Lecture – 37

Neuro-Adaptive Design – II:
A Robustifying Tool for Any Design

Dr. Radhakant Padhi
Asst. Professor
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore
Motivation

- Perfect system modeling is difficult
- Sources of imperfection
  - Unmodelled dynamics (missing algebraic terms in the model)
  - Inaccurate knowledge of system parameters
  - Change of system parameters/dynamics during operation
- “Black box” adaptive approaches exist. But, making use of existing design is better!
  (faster adaptation, chance of instability before adaptation is minimal)
- The adaptive design should preferably be compatible with “any nominal control” design
Reference

Modeling Inaccuracy: A Simple Example

\[ \dot{x} = 2\sin(x) + 0.1\sin(x) \]

- Known part of actual system (nominal system)
- Unknown part of actual system

\[ = 2\sin(x) + \Delta c \sin(x) \]
- Weight
- Basis Function
Objective:

To increase the robustness of a “nominal controller” with respect to parameter and/or modeling inaccuracies, which lead to imperfections in the system model.
Problem Description and Strategy

- Desired Dynamics: \( \dot{X}_d = f(X_d, U_d) \)
- Actual Plant: \( \dot{X} = f(X, U) + d(X) \) (unknown)
- Goal: \[ X \rightarrow X_d, \quad \text{as} \quad t \rightarrow \infty \]
- Approximate System: \( \dot{X}_a = f(X, U) + \hat{d}(X) + K_a (X - X_a) \) \( (K_a > 0) \)
- Strategy: \[ X \rightarrow X_a \rightarrow X_d, \quad \text{as} \quad t \rightarrow \infty \]
Steps for assuring $X_a \rightarrow X_d$

- Select a gain matrix $K > 0$ such that
  \[
  \dot{E}_d + K E_d = 0, \quad E_d \triangleq (X_a - X_d)
  \]

- This leads to
  \[
  \{ f(X,U) + \hat{d}(X) + K_a(X - X_a) \} - f(X_d,U_d) + K(X_a - X_d) = 0
  \]

- Solve for the control $U$

\[
\begin{align*}
  f(X,U) & = \left\{ f(X_d,U_d) - \hat{d}(X) - K_a(X - X_a) - K(X_a - X_d) \right\} \\
  i.e. \quad f(X,U) & = h(X,X,a,X_d,U_d)
\end{align*}
\]
Control Solution:
(No. of controls = No. of states)

- **Affine Systems:**
  \[ f(X) + [g(X)]U = h(X, X_d, X_a, U_d) \]

  \[
  U = [g(X)]^{-1} \{ h(X, X_d, X_a, U_d) - f(X) \}
  \]

- **Non-affine Systems:**
  \[ f(X, U) = h(X, X_d, X_a, U_d) \]

  Use Numerical Method
  (e.g. N-R Technique)

  \[
  \left( U_{\text{guess}} \right)_k = \begin{cases} 
  U_d & : \ k = 1 \\
  U_{k-1} & : \ k = 2, 3, \ldots 
  \end{cases}
  \]
Control Solution:
(No. of controls < No. of states)

- Modify $X_a$ dynamics:

$$\dot{X}_a = f(X,U) + \left[ \hat{d}(X) - \Psi(X)U_s \right] + \Psi(X)U_s + K_a (X - X_a)$$

$$= f(X,U) + \hat{d}_a(X) + \Psi(X)U_s + K_a (X - X_a)$$

- Solve for the control from:

$$\left\{ f(X,U) + \hat{d}_a(X) + \Psi(X)U_s + K_a (X - X_a) \right\} - f(X_d,U_d) + K(X_a - X_d) = 0$$

$$f(X,U) + \Psi(X)U_s = \left\{ f(X_d,U_d) - \hat{d}_a(X) - K_a (X - X_a) - K(X_a - X_d) \right\}$$

$$= h(X,X_a,X_d,U_d)$$
Solution for affine systems: 
(No. of controls < No. of states)

\[
\{f(X) + g(X)U\} + \Psi(X)U_s = h(X, X_a, X_d, U_d)
\]

\[
f(X) + \begin{bmatrix} g(X) & \Psi(X) \end{bmatrix} \begin{bmatrix} U \\ U_s \end{bmatrix} = h(X, X_a, X_d, U_d)
\]

\[
G(X) \ V = -f(X) + h(X, X_a, X_d, U_d)
\]

\[
V = \left[G(X)\right]^{-1} \{ -f(X) + h(X, X_a, X_d, U_d) \}
\]

