2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Design and convergence of iterative learning control based on neural networks

Krzysztof Patan and Maciej Patan[1]

Abstract— **The purpose of this study is to develop an effective approach to control design for repetitive nonlinear processes based on the iterative learning control technique. The idea adopted here to circumvent the inherent uncertainty of the controlled system modelling is to enhance the iterative learning scheme with neural networks used for controller design and error estimation. In result, we obtain the iterative control update rule, which can be calculated based on efficient data-driven technique for training neural network. Finally, the proposed approach is illustrated on the application example to nonlinear pneumatic servomechanism.**

I. Introduction

Increasing complexity of modern control systems together with restrictive requirements in the domain of nonlinear processes forced increased attention of researchers on using more sophisticated methods of control. Classical approaches to design the control systems dedicated for linear systems are insufficient to achieve the required quality of control or are not straightforward to apply in case of nonlinear systems. Therefore, engineering research focus on developing more accurate control techniques which are able to adapt to changing environment or/and uncertainties.

From among new modern control methods, the Iterative Learning Control (ILC) technique which emerged in the context of repetitive processes and quickly attracted a large interest in the community because its great flexibility to improve the control quality based on the measurement data [2]. The main objective is to replicate some operations in consecutive trials, as to obtain the best tracking quality for some arbitrarily given reference trajectory. However, the conventional industrial controllers usually replicate tracking errors at consecutive process trials without ability to improve tracking error in each trial (eg. we observe the same overshoot and oscillations). Since the repeatable regime of operations is very frequently encountered not only in industrial production, but also in various engineering and scientific research areas, the ILC proved its great potential in tangible engineering applications, starting from robotics and visual servoing [18], [21], [12], [16], [15], computer numerical control machine tools [8] through injection-molding machines [6], [5], rapid thermal processing [20], combustion processes [9] not to mention on health care systems [3], [4].

Although ILC is a efficient control technique with numerous engineering applications, it is still not able to compensate random disturbances. This leads to difficulties in learning performance and this may lead to slow convergence of the tracking error. In addition to this, the great majority of existing ILC schemes still requires a mathematical model of a plant (or its nonparametric equivalent) in order to properly design the control signal. The problem becomes even more difficult when the model cannot be easily inverted leading to highly non-trivial control design. In this context, the Artificial Neural Networks (ANNs) have been intensively studied during the last two decades and proved their usefulness in control theory, especially in area of system modelling and plant control [13]. Neural networks stand for an appealing and flexible alternative to the classical methods, because they can deal with nonlinear problems [7], [10]. They are also play an indispensable role when a mathematical model of a plant is difficult or even impossible to derive. Another attractive property is the self-learning ability. A neural network can extract the system features from historical training data using a learning algorithm, requiring little prior knowledge about the process. These properties make the ANNs especially suited to incorporate in the general ILC scheme. Therefore, the main idea here is to adopt ANNs as to accurately identify the mathematical model of a plant based on the available measurement data and properly train the neural controller.

The main objective of this work is twofold. Firstly, to develop a special ILC scheme with feedforward neural controller to compensate the nonlinear system dynamics. In particular, we substantially extend the approach developed in [15] providing additional neural network for estimation of tracking error gradient required to derive the iterative control update. This give a great flexibility to adapt the control in the case of non-linear systems. Another important contribution of this work is to provide the detailed convergence analysis together and discuss their relation to the concept of training stability for the neural controller. The resulting sufficient conditions for convergence of developed ILC scheme are derived and constructively incorporated into training process of neural controller. Thus, both the control quality and convergence rate can be improved.

II. System modelling

A. System description

Let consider a class of scalar nonlinear discrete-time systems described by

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

This work was supported by National Science Centre in Poland under the grant 2014/15/B/ST7/03208.

[1] K. Patan and M. Patan are with the Institute of Control and Computation Engineering, University of Zielona Góra, ul. Szafrana 2, Zielona Góra, Poland {k.patan, m.patan}@issi.uz.zgora.pl

