Neural networks in design of iterative learning control for nonlinear systems

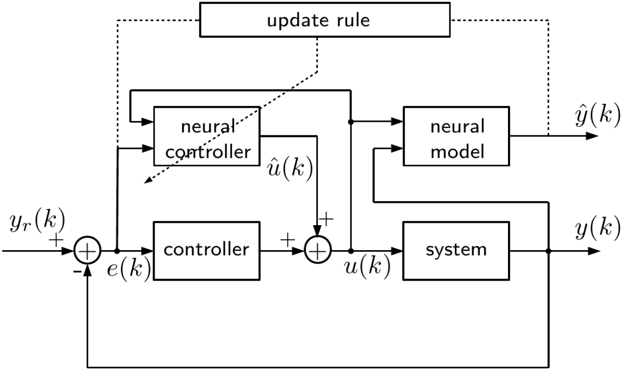
Krzysztof Patan, Maciej Patan
and Damian Kowalów

Institute of Control and Computation
Engineering
University of Zielona Góra, Poland

Introduction

- Iterative Learning Control – modern control strategy
- Neural networks – useful when dealing with nonlinear problems
- Purpose of the paper – to adopt artificial neural networks to develop iterative learning control
  1. to build an accurate neural model of the nonlinear plant based on the measurements from the previous trials
  2. to train the neural controller based on data provided by the model

General idea of neural network based ILC



Neural model

- Let consider a nonlinear system

$$y_p(k+1) = g(y_p(k), u_p(k), k), \ \ k = 0, \dots, N-1 \tag{1}$$

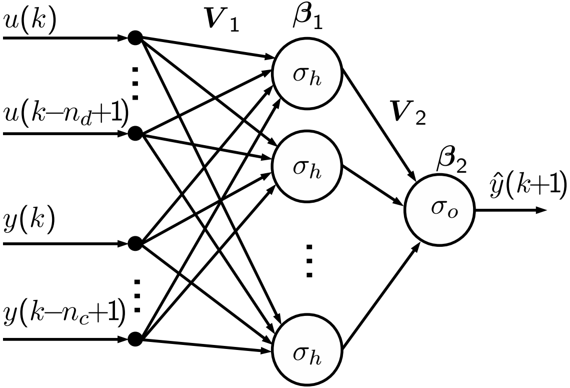
where $p \geq 0$ – the trial (cycle number)
N – a length of a trial
$u_p(k)$ – a system input along the p-th trial
g – a nonlinear function

- Function g can be realized using dynamic neural network

$$\hat{y}(k+1) = \hat{g}(\boldsymbol{\phi}(k)) = \sigma_o(\boldsymbol{V}_2^{\mathsf{T}} \sigma_h(\boldsymbol{V}_1^{\mathsf{T}} \boldsymbol{\phi}(k) + \boldsymbol{\beta}_1) + \boldsymbol{\beta}_2) \tag{2}$$

where $\boldsymbol{\phi}(k) = [y(k), \dots, y(k-n_c+1), u(k), \dots, u(k-n_d+1)]^{\mathsf{T}}$
$\boldsymbol{V}_1$, $\boldsymbol{V}_2$, $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ – weight matrices
σ_h and σ_o – activation functions

Structure of the neural network model



Neural controller

- The key idea – use the neural network to provide the realization of the function f (being implicitly an inverted model of the plant)
- Let consider the controller in the form:

$$\hat{u}_{p+1}(k+1) = \hat{f}(\boldsymbol{\varphi}_p(k)), \tag{3}$$

where $\hat{f}$ is a nonlinear function

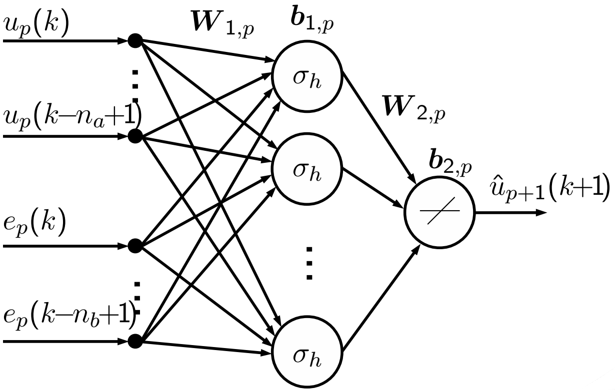
$$\boldsymbol{\varphi}_{p+1}(k) = [u_p(k), \ldots, u_p(k-n_a+1), e_p(k), \ldots, e_p(k-n_b+1)]^\mathsf{T}$$