Extract $U$ from $V$

For simplicity, we will not consider this special case in our further discussion.
Steps for assuring $X \rightarrow X_a$

- **Error:** $E_a \triangleq (X - X_a), \quad e_a \triangleq (x_i - x_{a_i})$

- **Error Dynamics:**

  \[
  \dot{x}_i = f_i (X, U) + d_i (X) \\
  \dot{x}_{a_i} = f_i (X, U) + \hat{d}_i (X) + k_{a_i} e_{a_i} \\
  \dot{e}_{a_i} = \dot{x}_i - \dot{x}_{a_i} \\
  = \left[ d_i (X) - \hat{d}_i (X) \right] - k_{a_i} e_{a_i} \\
  = \left\{ W_i^T \Phi_i (X) + \varepsilon_i \right\} - \hat{W}_i^T \Phi_i (X) - k_{a_i} e_{a_i} \\
  \dot{e}_{a_i} = \tilde{W}_i^T \Phi_i (X) + \varepsilon_i - k_{a_i} e_{a_i} \\
  \quad \left( \tilde{W}_i \triangleq W_i - \hat{W}_i \right)
  \]

Ideal neural network

\[d_i (X) = W_i^T \varphi_i (X) + \varepsilon_i\]

Actual neural network

\[\hat{d}_i (X) = \hat{W}_i^T \varphi_i (X)\]
Stable Function Learning

Lyapunov Function Candidate

\[ L_i = \frac{1}{2} (p_i e_i^2) + \frac{1}{2\gamma_i} (\tilde{W}_i^T \tilde{W}_i) \quad (p_i, \gamma_i > 0) \]

Derivative of Lyapunov Function

\[ \dot{L}_i = p_i e_i \dot{e}_i + \frac{1}{\gamma_i} \tilde{W}_i^T \dot{\tilde{W}}_i \]
\[ = p_i e_i \left( \tilde{W}_i^T \Phi_i (X) + \varepsilon_i - k_{ai} e_{ai} \right) - \frac{1}{\gamma_i} \tilde{W}_i^T \dot{\tilde{W}}_i \quad (\because \tilde{W}_i \triangleq W_i - \hat{W}_i) \]
\[ = \tilde{W}_i^T \left( p_i e_{ai} \Phi_i (X) - \frac{1}{\gamma_i} \dot{\tilde{W}}_i \right) + p_i e_{ai} \varepsilon_i - k_{ai} p_i e_{ai}^2 \]
\[ \Rightarrow 0 \]
Stable Function Learning

Weight update rule (Neural network training)

\[ \dot{W}_i = \gamma_i p_i e_{a_i} \Phi_i (X) \]

Derivative of Lyapunov Function

\[ \dot{L}_i = p_i e_{a_i} \varepsilon_i - k_{a_i} p_i e_{a_i}^2 \]

\[ \dot{L}_i < 0 \text{ if } |e_{a_i}| > \left( \frac{\varepsilon_i}{k_{a_i}} \right) \]

The system is “Practically Stable”
Neuro-adaptive Design: Implementation of Controller

Weight update rule:

\[ \dot{\hat{W}}_i = \gamma_i p_i e_{a_i} \Phi_i, \quad \hat{W}_i (0) = 0 \]

where, \( \gamma_i \): Learning rate

\[ e_{a_i} = x_i - x_{a_i} \]

\( \Phi_i \): Basis function

Estimation of unknown function:

\[ \hat{d}(X) = \hat{W}^T \Phi_i \]
**Neuro-adaptive Design: Implementation of Controller**

- **Desired Dynamics:**
  \[ \dot{X}_d = f \left( X_d, U_d \right) \]

- **Actual Plant:**
  \[ \dot{X} = f(X, U) + d(X) \]

  *In reality, \( X(t) \) should be available from sensors and filters!*

- **Approximate System:**
  \[ \dot{X}_a = f(X, U) + \hat{d}(X) + K_a \left( X - X_a \right), \quad K_a > 0 \text{ (pdf)} \]

  *NN Approximation*

- **Initial Condition:**
  \[ X_d(0) = X_a(0) = X(0) : \text{known} \]
Example – 1

A Motivating Scalar Problem

Dr. Radhakant Padhi
Asst. Professor
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore
Example – 1: A scalar problem