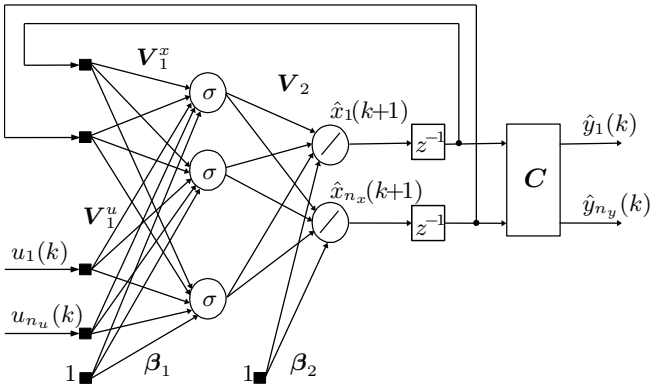

Fig. 1. Structure of the state-space neural network

where the integer $p \geq 0$ stands for the trial (cycle number), N is a length of a trial, $\boldsymbol{x}_p(k)$, $u_p(k)$ and $y_p(k)$ are state, input and output of a system along the p-th trial and g is a nonlinear function. Let make the following 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 $\boldsymbol{x}_d(0)$, i.e.

$$\begin{aligned} \boldsymbol{x}_r(k+1) &= g(\boldsymbol{x}_r(k), u_r(k)) \\ y_r(k) &= \boldsymbol{C}\boldsymbol{x}_r(k) \end{aligned}. \tag{2}$$

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

$$\forall p \; \boldsymbol{x}_p(0) = \boldsymbol{x}_r(0). \tag{3}$$

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

$$\|g(\boldsymbol{x}_1, u_1) - g(\boldsymbol{x}_2, u_2)\| \leq L\left(\|\boldsymbol{x_1} - \boldsymbol{x}_2\| + |u_1 - u_2|\right), \tag{4}$$

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

B. Model of the system

Let consider a state space neural network model of the nonlinear system (1):

$$\begin{aligned} \hat{\boldsymbol{x}}_p(k+1) &= \hat{g}(\hat{\boldsymbol{x}}_p(k), u_p(k)) \\ \hat{y}_p(k) &= \boldsymbol{C}\hat{\boldsymbol{x}}_p(k) \end{aligned}, \tag{5}$$

where $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_x}$ is the model state, $u \in \mathbb{R}^1$ and $\hat{y} \in \mathbb{R}^1$ are the input and output, respectively, the nonlinear function $\hat{g}(\cdot,\cdot)$ has the form:

$$\hat{g}(\hat{\boldsymbol{x}}_p(k), u_p(k)) = \boldsymbol{V}_2\sigma(\boldsymbol{V}_1^x\hat{\boldsymbol{x}}_p(k) + \boldsymbol{V}_1^u u_p(k) + \boldsymbol{\beta}_1) + \boldsymbol{\beta}_2), \tag{6}$$

where $\boldsymbol{V}_1^u \in \mathbb{R}^{v_m \times 1}$, $\boldsymbol{V}_1^x \in \mathbb{R}^{v_m \times n_x}$ and $\boldsymbol{V}_2 \in \mathbb{R}^{n_x \times v_m}$ are input-to-hidden, state-to-hidden and hidden-to-output layers weight matrices, respectively, $\boldsymbol{\beta}_1 \in \mathbb{R}^{v_m}$ and $\boldsymbol{\beta}_2 \in \mathbb{R}^{n_x}$ are bias vectors, $\sigma : \mathbb{R}^{v_m} \to \mathbb{R}^{v_m}$ is the activation function of the hidden layer and v_m stands for the number of hidden neurons. Finally $\boldsymbol{C} \in \mathbb{R}^{1 \times n_x}$ stands for the output (observation) matrix. The structure of the state-space neural network is shown in Fig. 1.

The network consists of two hidden layers of neurons. The first layer is composed of the neurons with nonlinear activation functions σ while the second layer includes linear units. The common choice of the nonlinear activation function is to use the hyperbolic tangent one $\sigma(x) = \tanh(x)$ satisfying the following conditions:

$$\begin{aligned} &\text{(i)} \quad \sigma(x) \to \pm 1 \text{ as } x \to \pm\infty, \\ &\text{(ii)} \quad \sigma(x) = 0 \text{ at a unique point } x = 0, \\ &\text{(iii)} \quad \sigma'(x) > 0 \text{ and } \sigma'(x) \to 0 \text{ as } x \to \pm\infty, \\ &\text{(iv)} \quad \sigma'(x) \text{ has a global maximum equal to } 1. \end{aligned} \tag{7}$$

The activation function $\sigma(x)$ is a short map then it is obvious that

$$|\sigma(x)| \leq |x|. \tag{8}$$

Lemma 1: The Lipschitz constant of the system represented by (6) is

$$L = \max\{\|\boldsymbol{V}_2\boldsymbol{V}_1^x\|, \|\boldsymbol{V}_2\boldsymbol{V}_1^u\|\}$$

