

# **Large Angle Maneuvering Using Adaptive Critic Control**

**Silvia Ferrari**

**Advisor: Prof. Robert F. Stengel**

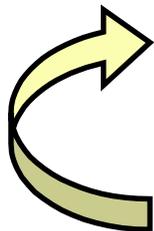
**Princeton University**

**FAA/NASA Joint University Program on Air Transportation,  
Ohio University, Athens, OH**

**June 21-22, 2001**

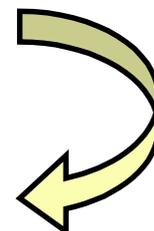
# Introduction

- **Classical/neural synthesis** of control systems
  - Linear control theory
  - Artificial neural networks
- **Adaptive critics**
  - Learn in real time
  - Cope with noise
  - Cope with many variables
  - Plan over time in a complex way
  - ...
- Adaptation takes place during every time interval:



**Action network** takes immediate control action

**Critic network** estimates projected cost



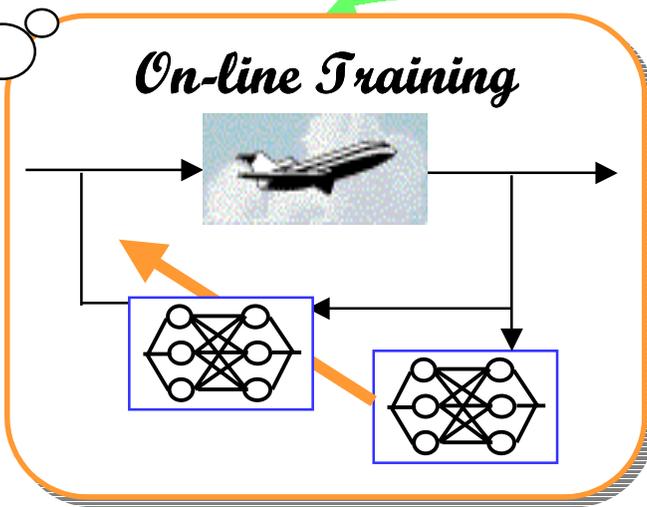
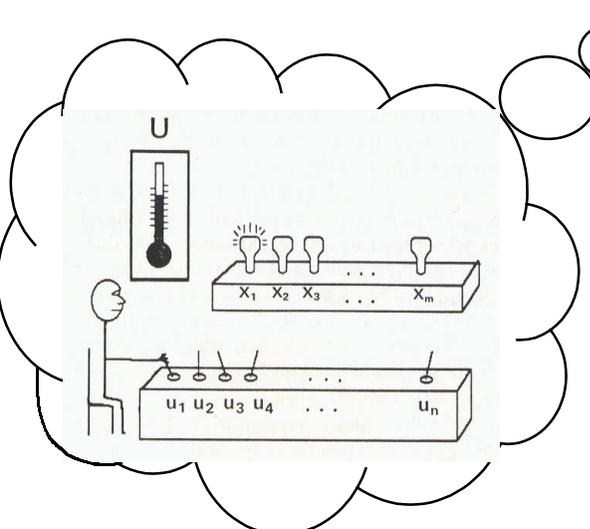
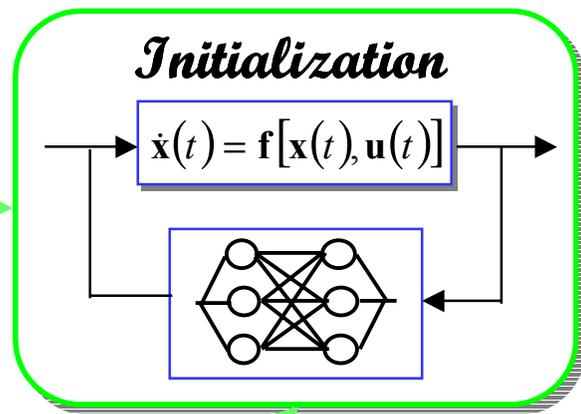
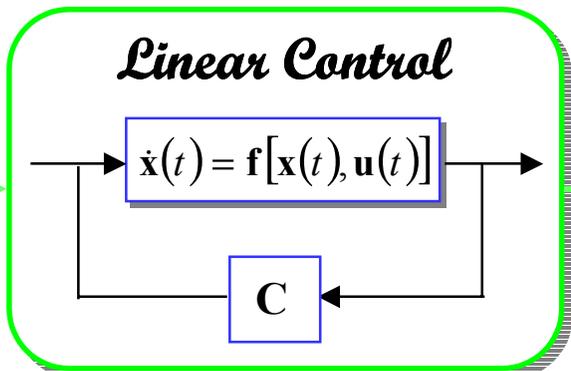
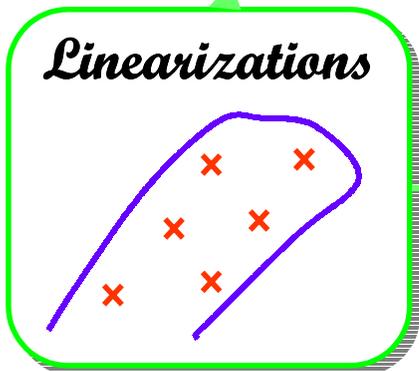
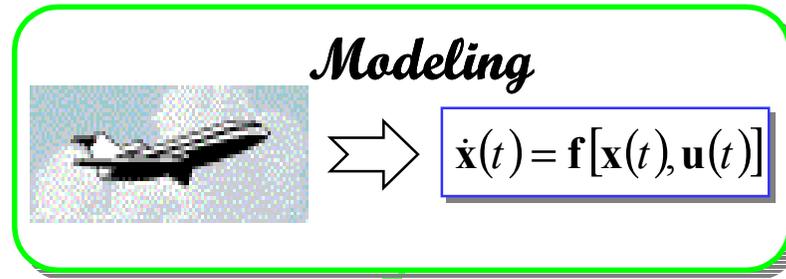
# Motivation

- Provide full envelope control
- **Multiphase** learning
  - Pre-training (off-line), motivated by linear controllers
  - On-line training, during full-scale simulations or aircraft testing
- On-line training improves performance w.r.t. linear controllers:
  - Differences** between **actual** and **assumed** dynamic models
  - Nonlinear effects** not captured in linearizations
- **Potential applications:**
  - Incorporate pilot's knowledge into controller *a-priori*
  - Uninhabited air vehicles control
  - Aerobatic flight control

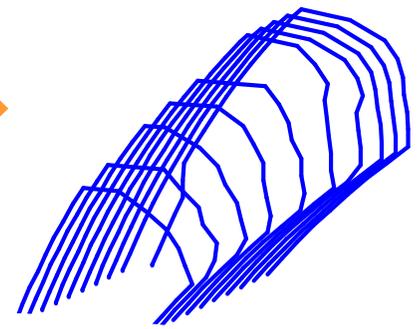
# Table of Contents

- Aircraft control design approach
- Pre-training phase
- Adaptive critic neural network control
- On-line training:
  - Resilient backpropagation algorithm
  - Improvements and implementation
- Results: large angle maneuvering

# Aircraft Control Design Approach



*Full Envelope Control!*



# Linear Control Design

## *Linearizations:*

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t)]$$



$$\Delta \dot{\mathbf{x}}(t) = \mathbf{F} \Delta \mathbf{x}(t) + \mathbf{G} \Delta \mathbf{u}(t)$$