- System dynamics (nominal system)
  \[ \dot{x}_d = \left( x_d + x_d^2 \right) + \left( 1 + x_d^2 \right) u_d \]

- System dynamics (actual system)
  \[ \dot{x} = \left( x + x^2 \right) + \left( 1 + x^2 \right) u + d(x) \]
  \[ d(x) = \sin(\pi x / 2) \]
  (unknown for control design)

- Problem objectives:
  * Nominal control design: \( x_d \to 0 \)
  * Adaptive control design: \( x \to x_d \)

Note: The objective \( x \to x_d \) should be achieved much faster than \( x_d \to 0 \)
Example – 1: A scalar problem

- Nominal control (dynamic inversion)
  \[
  (\dot{x}_d - 0) + \left(\frac{1}{\tau_d}\right) (x_d - 0) = 0 \quad (\tau_d = 1)
  \]

- Nominal control
  \[
  u_d = -\left(1 + x_d^2\right)^{-1} \left( x_d + x_d^2 + x_d \right)
  \]

- Adaptive control
  \[
  u = \frac{1}{1+x^2} \left[ \left( x_d + x_d^2 \right) + \left(1 + x_d^2\right) u_d - k(x_a - x_d) \right]
  \]

- Design parameters
  \[
  k = 2.5 \quad k_a = 1 \quad p = 1 \quad \gamma = 30
  \]

\[
\Phi(x) = \left[ \frac{x}{x_0} \quad \left(\frac{x}{x_0}\right)^2 \quad \left(\frac{x}{x_0}\right)^3 \right]^T
\]
Example – 1: A scalar problem

State Trajectory

Control Trajectory

ADVANCED CONTROL SYSTEM DESIGN
Dr. Radhakant Padhi, AE Dept., IISc-Bangalore
Example – 1:
A scalar problem

Approximation of the unknown function
Example – 2

Double Inverted Pendulum: A Benchmark Problem

Dr. Radhakant Padhi
Asst. Professor
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore
Example – 2:
Double inverted pendulum
Example – 2: Double inverted pendulum

- **Nominal Plant:**
  \[
  \begin{align*}
  \dot{x}_1^1 &= x_1^2 \\
  \dot{x}_1^2 &= \alpha_1 \sin(x_1^1) + \frac{kr}{2J_1}(l - b) + \left(\frac{u_{1_{\text{max}}}}{J_1}\right) \tanh(u_1) + \left(\frac{kr^2}{4J_1}\right) \sin(x_2^1) \\
  \dot{x}_2^1 &= x_2^2 \\
  \dot{x}_2^2 &= \alpha_2 \sin(x_2^1) + \frac{kr}{2J_2}(l - b) + \left(\frac{u_{2_{\text{max}}}}{J_2}\right) \tanh(u_2) + \left(\frac{kr^2}{4J_2}\right) \sin(x_1^1)
  \end{align*}
  \]

- **Actual Plant:**
  \[
  \begin{align*}
  \dot{x}_1^1 &= x_1^2 \\
  \dot{x}_1^2 &= (\alpha_1 + \Delta \alpha_1) \sin(x_1^1) + \frac{kr}{2J_1}(l - b) + \frac{u_{1_{\text{max}}}}{J_1} \tanh(u_1) + \frac{kr^2}{4J_1} \sin(x_2^1) + K_{m_1} e^{a_{1,1} x_1^1} \\
  \dot{x}_2^1 &= x_2^2 \\
  \dot{x}_2^2 &= (\alpha_2 + \Delta \alpha_2) \sin(x_2^1) + \frac{kr}{2J_2}(l - b) + \frac{u_{2_{\text{max}}}}{J_2} \tanh(u_2) + \frac{kr^2}{4J_2} \sin(x_1^1) + K_{m_2} e^{a_{2,2} x_2^1}
  \end{align*}
  \]
Example – 2: Double inverted pendulum

\[
\alpha_i \triangleq \left( \frac{m_i gr}{J_i} - \frac{kr^2}{4J_i} \right)
\]

\[
\beta_i \triangleq \frac{kr}{2J_i} (l - b)
\]

\[
\eta_i \triangleq \frac{u_{i_{\max}}}{J_i}
\]

\[
\sigma_i \triangleq \frac{kr^2}{4J_i}
\]

