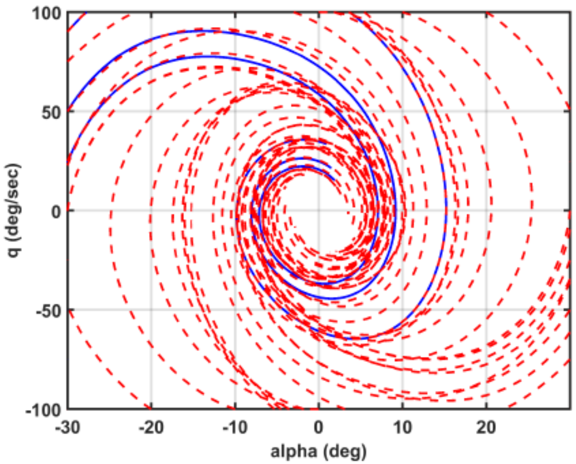
Deep Learning Controller



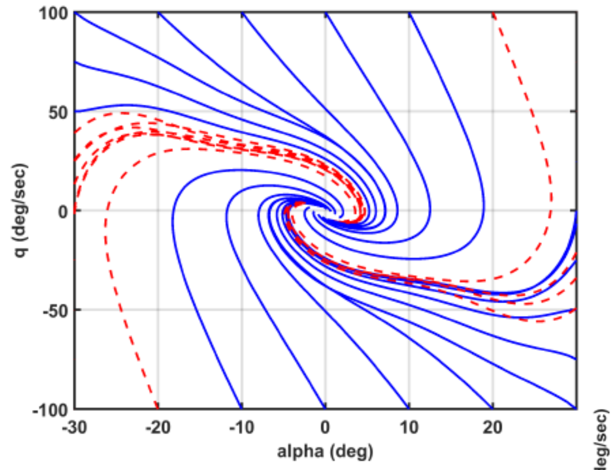
$\lambda_u = 0.5,\ \rho_u = 0,\ \rho_\alpha = 0.05$, and $\rho_q = 2.5$

DLC - Results

Gain Scheduled Controller

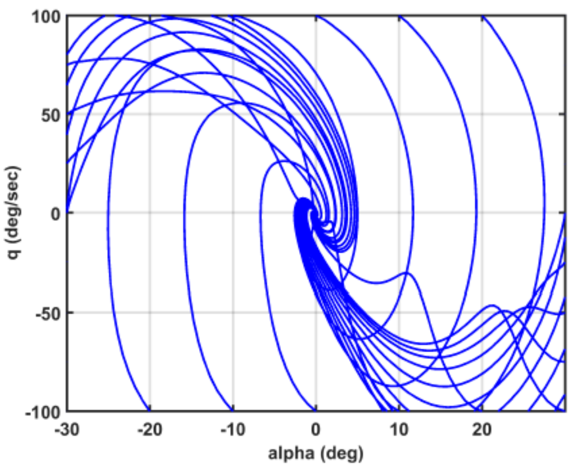


$\lambda_u = 0.75,\ \rho_u = 0,\ \rho_\alpha = 0.025, \text{ and } \rho_q = 5$

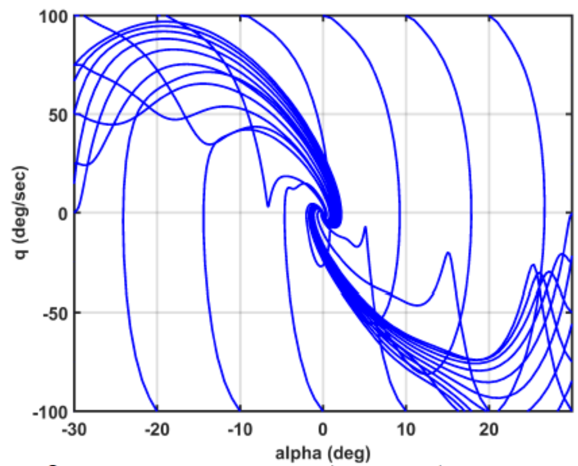


$\lambda_u = 0.5,\ \rho_u = 0,\ \rho_\alpha = 0.05, \text{ and } \rho_q = 2.5$

Deep Learning Controller

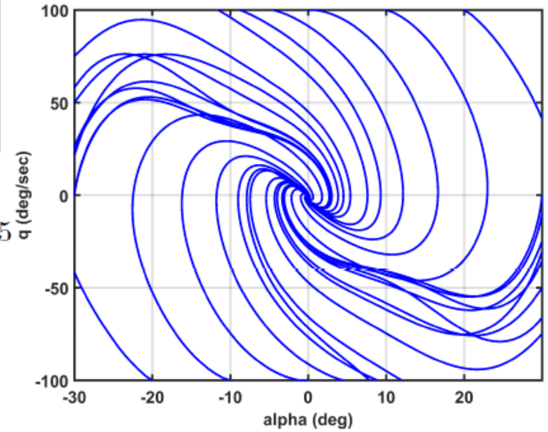


$\lambda_u = 0.75,\ \rho_u = 0,\ \rho_\alpha = 0.025, \text{ and } \rho_q = 5$



$\lambda_u = 0.5,\ \rho_u = 0,\ \rho_\alpha = 0.05, \text{ and } \rho_q = 2.5$

Gain Scheduled Controller



No uncertainty or disturbances

Contribution and Conclusion

- Created a novel training procedure focused on bringing deep learning benefits to flight control.
- Trained controller using a set of sample trajectories that contain disturbances, aerodynamic uncertainties, and significant control attenuation/amplification in the plant dynamics during optimization.
- We found benefits of using a piecewise cost function that allows the designer to solve both robustness and performance criteria simultaneously.
- We utilized an incremental initialization training procedure for deep recurrent neural networks.

Future Work

- Pursue a vehicle model with flexible body effects, time delays, controller effectiveness, center of gravity changes, and aerodynamic parameter variations.
- Explore improving parameter convergence and analytic guarantees: Kullback-Leibler (KL) divergence, importance sampling, etc.
- Pursue development/use of robustness analysis tools for deep learning controllers to provide region of attraction estimates and time delay margins: sum-of-squares programming etc.

Questions?