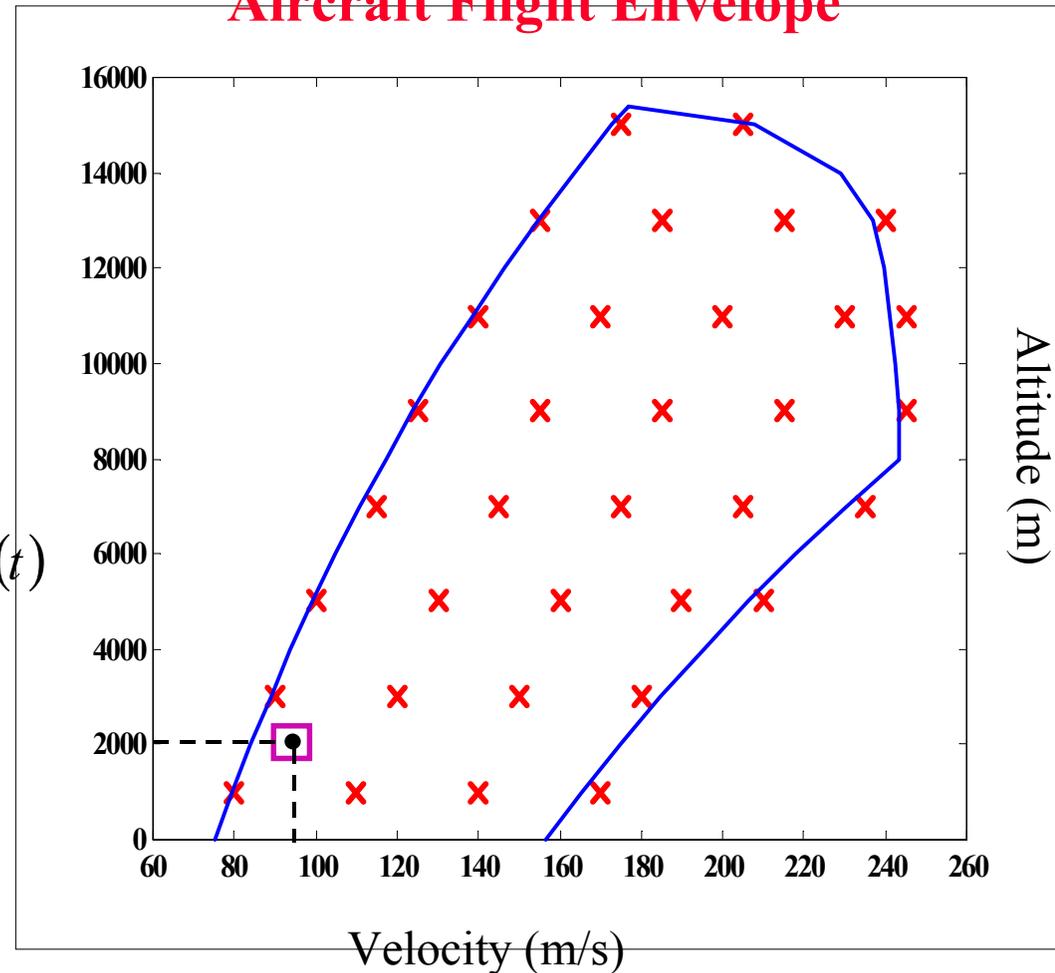


$$\begin{cases} \Delta \dot{\mathbf{x}}_L(t) = \mathbf{F}_L \Delta \mathbf{x}_L(t) + \mathbf{G}_L \Delta \mathbf{u}_L(t) \\ \Delta \dot{\mathbf{x}}_{LD}(t) = \mathbf{F}_{LD} \Delta \mathbf{x}_{LD}(t) + \mathbf{G}_{LD} \Delta \mathbf{u}_{LD}(t) \end{cases}$$