<table>
<thead>
<tr>
<th>System Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>End mass of pendulum 1 ( (m_1) )</td>
<td>2</td>
<td>kg</td>
</tr>
<tr>
<td>End mass of pendulum 2 ( (m_2) )</td>
<td>2.5</td>
<td>kg</td>
</tr>
<tr>
<td>Moment of inertia ( (J_1) )</td>
<td>0.5</td>
<td>kg ( m^2 )</td>
</tr>
<tr>
<td>Moment of inertia ( (J_2) )</td>
<td>0.625</td>
<td>kg ( m^2 )</td>
</tr>
<tr>
<td>Spring constant of connecting spring ( (k) )</td>
<td>100</td>
<td>N/m</td>
</tr>
<tr>
<td>Pendulum height ( (r) )</td>
<td>0.5</td>
<td>( m )</td>
</tr>
<tr>
<td>Natural length of spring ( (l) )</td>
<td>0.5</td>
<td>( m )</td>
</tr>
<tr>
<td>Gravitational acceleration ( (g) )</td>
<td>9.81</td>
<td>( m/s^2 )</td>
</tr>
<tr>
<td>Distance between pendulum hinges ( (b) )</td>
<td>0.4</td>
<td>( m )</td>
</tr>
<tr>
<td>Maximum torque input ( (u_{1_{\max}}) )</td>
<td>20</td>
<td>Nm</td>
</tr>
<tr>
<td>Maximum torque input ( (u_{2_{\max}}) )</td>
<td>20</td>
<td>Nm</td>
</tr>
</tbody>
</table>
Example – 2: Double inverted pendulum

- Parameters in unknown function
  \( \Delta \alpha_1, \Delta \alpha_2 : 20\% \) off \( a_1 = a_2 = 0.01 \) \( K_{m_1} = K_{m_2} = 0.1 \)

- Control design parameters

\[
K = 0.2 \; I_4, \quad K_a = I_4
\]

\[
\psi(X) = \begin{bmatrix}
-10 & 10 & 0 & 0 \\
10 & -10 & 10 & -10
\end{bmatrix}^T
\]

\[
p_2 = p_4 = 1
\]

\[
\gamma_2 = \gamma_4 = 20
\]

\[
\Phi_2(X) = \begin{bmatrix}
1 & x_1 /1! & \cdots & x_1^{17} /17! & 1 & x_2 /1! & \cdots & x_2^{17} /17!
\end{bmatrix}^T
\]

\[
\Phi_4(X) = \begin{bmatrix}
1 & x_3 /1! & \cdots & x_3^{17} /17! & 1 & x_4 /1! & \cdots & x_4^{17} /17!
\end{bmatrix}^T
\]
Example – 2: Double inverted pendulum

Position of Mass – 1

Velocity of Mass – 1

- Desired trajectory
- Plant trajectory with nominal control
- Plant trajectory with adaptive control
Example – 2: Double inverted pendulum

Position of Mass – 2

Velocity of Mass – 2
Example – 2:  
Double inverted pendulum

Torque for Mass – 1

Torque for Mass – 2

![Graphs showing control efforts for double inverted pendulum.](image-url)
Example – 2: Double inverted pendulum

Capturing $d_2(X)$

Capturing $d_4(X)$
Neuro-Adaptive Design for Output Robustness

Dr. Radhakant Padhi
Asst. Professor
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore
N-A for Robustness of Output Dynamics

Desired output dynamics:

\[ \dot{Y}_d = f_{Y_d}(X_d) + G_{Y_d}(X_d)U_d \]

Actual output dynamics:

\[ \dot{Y} = f_{Y_d}(X) + G_{Y_d}(X)U + d(X) \]

Objective: \( Y \to Y_d \) as soon as possible
N-A for Robustness of Output Dynamics

- Dynamics of auxiliary output:

\[
\dot{Y}_a = f_{Y_a}(X) + G_{Y_a}(X)U + \hat{d}(X) + K_a(Y - Y_a)
\]

- Strategy:

Approximate state

\[
Y_a
\]

Actual state

\[
Y
\]

Desired state

\[
Y_d
\]
Steps for assuring \( Y_a \rightarrow Y_d \)