- Let consider a neural network model with one hidden layer:

$$\hat{u}_{p+1}(k+1) = \hat{f}(\boldsymbol{\varphi}_p(k)) = \boldsymbol{W}_{2,p}^\mathsf{T}\sigma_h(\boldsymbol{W}_{1,p}^\mathsf{T}\boldsymbol{\varphi}_p(k) + \boldsymbol{b}_{1,p}) + \boldsymbol{b}_{2,p}, \tag{4}$$

where $\boldsymbol{W}_{1,p}$, $\boldsymbol{W}_{2,p}$, $\boldsymbol{b}_{1,p}$ and $\boldsymbol{b}_{2,p}$ are weight matrices
σ_h is the activation function

Structure of the neural network controller



Update rule

- After each trail the controller parameters are updated according to:

$$\theta_{p+1} = \theta_p + \Delta\theta_p, \tag{5}$$

where θ_p – the generalized network parameter
$\Delta\theta_p$ – a correction term

- Learning objective – at each trial p minimize the criterion

$$J = J_0 + \frac{1}{2}\lambda\sum_{i=1}^{M}\theta_i^2, \quad J_0 = \frac{1}{2}\sum_{k=1}^{N}(y_r(k) - y(k))^2 \tag{6}$$

where M – the number of the controller parameters
λ – a parameter governing how strongly large weights are penalized

- Using the gradient descent

$$\Delta\theta = -\eta\frac{\partial J_0}{\partial\theta} - \lambda\theta, \tag{7}$$

where η – the learning rate

- The gradient of the cost function J with respect to the parameter θ

$$\frac{\partial J_0}{\partial \theta} = -\sum_{k=1}^{N} e(k) \frac{\partial y(k)}{\partial \theta} = -\sum_{k=1}^{N} e(k) \frac{\partial y(k)}{\partial u(k-1)} \frac{\partial u(k-1)}{\partial \theta}. \quad (8)$$

- The first partial derivative in (8), due to the equivalence rule, can be calculated using the model of the system (2):

$$\frac{\partial y(k)}{\partial u(k-1)} \approx \frac{\partial \hat{y}(k)}{\partial u(k-1)} = \boldsymbol{V}_2^{\mathsf{T}} \left(\sigma_h' \circ \boldsymbol{V}_1^{u\mathsf{T}} \right) \quad (9)$$

where $\boldsymbol{V}_1^{u\mathsf{T}}$ is the weight vector associated with the input $u(k-1)$
$\circ$ – the Hadamard product (element-wise)

- The second derivative can be calculated using the estimate $\hat{u}(k-1)$
- To simplify the algorithm it is assumed that regressors' dependency on the neural network weights is ignored – the recursive pseudo-linear regression method

- for weights of the first layer $\boldsymbol{W}_1$:

$$\frac{\partial \hat{u}(k-1)}{\partial \boldsymbol{W}_1} = \boldsymbol{W}_2^{\mathsf{T}} \left(\sigma_h^{'} \circ \varphi(k-1) \right) \tag{10}$$

- for biases of the first layer $\boldsymbol{b}_1$:

$$\frac{\partial \hat{u}(k-1)}{\partial \boldsymbol{b}_1} = \boldsymbol{W}_2^{\mathsf{T}} \sigma_h^{'} \tag{11}$$

- for weights of the second layer $\boldsymbol{W}_2$:

$$\frac{\partial \hat{u}(k-1)}{\partial \boldsymbol{W}_2} = \sigma_h(\boldsymbol{W}_1^{\mathsf{T}} \boldsymbol{\varphi}(k) + \boldsymbol{b}_1) \tag{12}$$

- for biases of the second layer $\boldsymbol{b}_2$:

$$\frac{\partial \hat{u}(k-1)}{\partial \boldsymbol{b}_2} = \mathbf{1} \tag{13}$$

Neural network based ILC

Step 0. Initialization. Set the feedback controller parameters, set η and λ

Step 1. Design the neural network controller (3)

Step 2. Design the neural network model (2)

Step 3. Evaluate the control system and record the data

$$\{u(k), \hat{u}(k), y(k), \hat{y}(k), e(k)\}_{k=1}^{N}$$

Step 4. Controller parameters update

i) using the set $\{\hat{u}(k), e(k)\}_{k=1}^{N}$ calculate derivatives (10)-(13)

ii) using $\{\hat{y}(k)\}_{k=1}^{N}$ calculate the derivative (9)

iii) using $\{e(k)\}_{k=1}^{N}$ calculate the gradient (8)

iv) update the neural network controller parameters according to (7)