## *Linear control design:*

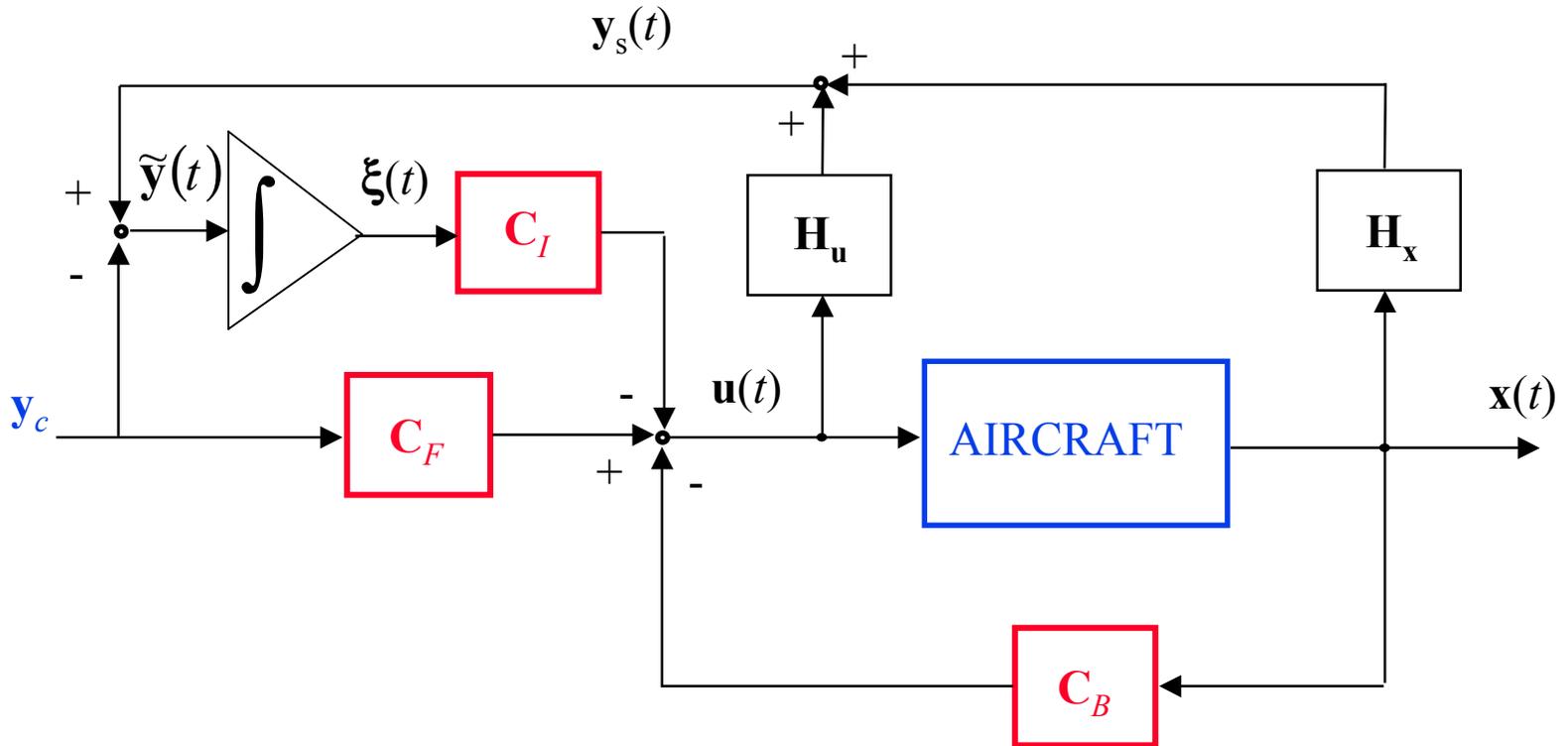
- Longitudinal
- Lateral-directional

## Aircraft Flight Envelope



# Linear Proportional-Integral Controller

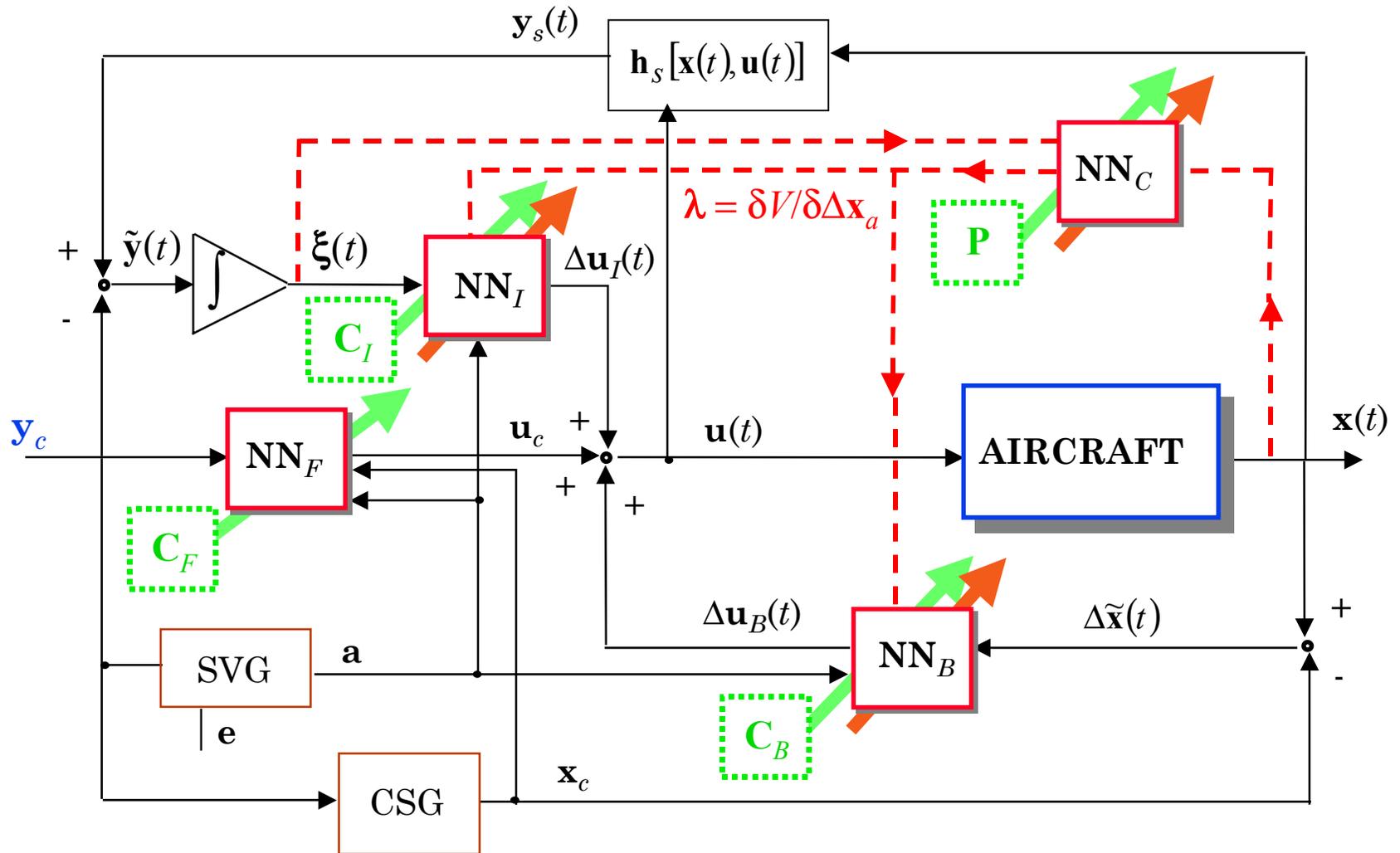
Closed-loop stability:  $\mathbf{x}(t) \rightarrow \mathbf{x}_c$ ,  $\mathbf{u}(t) \rightarrow \mathbf{u}_c$ ,  $\tilde{\mathbf{y}}(t) \rightarrow 0$



Omitting  $\Delta$ 's, for simplicity:

$\tilde{\mathbf{y}}(t) = \mathbf{y}_s(t) - \mathbf{y}_c$ ,  $\tilde{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_c, \dots$ ,  $\mathbf{y}_c =$  desired output,  $(\mathbf{x}_c, \mathbf{u}_c) =$  set point.

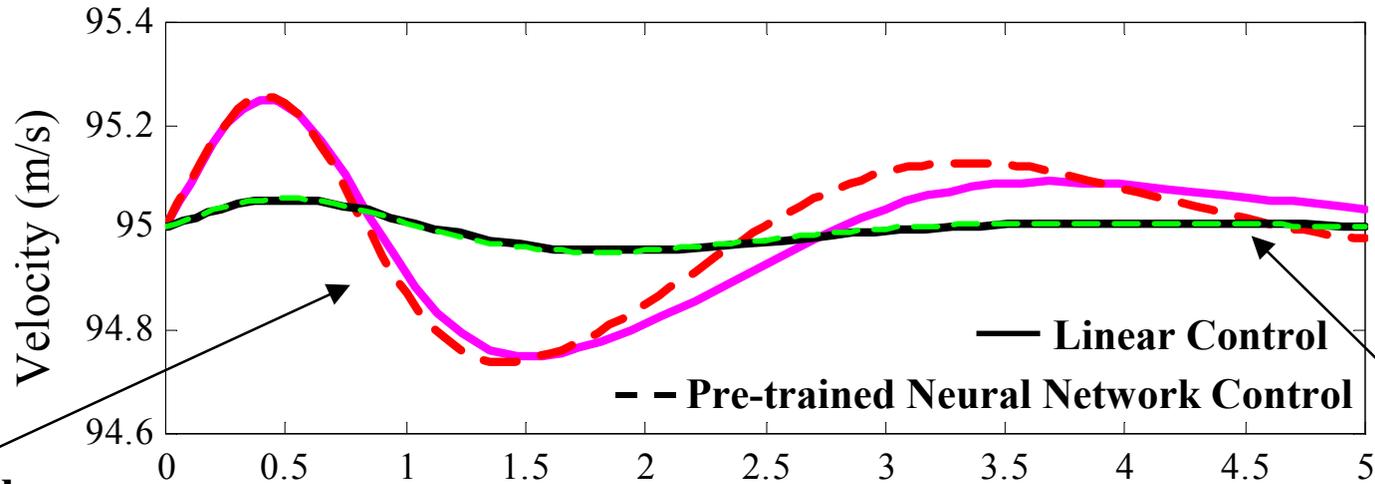
# Proportional-Integral Neural Network Controller



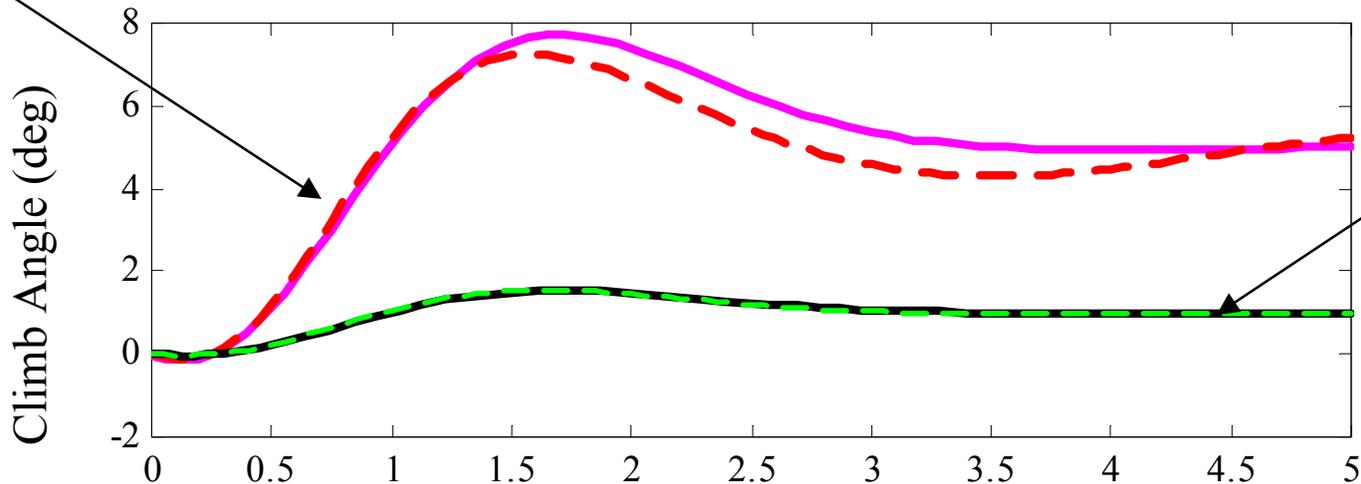
Where:  $\tilde{y}(t) \rightarrow 0$ ,  $y_s(t) \rightarrow y_c$ ,  $\nearrow$ : Algebraic Training,  $\nearrow$ : On-line Training.

