NARMA-L2 Controller

**NARMA discrete-time model:**

\[
y(k + d) = N[y(k), y(k - 1), ..., y(k - n + 1), u(k), u(k - 1), ..., u(k - n + 1)]
\]

Non Linear  Auto-Regressive  Moving Average

Narendra e Mukhopahyay, 1997: NARMA L2 Norm Model

\[
y(k + d) = f[y(k), y(k - 1), ..., y(k - n + 1), u(k - 1), ..., u(k - m + 1)]
+ g[y(k), y(k - 1), ..., y(k - n + 1), u(k - 1), ..., u(k - m + 1)] \cdot u(k)
\]

**Control Law:**

\[
u(k) = y_r(k + d) - f[y(k), y(k - 1), ..., y(k - n + 1), u(k - 1), ..., u(k - n + 1)]
\]

\[
g[y(k), y(k - 1), ..., y(k - n + 1), u(k - 1), ..., u(k - n + 1)]
\]

**Applying the Control Law to the NARMA-LW model results:**

\[
y_r(k + d) \equiv y(k + d)
\]

\[
y follows y_r exactly!!
\]
NARMA-L2 Controller

\[ u(k) = y_r(k + d) - f[y(k), y(k - 1), ..., y(k - n + 1), u(k - 1), ..., u(k - n + 1)] \\
g[y(k), y(k - 1), ..., y(k - n + 1), u(k - 1), ..., u(k - n + 1)] \]

ANN can be used to learn \( f(.) \): the additive non-linear term and \( g(.) \) the multiplicative non-linear term

\[ y(k + d) = N[y(k), y(k - 1), ..., y(k - n + 1), u(k), u(k - 1), ..., u(k - n + 1)] \]
NARMA-L2 controller

Switch normalisation

\[ f(.) \]

\[ g(.) \]
Ex: Liquid Level Control

\[ A \frac{dh_1}{dt} = q_i - k_{12} \sqrt{h_1 - h_2} \]
\[ A \frac{dh_2}{dt} = k_{12} \sqrt{h_1 - h_2} - k_2 \sqrt{h_2} \]
Integrated Interface to Process

State Space Controller – Designed for a specific Operating Point
State-Space Results

- State-Space Results

![Graph 1](image1)

![Graph 2](image2)

![Graph 3](image3)

![Graph 4](image4)
Purpose: compare PI and NARMA-L2 control.

Because of the square root in the model a different behaviour is expect for different reference levels!
Generated Training Sets

Cover the time domain range (dynamics) and amplitude range (operating points).
NARMA-L2 tool

Plant Identification - NARMA-L2

- **Network Architecture**
  - Size of Hidden Layer: 20
  - No. Delayed Plant Inputs: 3
  - Sampling Interval (sec): 1
  - No. Delayed Plant Outputs: 3

- **Training Data**
  - Training Samples: 30000
  - Maximum Plant Input: 90
  - Minimum Plant Input: 0
  - Maximum Interval Value (sec): 250
  - Minimum Interval Value (sec): 1

- **Simulink Plant Model**: Browse

- **Generate Training Data**

- **Import Data**

- **Export Data**

- **Training Parameters**
  - Training Epochs: 500
  - Training Function: trainlm

- **Use Current Weights**

- **Use Validation Data**

- **Use Testing Data**

- **Train Network**

- **OK**

- **Cancel**

- **Apply**

Generate or import data before training the neural network plant.

**Algorithms**
- **Training**: Levenberg-Marquardt (trainlm)
- **Performance**: Mean Squared Error (mse)
- **Derivative**: Default (defaultderiv)

**Progress**
- **Epoch**: 0
- **Time**: 0:00:36
- **Performance**: 1.40e-06
- **Gradient**: 0.00773
- **Mu**: 0.00100
- **Validation Checks**: 0

- **34 iterations**

- **500**

- **1.34e-06**

- **1.00e-10**

- **0.00**

- **0.00130**

- **1.00e-06**

- **1.00e+10**

- **0**

- **6**

**Plots**
- **Performance** (plotperform)
- **Training State** (plottrainstate)
- **Regression** (plotregression)

**Plot Interval**: 1 epochs
- PI gives no error in steady state, but with very different dynamics.
- NARMA-L2 has almost the same behaviour in all operating points.
PI vs NARMA-L2 Control Signal

- NARMA-L2 uses often the maximum available \( u \).
- PI calculate signals that are clamped by the saturation.
- NARMA-L2 tends to “chattering”

-NARMA-L2 design parameters:
  - sampling rate
  - input/output delays
  - training sets
  - training algorithms

NARMA-L2 can be trained with real data!!
Real processes are quite more complex than models!