Neural-network-based high-order iterative learning control

Krzysztof Patan, Maciej Patan

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

Introduction

- High-order iterative learning control (ILC) – to use data of more than one trial to improve reference tracking
- High-order ILC – able to achieve better convergence than the first-order ILC
- A simple linear combination of previous control signals may not provide new significant information
- High-order ILC – required for monotonic convergence
- Problems in question: what about nonlinear systems and nonlinear compilation of previous control information
- Neural networks – useful to design nonlinear ILC to control of nonlinear plants
- Using data of more previous trials – more stable training of neural network controller

Introduction

Objectives of the paper

1. to develop a novel nonlinear high-order ILC scheme using neural network based controller
2. to provide convergence analysis of the proposed control scheme and discuss how convergence conditions can be incorporated into the controller training

General idea of neural-network-based ILC

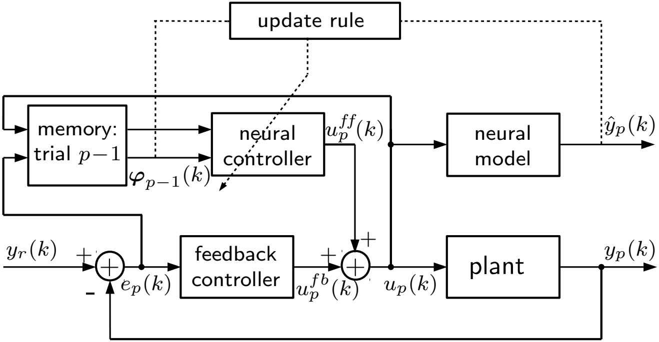
- Adaptation of so-called first-order ILC scheme (*current iteration ILC*)
  - use of existing feedback controller for stabilization,
  - adding supporting feedforward neural controller for tracking improvement,

$$u_p(k) = u_p^{fb}(k) + u_p^{ff}(k)$$

where p – trial number
k – discrete-time index
$u_p^{fb}(k)$ – feedback control
$u_p^{ff}(k)$ – ILC update

- Objective of neural controller – significant improvement of tracking for reference trajectory $y_r(k)$

Structure of the neural based ILC scheme



- Model of the process required for synthesis of the ILC controller – reasonable application of data-driven neural modelling
- Efficient scheme in case of control of nonlinear processes supporting existing feedback controller

System description

Consider a class of discrete-time nonlinear systems

$$\begin{aligned} \boldsymbol{x}_p(k+1) &= g(\boldsymbol{x}_p(k), u_p(k)), \;\; k = 0, \ldots, N-1, \\ y_p(k) &= \boldsymbol{C}\boldsymbol{x}_p(k) \end{aligned}$$

where $p \geq 0$ – a trial number
N – a trial length
$\boldsymbol{x}_p(k)$, $u_p(k)$, $y_p(k)$ – system state, input and response
g – some nonlinear function

Assumptions

Assumption A1. Let $y_r(k)$ be a reference trajectory defined over a discrete time k, which is assumed to be realizable, that is there exists a unique $u_r(k)$ and an initial state $\boldsymbol{x}_r(0)$, i.e.

$$\begin{aligned} \boldsymbol{x}_r(k+1) &= g(\boldsymbol{x}_r(k), u_r(k)) \\ y_r(k) &= \boldsymbol{C}\boldsymbol{x}_r(k) \end{aligned}$$

Assumption A2. The identical initial condition holds for all trials, i.e.

$$\forall p \ \ \boldsymbol{x}_p(0) = \boldsymbol{x}_r(0)$$

Assumption A3. The nonlinear function g satisfies the global Lipschitz condition

$$\|g(\boldsymbol{x}_1, u_1) - g(\boldsymbol{x}_2, u_2)\| \leq L\left(\|\boldsymbol{x_1} - \boldsymbol{x}_2\| + |u_1 - u_2|\right)$$

where $L > 0$ stands for the Lipschitz constant.