# Comparison of Neural Network and Linear Controllers Between Training Points, at Flight Conditions $(H_0, V_0) = (2\text{Km}, 95 \text{ m/s})$

## Aircraft Response to Climb-Angle Command Input



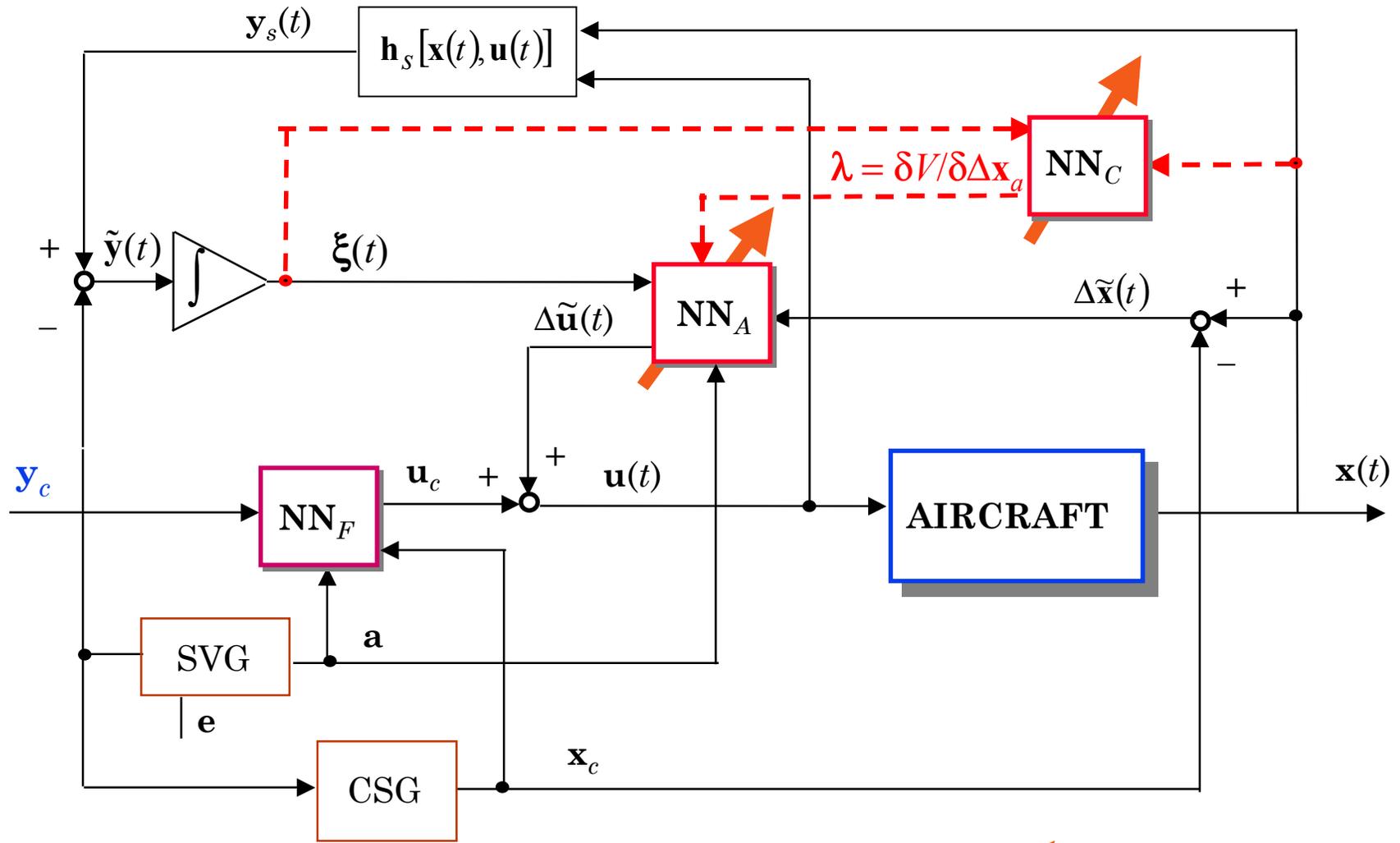
Large-Angle  
Maneuver



Small-Angle  
Maneuver

Time (sec)