- Enforce the error dynamics
  \[
  \dot{E}_d + KE_d = 0 \quad E_d \triangleq (Y_a - Y_d)
  \]
- After carrying out the necessary algebra
  \[
  f(X, U) = h(X, X_a, X_d, U_d)
  \]
- In case of control affine system
  \[
  f(X) + [g(X)]U = h(X, X_d, X_a, U_d)
  \]
- The control is given by
  \[
  U = [g(X)]^{-1}\{h(X, X_d, X_a, U_d) - f(X)\}
Steps for assuring $Y \rightarrow Y_a$

- The error in the output is defined as
  
  $$E_a \triangleq (Y - Y_a) \quad e_{a_i} \triangleq (y_i - y_{ai})$$

- Ideal neural network is given by:
  
  $$d_i(X) = W_i^T \varphi_i(X) + \varepsilon_i$$

  where $W_i$ is the weight matrix and $\varphi_i(X)$ is the radial basis function
Function Learning:

Define error

\[ e_{a_i} \overset{\Delta}{=} (y_i - y_{a_i}) \]

Output dynamics

\[
\dot{y}_i = f_{Y_i}(X) + g_{Y_i}(X)U + d_i(X) \\
\dot{y}_{a_i} = f_{Y_i}(X) + g_{Y_i}(X)U + \hat{d}_i(X) + k_{a_i} e_{a_i}
\]

From universal function approximation property

\[
d_i(X) = W_i^T \varphi_i(X) + \varepsilon_i \\
\hat{d}_i(X) = \hat{W}_i^T \varphi_i(X)
\]

Error dynamics

\[
\dot{e}_{a_i} = d_i(X) - \hat{d}_i(X) - k_{a_i} e_{a_i} \\
\dot{e}_{a_i} = \tilde{W}_i^T \Phi_i(X) + \varepsilon_i - k_{a_i} e_{a_i}
\]
Lyapunov Stability Analysis

Lyapunov Function Candidate:

\[
L_i = \frac{1}{2} (e_{ai} p_i e_{ai}) + \frac{1}{2} (\tilde{W}_i^T \gamma_i \tilde{W}_i)
\]

Derivative of Lyapunov Function:

\[
\dot{L}_i = e_{ai} p_i \dot{e}_{ai} + \tilde{W}_i^T \gamma_i \dot{\tilde{W}}_i
\]

\[
= e_{ai} p_i \left[ \tilde{W}_i^T \Phi_i(X) + \varepsilon_i - k_{ai} e_{ai} \right] - \tilde{W}_i^T \gamma_i \dot{\tilde{W}}_i
\]

\[
= \tilde{W}_i^T \left[ e_{ai} p_i \Phi_i(X) - \gamma_i^{-1} \dot{\tilde{W}}_i \right] + e_{ai} p_i \varepsilon_i - k_{ai} e_{ai}^2 p_i
\]

Weight Update Rule:

\[
\dot{\tilde{W}}_i = \gamma_i e_{ai} p_i \Phi_i(X,X_d)
\]
Lyapunov Stability Analysis

This condition leads to

\[
\dot{L}_i = e_{a_i} p_i \varepsilon_i - k_{a_i} e_{a_i}^2 p_i
\]

Whenever

\[
\dot{L}_i < 0 \quad \text{whenever} \quad |e_{a_i}| > \frac{|\varepsilon_i|}{k_{a_i}}
\]

Using the Lyapunov stability theory, we conclude that the trajectory of \( e_{a_i} \) and \( \tilde{W}_i \) are pulled towards the origin.

Hence, the output dynamics is “Practically Stable”!
Neuro-Adaptive Design with Modified Weight Update Rule (with \( \sigma \) modification)

Dr. Radhakant Padhi
Asst. Professor
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore
Neuro-adaptive Design with $\sigma$ Modification

Weight update rule:

$$\dot{\hat{W}}_i = \gamma_i e_{a_i} \Phi_i - \gamma_i \sigma_i \hat{W}_i, \quad \hat{W}_i (0) = 0$$

where, $\gamma_i$: Learning rate, $\sigma_i > 0$: Stabilizing factor

$$e_{a_i} = x_i - x_{a_i}$$

$\Phi_i$: Basis function

Estimation of unknown function:

$$\hat{d}(X) = \hat{W}^T \Phi_i$$
Neuro-adaptive Design with $\sigma$ Modification

Lyapunov function candidate:

$$v_i = \frac{1}{2}e_{a_i}^2 + \frac{1}{2}\tilde{W}_i^T\gamma_i^{-1}\tilde{W}_i$$

(Note: $p_i = 1$)