Neural controller

- The key idea – use the neural network to realize the function $u_p^{ff}(k)$
- Let consider the controller in the form:

$$u_p^{ff}(k) = f(\boldsymbol{\varphi}_{p-1}(k))$$

where f is a nonlinear function

- $\boldsymbol{\varphi}_{p-1}(k)$ – regression vector, e.g.
  - $\boldsymbol{\varphi}_{p-1}(k) = [u_{p-1}(k), e_{p-1}(k)]^\mathsf{T}$ – P-type controller
  - $\boldsymbol{\varphi}_{p-1}(k) = [u_{p-1}(k), e_{p-1}(k+1)]^\mathsf{T}$ – D-type controller
- high-order P-type ILC controller

$$\boldsymbol{\varphi}_{p-1}(k) = [u_{p-1}(k), ..., u_{p-M}(k), e_{p-1}(k), ..., e_{p-M}(k)]^\mathsf{T}$$

where M – order of the learning controller

Structure of the neural network controller

- Neural network with one hidden layer

$$u_p^{ff}(k) = f(\boldsymbol{\varphi}_{p-1}(k)) = \boldsymbol{W}_{2,p}\sigma(\boldsymbol{W}_{1,p}\boldsymbol{\varphi}_{p-1}(k) + \boldsymbol{b}_{1,p}) + \boldsymbol{b}_{2,p}$$

where $\boldsymbol{W}_{1,p}$, $\boldsymbol{W}_{2,p}$ – weight matrices
$\boldsymbol{b}_{1,p}$, $\boldsymbol{b}_{2,p}$ – bias vectors
σ – hidden neurons activation function

- Network parameters are updated after each process trial
- Training process

$$\boldsymbol{\theta}_p^{\star} = \arg\min\left[\frac{1}{2}\sum_{k=0}^{N-1}(y_r(k) - y_p(k;\boldsymbol{\theta}_p))^2 + \frac{1}{2}\mu_p\sum_{i=1}^{P}\theta_{p,i}^2\right]$$

where $\boldsymbol{\theta}_p$ is the vector of controller parameters

Convergence analysis

- Controlled system

$$\begin{aligned} \boldsymbol{x}_p(k+1) &= g(\boldsymbol{x}_p(k), u_p(k)), \;\; k = 0, \ldots, N-1, \\ y_p(k) &= \boldsymbol{C}\boldsymbol{x}_p(k) \end{aligned} \tag{1}$$

- Neural controller

$$u_p^{ff}(k) = f(\boldsymbol{\varphi}_{p-1}(k)) = \boldsymbol{W}_{2,p}\sigma(\boldsymbol{W}_{1,p}\boldsymbol{\varphi}_{p-1}(k) + \boldsymbol{b}_{1,p}) + \boldsymbol{b}_{2,p}, \tag{2}$$

with

$$\boldsymbol{\varphi}_{p-1}(k) = [u_{p-1}(k), ..., u_{p-M}(k), e_{p-1}(k), ..., e_{p-M}(k)]^{\mathsf{T}} \tag{3}$$

- Define controller sensitivities (with respect to input and error, respectively)

$$f_{i+1}^u(k) = \frac{\partial f(\boldsymbol{\varphi}_p(k))}{\partial u_{p-i}(k)}, \quad f_{i+1}^e(k) = \frac{\partial f(\boldsymbol{\varphi}_p(k))}{\partial e_{p-i}(k)}$$

Main result

Lemma 1

Let us suppose a real positive sequence $\{a_p\}_1^\infty$ *satisfying*

$$a_p \leq \beta_1 a_{p-1} + \beta_2 a_{p-2} + \ldots, \beta_N a_{p-M} + \epsilon,$$

where $\beta_i \geq 0$, $\epsilon \geq 0$ *and*

$$\beta = \sum_{i=1}^{M} \beta_i < 1.$$

Then, the following inequality holds

$$\lim_{p \to \infty} a_p \leq \frac{\epsilon}{1-\beta}.$$

Theorem 1

Let consider the second order ILC law (2)-(3) $(M = 2)$ *applied to the nonlinear system* (1) *satisfying Assumptions A1-A3. If*

$$\beta_1 + \beta_2 < 1 \tag{4}$$

is satisfied then the convergence of the control law is guaranteed, i.e.