# Proportional-Integral Neural Network Controller: Action and Critic Networks Implementation

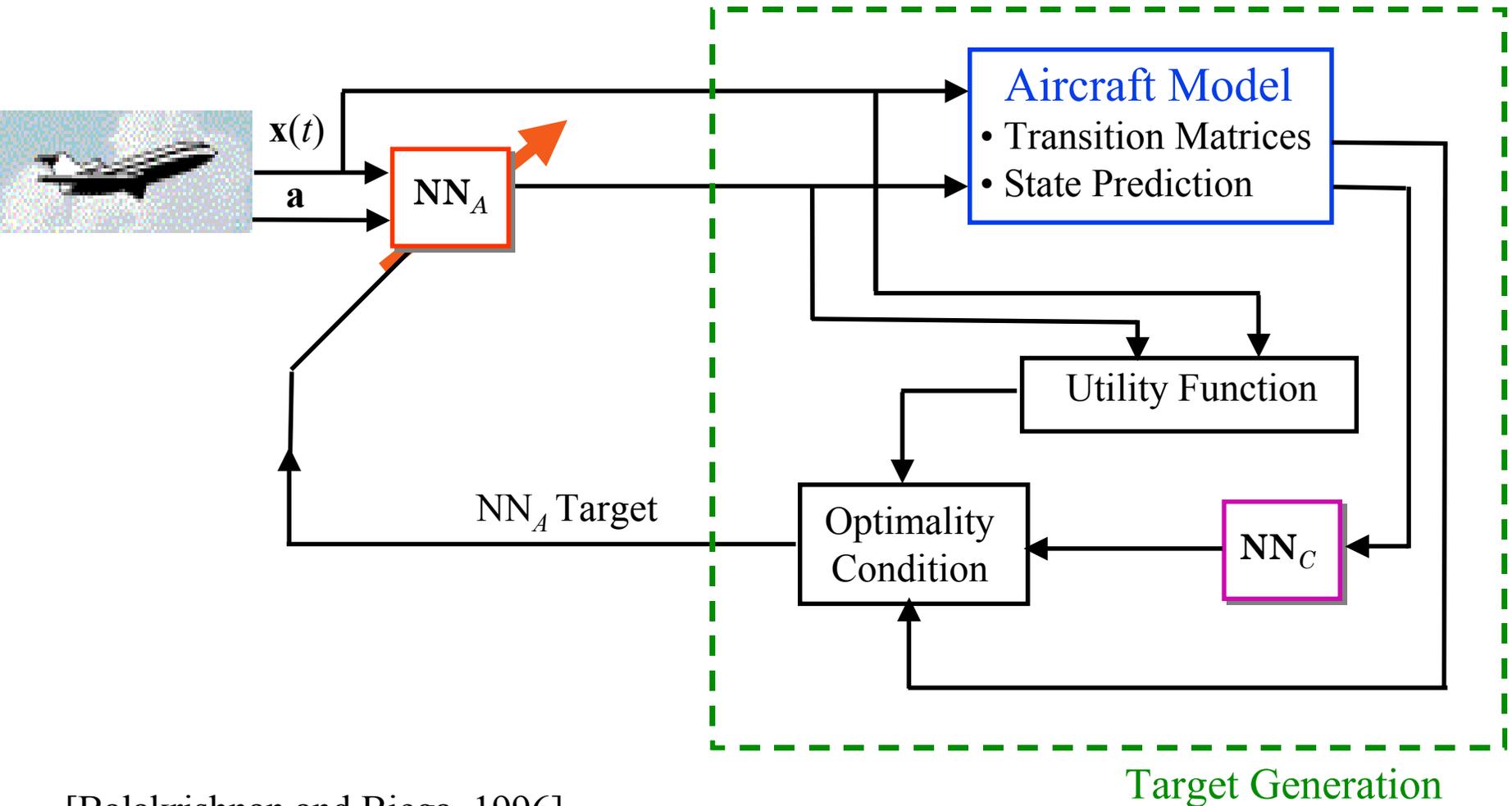


 : On-line Training

# Adaptive Critic Implementation: Action Network

## On-line Training

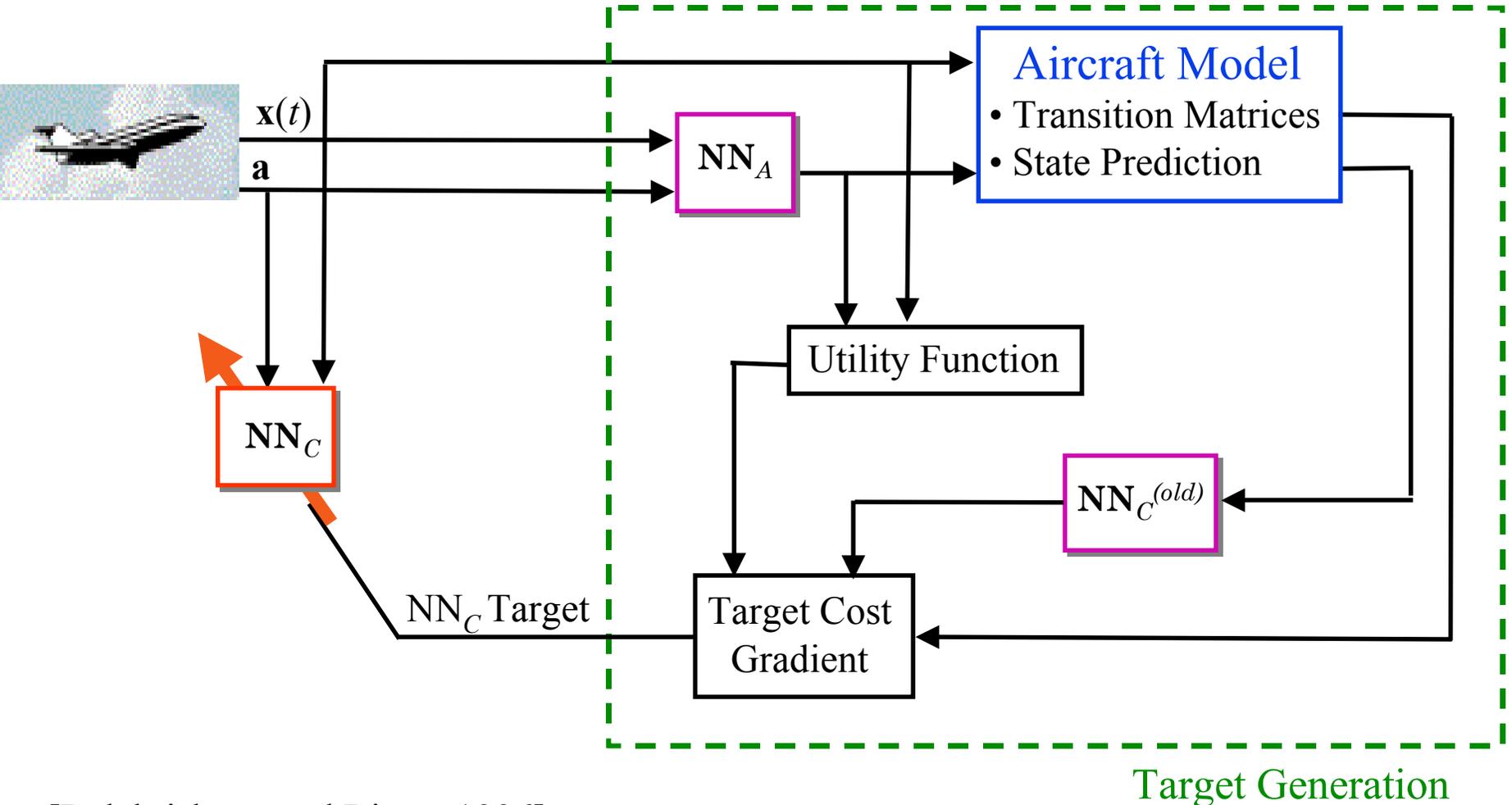
Train action network, at time  $t$ , holding the critic parameters fixed



# Adaptive Critic Implementation: Critic Network

## On-line Training

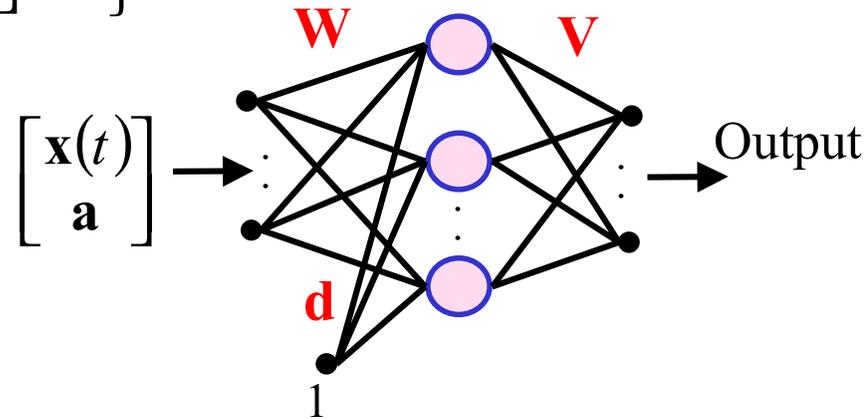
Train critic network, at time  $t$ , holding the action parameters fixed



# Action/Critic Network On-line Training at Time $t$

$$\text{Neural network output} = \mathbf{V}\sigma\left\{\mathbf{W}\begin{bmatrix} \mathbf{x}(t) \\ \mathbf{a} \end{bmatrix} + \mathbf{d}\right\}$$