Step 5. Go to Step 3

Stability analysis

- Let $\boldsymbol{x}(k) = [u(k), \ldots, u(k-n_a+1)]^\mathsf{T}$ be a state vector of the neural controller
- Then the state of the neural controller

$$\boldsymbol{x}(k+1) = \boldsymbol{W}_2^{x\mathsf{T}} \sigma_h(\boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{x}(k) + \boldsymbol{W}_1^{e\mathsf{T}} \boldsymbol{e}(k) + \boldsymbol{b}_1) + \boldsymbol{b}_2 \tag{14}$$

where $\boldsymbol{W}_2^{x\mathsf{T}} = [\boldsymbol{W}_2^\mathsf{T} \ \boldsymbol{0}]^\mathsf{T}$, $\boldsymbol{e}(k) = [e(k), \ldots, e(k-n_b+1)]^\mathsf{T}$, $\boldsymbol{W}_1^{x\mathsf{T}}$ and $\boldsymbol{W}_1^{e\mathsf{T}}$ are weight matrices associated with vectors $\boldsymbol{x}(k)$ and $\boldsymbol{e}(k)$ respectively

- The system (14) can be transformed to the autonomous form

$$\boldsymbol{z}(k+1) = \boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{W}_2^{x\mathsf{T}} \bar{\sigma}(\boldsymbol{z}(k)), \tag{15}$$

where $\bar{\sigma}(\boldsymbol{z}(k)) = \sigma_h(\boldsymbol{z}(k) + \boldsymbol{v}^*) - \sigma_h(\boldsymbol{v}^*)$.

Theorem

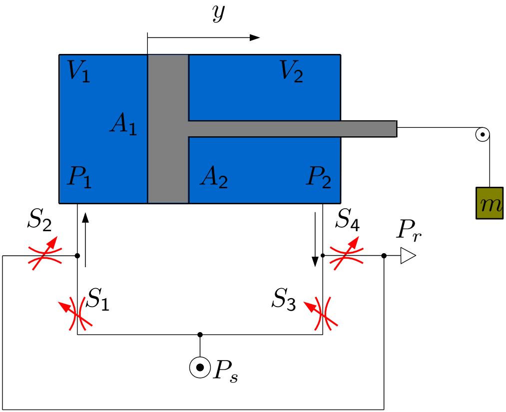
The neural system represented by (15) is globally asymptotically stable if the following condition is satisfied:

$$\|{\boldsymbol{W}_1^x}^{\mathsf{T}} {\boldsymbol{W}_2^x}^{\mathsf{T}}\| < 1 \tag{16}$$

Sketch of proof

- proof is based of the second method of Lyapunov
- candidate Lyapunov function – a norm of the state vector
- contraction mapping theorem is used

Illustrative example – pneumatic servomechanism



V_1, V_2 – cylinder volumes
A_1, A_2 – chamber areas
P_1, P_2 – chamber pressures
P_s – supplied pressure
P_r – exhaust pressure
m – load mass
y – piston position
$S_1, \ldots, S_4$ – operating valves
u – control signal

S_1 and S_4 are open for $u \geqslant 0$
S_2 and S_3 are open for $u < 0$

Modelling

- data recorded in the closed-loop control with the PI controller
- reference: random steps trigerred randomly with levels covering possible piston positions from the interval $(-0.245, 0.245)$
- neural model setting: number of delayed outputs and inputs $n_c = 4$, $n_d = 2$, number of hidden neurons $v_m = 5$, activation function of hidden neurons $\sigma_h = \tanh$, $sigma_o$ – linear function
- training process carried out for 100 epochs
- modelling quality – Sum of Squared Errors is SSE=0.0438

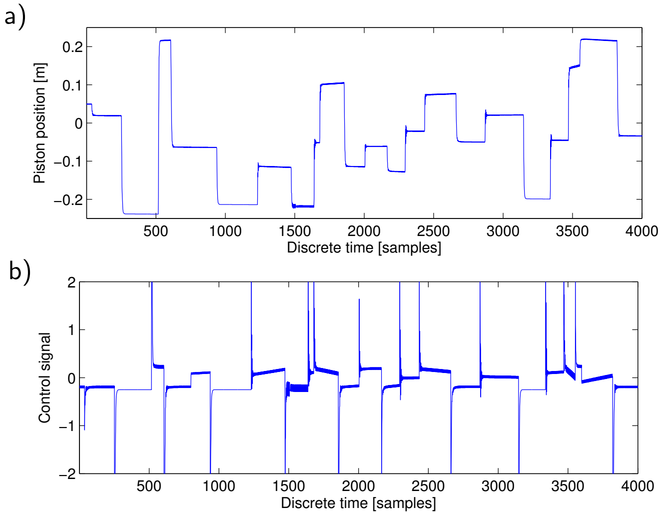


Figure: Exemplary interval of training data: the output signal (a), the control (b)