$$\forall k \quad \lim_{p \to \infty} u_p(k) = u_r(k). \tag{5}$$

where $\beta_i = \gamma_{i1} + \gamma_{i2} S_\alpha$

$\gamma_{i1} = \sup_k \| f_i^u(k) \|$

$\gamma_{i2} = \sup_k \| f_i^e(k) \boldsymbol{C} \|$

$S_\alpha = \frac{1-\alpha^{-(\lambda-1)N}}{1-\alpha^{-(\lambda-1)}} - 1$

α – *the Lipschitz constant of the system*

Sketch of proof

- the proof can be obtained as an extension of the approach presented in: K. Patan: Robust and fault-tolerant control. Neural-network-based solutions, Springer, 2019
- proof is based on deriving uniform convergence property

$$\lim_{p\to\infty} u_p(k) = u_r(k),$$

through analysis of the induced norm imposed on the control law

$$\|z(k)\|_\lambda = \sup_{k\in[0,N-1]} \alpha^{-\lambda k}\|z(k)\|$$

- to deal with a nonlinear representation, the learning controller is expanded into Taylor series
- recursive nature of the state-space representation is also used

Corollary 1

Let consider M order ILC law (2)-(3) *applied to the nonlinear system* (1) *satisfying Assumptions A1-A3. If*

$$\sum_{i=1}^{M} \beta_i < 1 \tag{6}$$

is satisfied then the convergence of the control law is guaranteed, i.e.

$$\forall k \quad \lim_{p \to \infty} u_p(k) = u_r(k). \tag{7}$$

Sketch of proof

- the proof can be obtained as an extension of Theorem 1 and using Lemma 1
- expanding the proof of Theorem 1 to M-order controller

$$\|\Delta u_{p+1}(k)\|_\lambda \leq \sum_{i=1}^{M} (\gamma_{i1} + \gamma_{i2} \cdot S_\alpha) \|\Delta u_{p+i-1}(k)\|_\lambda,$$

where $\gamma_{i1} = \sup_k \|f_i^u(k)\|$, $\gamma_{i2} = \sup_k \|f_i^e(k)\boldsymbol{C}\|$.

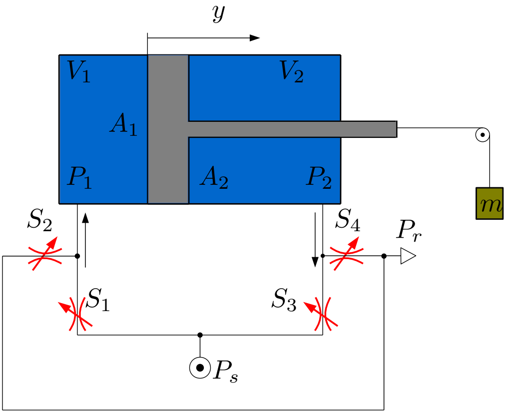
- let define

$$\beta_i = \gamma_{i1} + \gamma_{i2} \cdot S_\alpha,$$

- based on Lemma 1 we obtain

$$\sum_{i=1}^{M} \beta_i < 1$$

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 $u < 0$

Synthesis of ILC neural controller

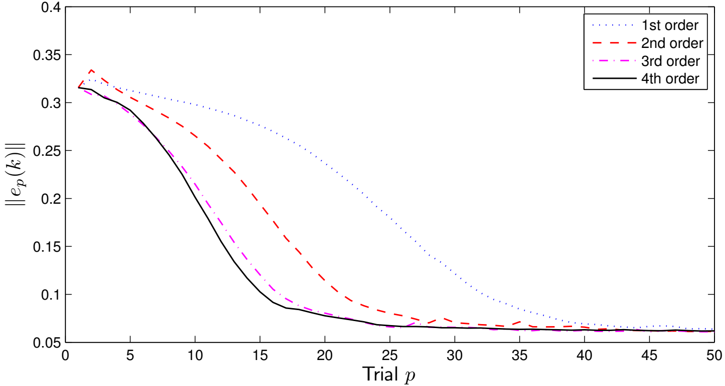
- investigated controllers: $M = 1, \ldots, 4$
- structure of the neural controller: the umber of hidden neurons $v = 1, \ldots, 100$, the activation function $\sigma_h \equiv \tanh$
- controller parameters – randomly initiated
- performance index:

$$\|e_p(k)\| = \sqrt{\sum_{j=1}^{N} |e_p(j)|^2},$$

where $e_p(k) = y_r(k) - y_p(k)$

Experiment 1

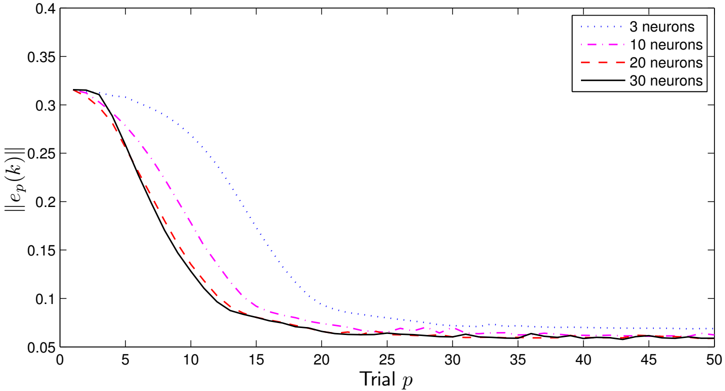
Comparison of convergence rates for $v = 20$



Performance index	ILC order			
	1	2	3	4
$\|e_{50}(k)\|$	0.0702	0.0673	0.0627	0.0603

Experiment 2

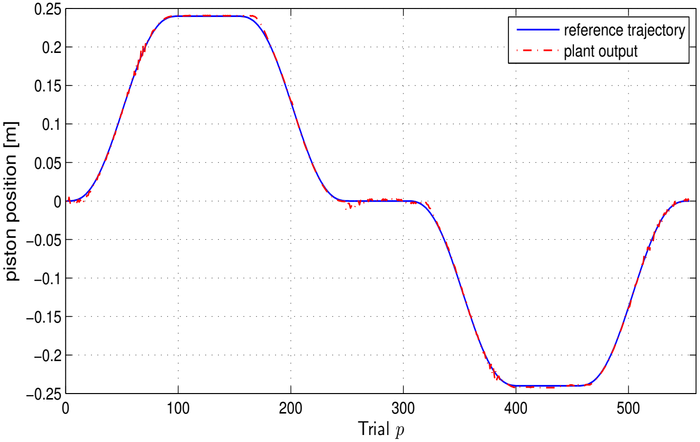
Convergence rate vs. the number of hidden units



Performance index	Neuron number 7	10	12	15	20	30
$\|e_{50}(k)\|$	0.0648	0.0634	0.061	0.06094	0.0603	0.0581

Experiment 3

Reference trajectory tracking



- a plant output was disturbed by a white noise of the magnitude equal to 2% of the maximum output value of the plant
- quality of the fourth-order neural controller: $J_n = \|e_{50}(k)\| = 0.0603$.
- for linear fourth-order controller:

$$u_{p+1}(k) = u_p(k) + \sum_{j=0}^{3} q_i e_{p-i}(k).$$

$J_l = \|e_{50}(k)\| = 0.0659$

- relative error represented as

$$\delta = \frac{J_l - J_n}{J_n} \cdot 100\% = 9.3\% \qquad (8)$$

Concluding remarks

- A novel approach for synthesis of nonlinear high-order ILC based on neural networks was proposed
- The proposed control scheme may lead to significant improvement of the convergence rate
- Advantages of the proposed approach:
  1. flexibility of neural controller in adaptation to plant nonlinearities
  2. versality in terms of developing different ILC schemes, e.g. D-type ILC
- There is still open problems:
  - automatic selection of ILC order as a trade-off between the controller complexity and control performance
  - developing more robust optimization procedures for neural network training