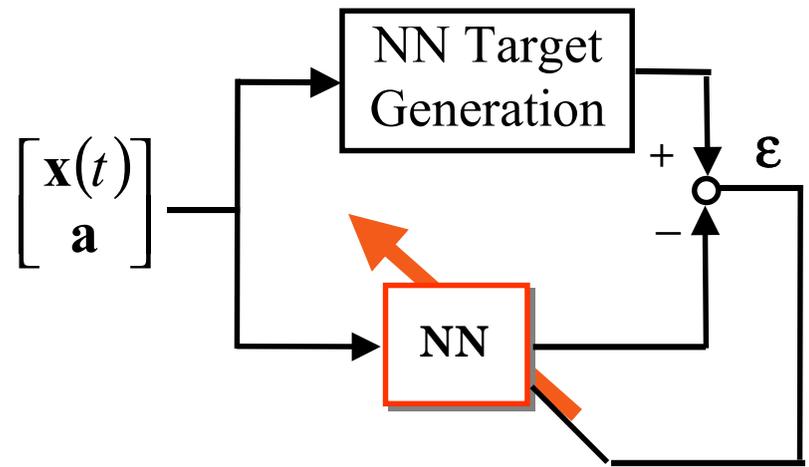
$$\text{Weight vector: } \mathbf{w} \equiv \begin{bmatrix} \text{Vec}(\mathbf{W}) \\ \mathbf{d} \\ \text{Vec}(\mathbf{V}) \end{bmatrix}$$



The (action/critic) network must meet its target,

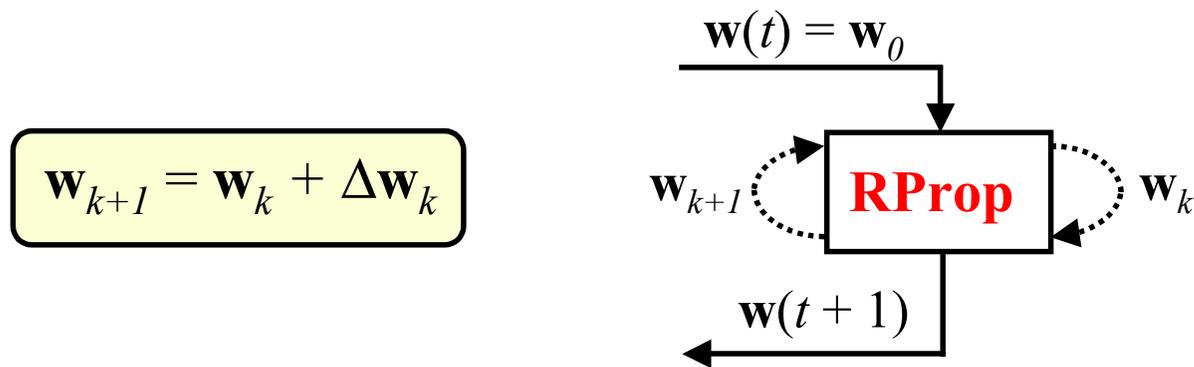
$$\min_{\mathbf{w}} E \equiv \min_{\mathbf{w}} |\boldsymbol{\varepsilon}|^2$$

where:  $\left\{ \begin{array}{l} E \equiv \text{Network performance} \\ \boldsymbol{\varepsilon} \equiv \text{Network error} \\ \text{Vec} \equiv \text{Vec operation} \end{array} \right.$



# Resilient Backpropagation for Network Error Minimization

At time  $t$ , given the weight vector,  $\mathbf{w}(t)$ , **Resilient Backpropagation (RProp)** minimizes  $E$ , by computing a new weight vector  $\mathbf{w}(t + 1)$ ,



During each epoch, the algorithm adjusts  $\mathbf{w}_k$  of an increment  $\Delta\mathbf{w}_k$ :

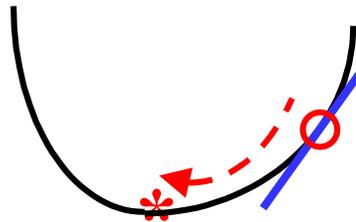
$t \equiv$  Real time

Epoch  $\equiv$  Algorithm iteration (indexed by  $k$ )

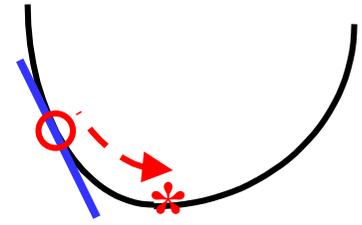
# MATLAB<sup>®</sup> Resilient Backpropagation Algorithm

The size and direction of each weight's increment,  $\Delta w_k$ , are based on the sign of the gradient of the performance,  $E$ , w.r.t. the weight,  $w$

## Increment Direction:

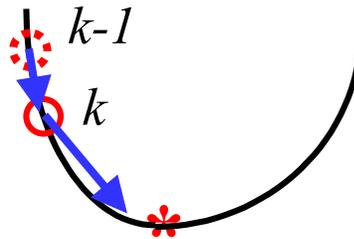


$$\left. \frac{\partial E}{\partial w} \right|_k > 0$$

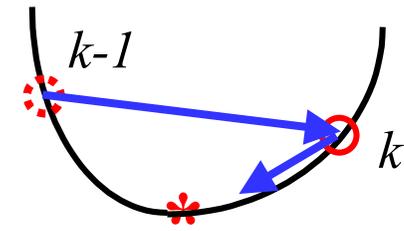


$$\left. \frac{\partial E}{\partial w} \right|_k < 0$$

## Increment Size:



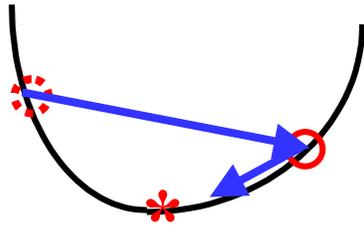
$$\left. \frac{\partial E}{\partial w} \right|_{k-1} \left. \frac{\partial E}{\partial w} \right|_k > 0$$



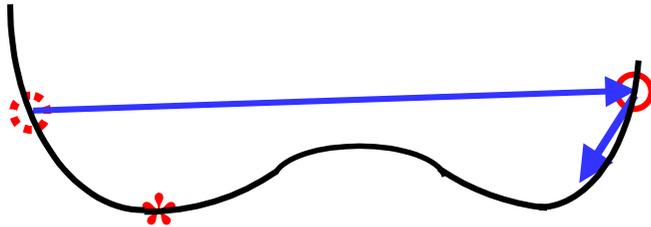
$$\left. \frac{\partial E}{\partial w} \right|_{k-1} \left. \frac{\partial E}{\partial w} \right|_k < 0$$

# Improving Resilient Backpropagation

## MATLAB<sup>®</sup> Algorithm



No backtracking

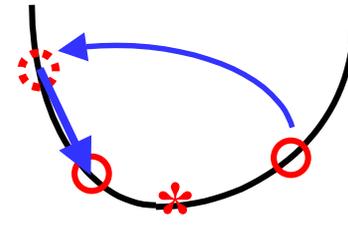


No backtracking, with local minima

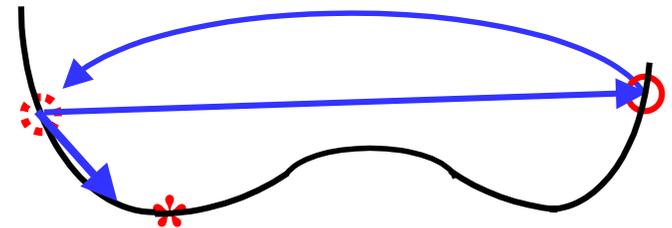
$$\Delta \mathbf{w}_0 = [0.07 \dots 0.07]^T$$