Then

$$\dot{v}_i = e_{a_i}\dot{e}_{a_i} + \tilde{W}_i^T\gamma_i^{-1}\dot{\tilde{W}}_i$$

$$= (e_{a_i}e_i - e_{a_i}^2) + \sigma_i\tilde{W}_i^T\dot{\hat{W}}_i$$

(can be derived so, with $k_{a_i} = 1$)

Consider the last term in $\dot{v}_i$

$$\tilde{W}_i^T\hat{W}_i = \frac{1}{2} \times 2(\tilde{W}_i^T\hat{W}_i)$$

$$= \frac{1}{2} \times 2\tilde{W}_i^T(W_i - \tilde{W}_i) = \frac{1}{2}(2\tilde{W}_i^TW_i - 2\tilde{W}_i^T\tilde{W}_i)$$
Neuro-adaptive Design with $\sigma$ Modification

However, $2\tilde{W}_i^T W_i = \tilde{W}_i^T W_i + \tilde{W}_i^T \hat{W}_i$

$$= \tilde{W}_i^T (\hat{W}_i + \tilde{W}_i) + (W_i - \hat{W}_i)^T W_i$$

$$= \tilde{W}_i^T \hat{W}_i + \tilde{W}_i^T \tilde{W}_i + W_i^T W_i - \hat{W}_i^T W_i$$

$$= \hat{W}_i^T (\tilde{W}_i - W_i) + \tilde{W}_i^T \tilde{W}_i + W_i^T W_i$$

$$= -\tilde{W}_i^T \hat{W}_i + \tilde{W}_i^T \tilde{W}_i + W_i^T W_i$$

Hence, $\sigma\tilde{W}_i^T \hat{W}_i = \sigma \frac{1}{2} (-\tilde{W}_i^T \hat{W}_i + \tilde{W}_i^T \tilde{W}_i + W_i^T W_i - \tilde{W}_i^T \tilde{W}_i - \tilde{W}_i^T \tilde{W}_i)$

$$= \frac{1}{2} (-\sigma \tilde{W}_i^T \hat{W}_i - \sigma \tilde{W}_i^T \tilde{W}_i + \sigma W_i^T W_i)$$

$$\leq \frac{1}{2} \left( -\sigma \|\tilde{W}_i\|^2 - \sigma \|\hat{W}_i\|^2 + \sigma \|W_i\|^2 \right)$$
Neuro-adaptive Design with $\sigma$ Modification

Hence, the equation for $\dot{v}_i$ becomes

$$\dot{v}_i \leq e_{a_i} e_i - e_{a_i}^2 - \frac{1}{2} \sigma_i \|\tilde{W}_i\|^2 - \frac{1}{2} \sigma_i \|\hat{W}_i\|^2 + \frac{1}{2} \sigma_i \|W_i\|^2$$

$$\leq \frac{e_{a_i}^2}{2} + \frac{e_i^2}{2} - e_{a_i}^2 - \frac{1}{2} \sigma_i \|\tilde{W}_i\|^2 - \frac{1}{2} \sigma_i \|\hat{W}_i\|^2 + \frac{1}{2} \sigma_i \|W_i\|^2$$

$$\leq -\frac{e_{a_i}^2}{2} + \left(\frac{e_i^2}{2} + \frac{1}{2} \sigma_i \|W_i\|^2 - \frac{1}{2} \sigma_i \|\tilde{W}_i\|^2 - \frac{1}{2} \sigma_i \|\hat{W}_i\|^2\right)$$
Neuro-adaptive Design with $\sigma$ Modification

Defining
\[ \beta_i \triangleq \frac{\varepsilon_i^2}{2} + \frac{1}{2} \sigma_i \left( \| W_i \|^2 - \| \tilde{W}_i \|^2 - \| \hat{W}_i \|^2 \right) \]

We have
\[ \dot{\nu}_i < 0, \quad \text{whenever} \quad \frac{e_{a_i}^2}{2} > \beta_i \]

i.e.
\[ \dot{\nu}_i < 0, \quad \text{whenever} \quad |e_{a_i}| > \sqrt{2\beta_i} \]
Summary: Neuro-Adaptive Design

- N-A Design: A generic model-following adaptive design for robustifying “any” nominal controller
- It is valid for both non-square and non-affine problems in general
- Extensions:
  - Robustness of output dynamics only
  - “Structured N-A design” for efficient learning
References


Thanks for the Attention...!