Neural controller

- neural controller design – to mimic the behaviour of the fundamental controller
- input signal – the tracking error $e(k)$,
- desired output – the control signal $u(k)$
- training set $\{e(k), u(k)\}_{k=1}^{N}$ – recorded during the evaluation of the closed-loop control with the feedback controller
- structure of the neural controller – $v_c = 5$, $n_a = 2$, $n_b = 2$, $\sigma_h = \tanh$, σ_o was linear one
- training carried out off-line for 100 steps
- neural controller will not change the stability properties; according to Theorem 1

$$\|\boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{W}_2^{x\mathsf{T}}\| = 0.7965 < 1$$

ILC synthesis

- reference trajectory – ramp signals with different slopes
- at each trail the neural controller is retrained according to presented algorithm
  - learning rate: $\eta = 0.1$
  - decay parameter: $\lambda = 0.00005$
  - small number of training epochs: 5
  - when the value of the error at the current trial is larger that the norm derived at the previous one the controller weights are restored from the previous trial
  - at each learning epoch the stability is checked, if violated weights are restored
- each trial was examined using the criterion in the form of the tracking error vector norm
- in the case of the PI control $\|e(k)\| = 0.5057$

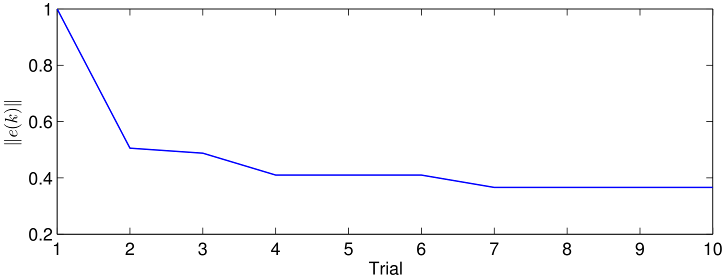


Figure: Norm of the tracking error over 8 trials.

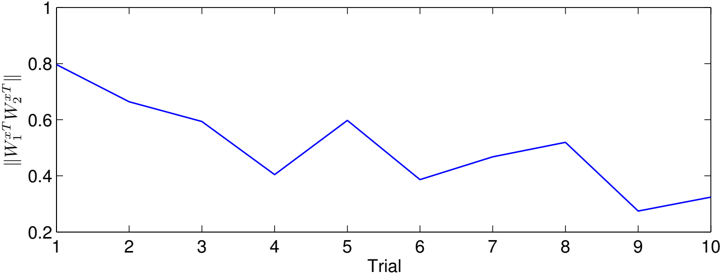


Figure: Stability: values of the criterion (16) along trials.

Concluding remarks

- A new approach for ILC synthesis based on neural networks was proposed
- The proposed control scheme may lead to significant improvement of control system performance.
- Advantages of the proposed approach are the great flexibility of neural controller in adaptation to plant nonlinearities and simplicity of the ultimate training algorithm
- The solution was tested on the pneumatic servomechanism using different working conditions of the plant with promising results
- There is still a room for refinements:
  - improving the performance of neural controller
  - developing robust neural netwotrk based ILC

Proof.

Let $\boldsymbol{V}(\boldsymbol{z}) = \|\boldsymbol{z}\|$ be a Lyapunov function for the system (15). This function is positive definite with the minimum at $\boldsymbol{z}(k) = 0$. The difference along the trajectory of the system is given as follows:

$$\Delta \boldsymbol{V}(\boldsymbol{z}(k)) = \|\boldsymbol{z}(k+1)\| - \|\boldsymbol{z}(k)\| = \|\boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{W}_2^{x\mathsf{T}} \bar{\sigma}(\boldsymbol{z}(k))\| - \|\boldsymbol{z}(k)\|. \qquad (17)$$

The activation function σ_h is a short map with the Lipschitz constant $L = 1$. Then $\bar{\sigma}$ is also a short map, with the property $\|\bar{\sigma}(\boldsymbol{z}(k))\| \leq \|\boldsymbol{z}(k)\|$, and (17) can be expressed in the form

$$\Delta \boldsymbol{V}(\boldsymbol{z}(k)) \leq \|\boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{W}_2^{x\mathsf{T}} \boldsymbol{z}(k)\| - \|\boldsymbol{z}(k)\| \leq (\|\boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{W}_2^{x\mathsf{T}}\| - 1)\|\boldsymbol{z}(k)\|. \qquad (18)$$

From (18) one can see that if

$$\|\boldsymbol{W}_1^{x\mathsf{T}} \boldsymbol{W}_2^{x\mathsf{T}}\| < 1, \qquad (19)$$

then $\Delta \boldsymbol{V}(\boldsymbol{z}(k))$ is negative definite and the system (15) is globally asymptotically stable, which completes the proof.

$\square$