Arbitrary initial increment

## Improved Algorithm



Backtracking

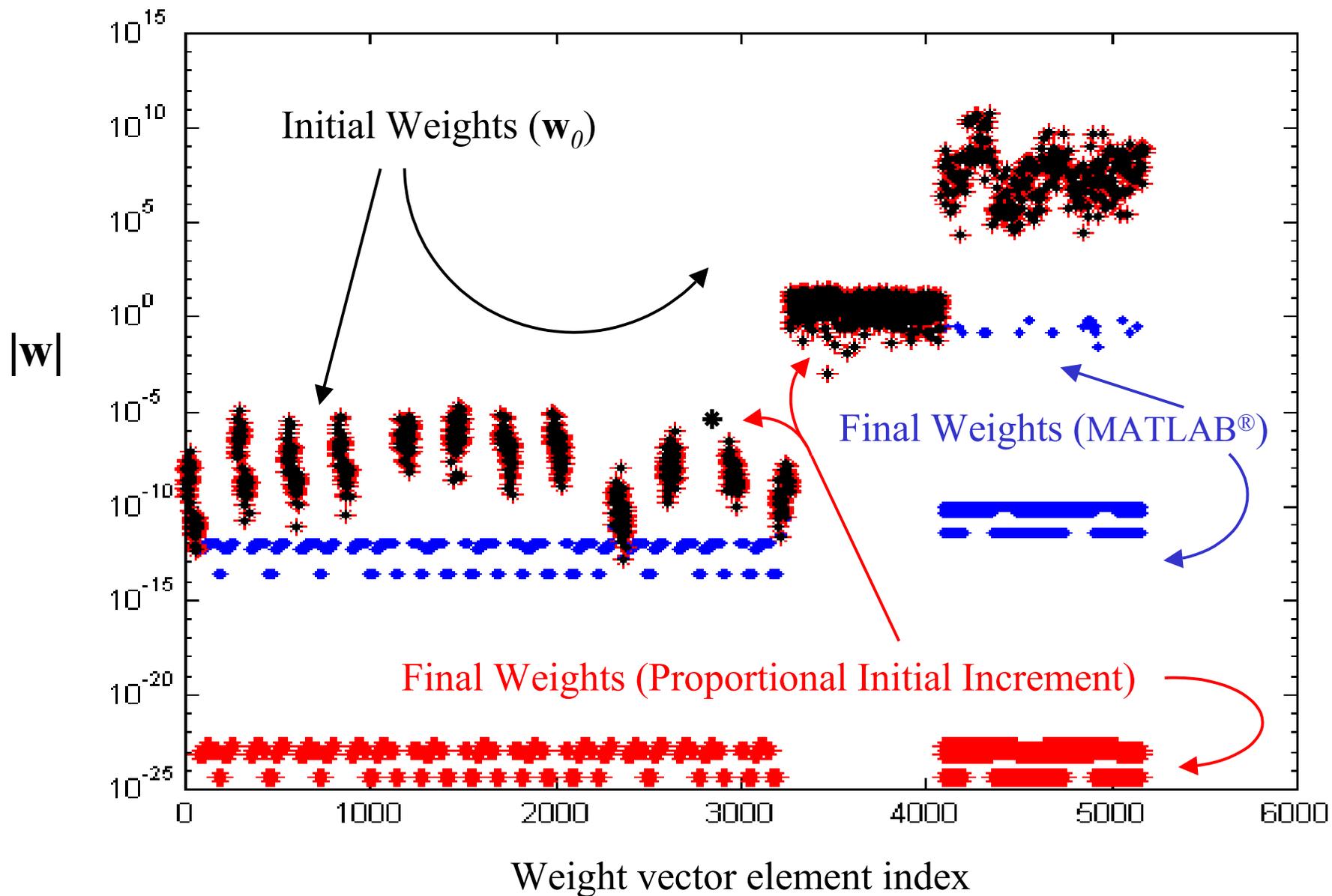


Backtracking, with local minima

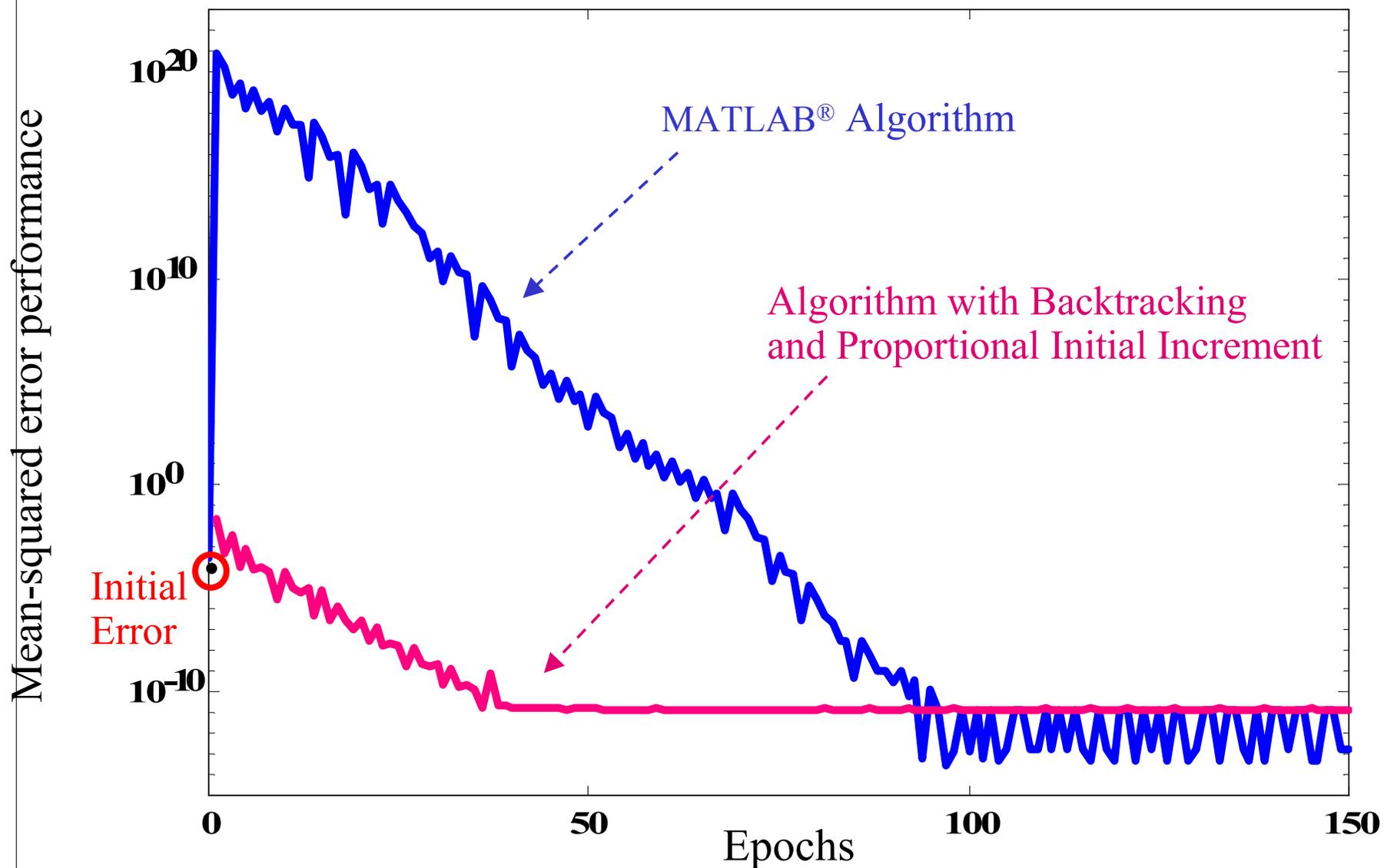
$$\Delta \mathbf{w}_0 = f_1 |\mathbf{w}_0| + f_2$$

Proportional initial increment

# Performance Comparison: Trained Weights Distribution w.r.t. Initial Weights Distribution

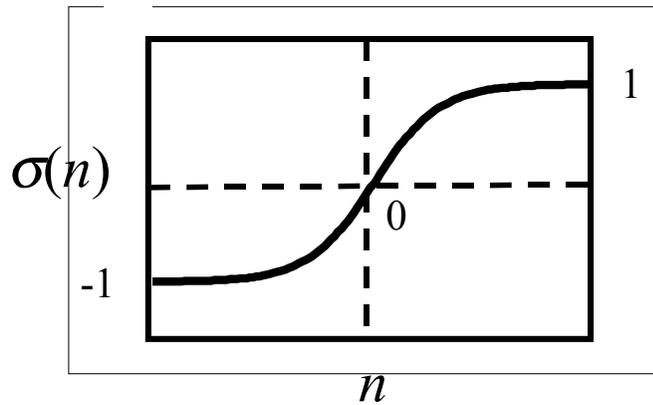


# Performance Comparison: Action Network Training at $t = 0.2$ s



# Resilient Backpropagation: Actual On-line Implementation

- Estimate the **sign of the gradient**, without computing  $\frac{\partial E}{\partial w}$  :



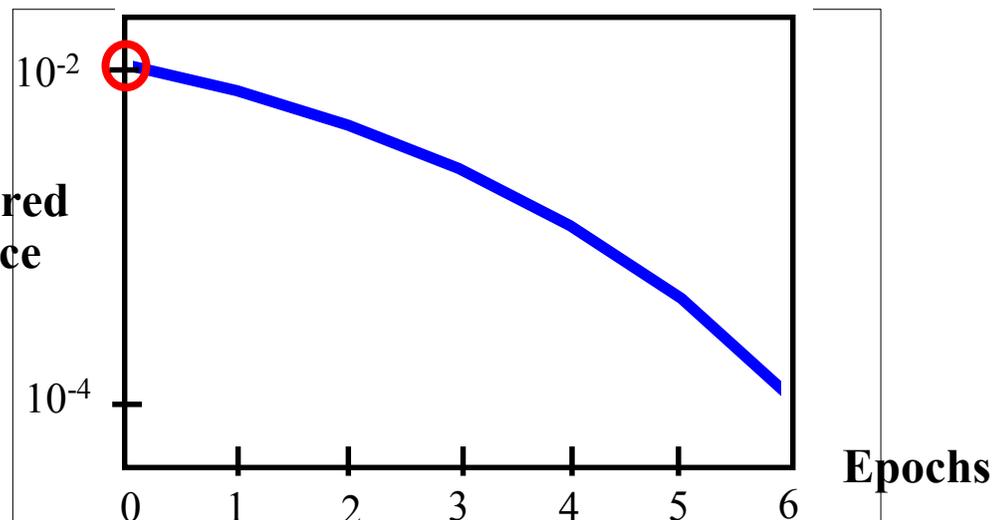
$$\begin{cases} \text{sgn}[\sigma(n)] = \text{sgn}[n] \\ \text{sgn}[\sigma'(n)] > 0, \forall n \end{cases}$$



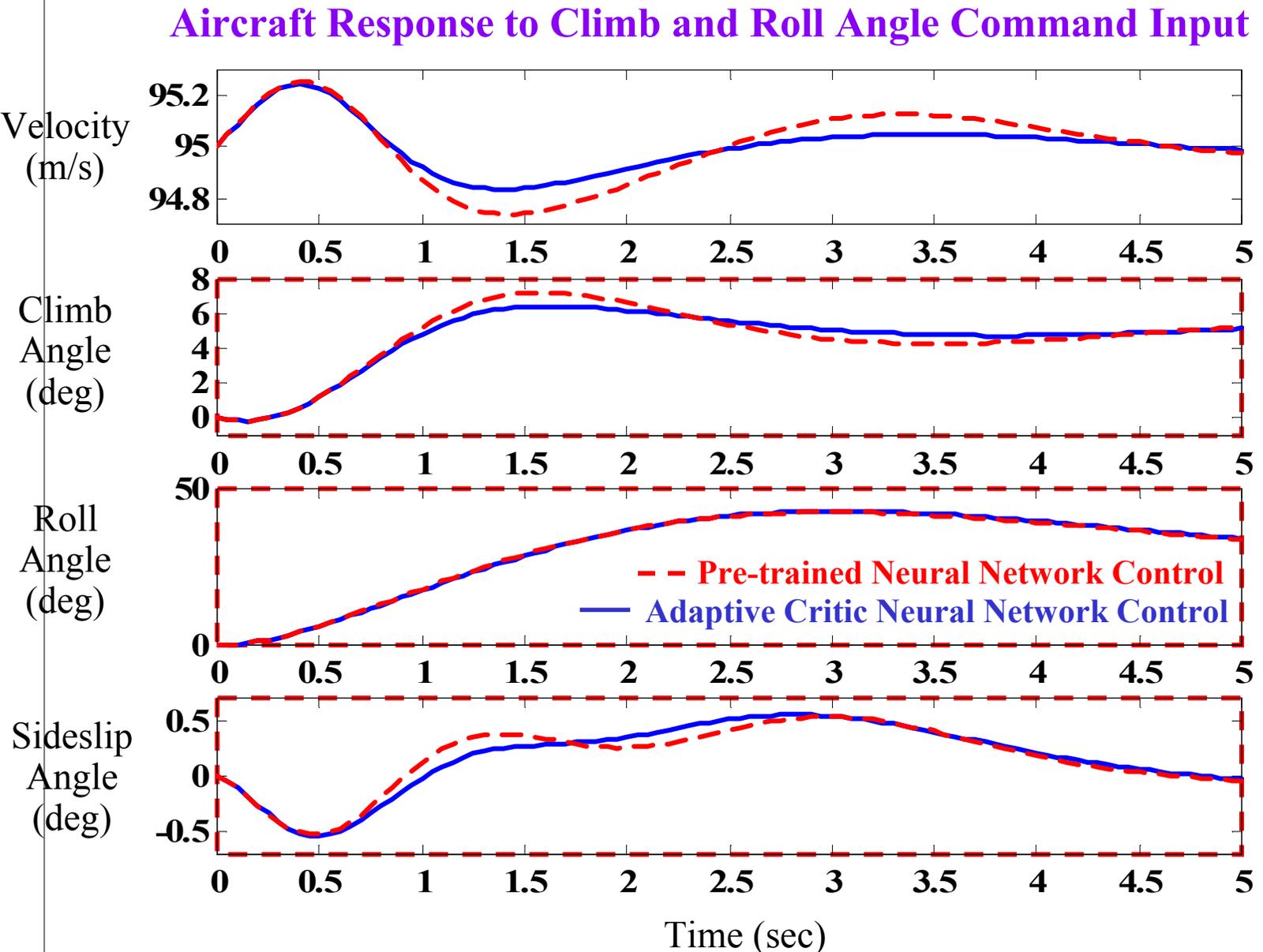
**Speed increase:  $O(200)$**

- Only **few epochs** are used to minimize  $E$ , at time  $t$ :

**Actual mean-squared error performance**



# Comparison of Adaptive Critic Controller with Pre-trained Neural Network Controller, at Flight Conditions $(H_0, V_0) = (2 \text{ Km}, 95 \text{ m/s})$



## Summary and Conclusions

- Adaptive critic flight control design:
  - ❖ Algebraic pre-training based on available linear control knowledge
  - ❖ On-line training during simulations (full envelope, severe conditions)

Objective: improve aircraft control performance under extreme conditions

- Achievements:
  - Systematic approach for designing nonlinear control systems
  - Innovative neural network (off-line and on-line) training techniques
- Results: improved performance during large angle maneuvers

### Future Work:

Continue testing: acrobatic maneuvers, severe operating conditions, coupling and nonlinear effects!