Proof: Taking into account the form of the model (6) and the property of the activation function (8) it follows:

$$\begin{aligned} &\|\hat{g}(\boldsymbol{x}_1, u_1) - \hat{g}(\boldsymbol{x}_2, u_2)\| \\ &\quad \leq \|\boldsymbol{V}_2\sigma(\boldsymbol{V}_1^x\boldsymbol{x}_1 + \boldsymbol{V}_1^u u_1 + \boldsymbol{\beta}_1) - \boldsymbol{V}_2\sigma(\boldsymbol{V}_1^x\boldsymbol{x}_2 + \boldsymbol{V}_1^u u_2 + \boldsymbol{\beta}_1)\| \\ &\quad \leq \|\boldsymbol{V}_2(\boldsymbol{V}_1^x\boldsymbol{x}_1 + \boldsymbol{V}_1^u u_1 + \boldsymbol{\beta}_1) - \boldsymbol{V}_2(\boldsymbol{V}_1^x\boldsymbol{x}_2 + \boldsymbol{V}_1^u u_2 + \boldsymbol{\beta}_1)\| \\ &\quad \leq \|\boldsymbol{V}_2\boldsymbol{V}_1^x(\boldsymbol{x}_1 - \boldsymbol{x}_2) + \boldsymbol{V}_2\boldsymbol{V}_1^u(u_1 - u_2)\| \\ &\quad \leq \|\boldsymbol{V}_2\boldsymbol{V}_1^x\|\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| + \|\boldsymbol{V}_2\boldsymbol{V}_1^u\||u_1 - u_2| \end{aligned} \tag{9}$$

Let

$$L = \max\{\|\boldsymbol{V}_2\boldsymbol{V}_1^x\|, \|\boldsymbol{V}_2\boldsymbol{V}_1^u\|\} \tag{10}$$

then

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

■

III. Controller design

A. Neural controller

In the context of this work, a typical scheme of ILC can be applied, where data from whole previous trial of repetitive process are used to design of new control input (so-called first order ILC) [1], [19]. The ultimate goal of iterative control algorithm represented by some function f is to follow the reference trajectory to very high accuracy, i.e. to converge uniformly with tracking error $e_p(k) = y_r(k) - y_p(k)$ as close to zero as possible.

The key idea is to use the neural network to provide the realization of the function f (being implicitly an inverted model of the plant) based on the observational data. However, the training process of this neural controller requires an accurate estimation of the system response sensitivities with respect to neural network parameters. This can be achieved with the use of another neural network modelling the controlled plant. The general ILC scheme is depicted in Fig. 2. This scheme is based on the existing feedback controller and it is called current iteration ILC. The control law has the form:

$$u_p(k) = u_p^{fb}(k) + u_p^{ff}(k), \tag{11}$$

where $u_p^{fb}(k)$ is the control law provided by the feedback controller and $u_p^{ff}(k)$ is the learning component provided in

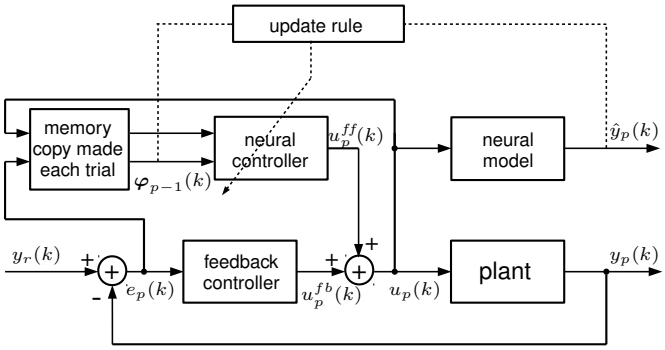

Fig. 2. General structure of iterative learning control based on neural networks.

the feedforward manner. Let consider the learning controller in the form:

$$u_p^{ff}(k) = f(\boldsymbol{\varphi}_{p-1}(k)), \tag{12}$$

where f is a nonlinear function, $\boldsymbol{\varphi}_{p-1}(k)$ is a regression vector from the previous trial. The form of the regression vector determines the controller structure:

- for P-type controller:

$$\boldsymbol{\varphi}_{p-1}(k) = [u_{p-1}(k), e_{p-1}(k)]^{\mathsf{T}}, \tag{13}$$

- for D-type controller:

$$\boldsymbol{\varphi}_{p-1}(k) = [u_{p-1}(k), e_{p-1}(k+1)]^{\mathsf{T}}. \tag{14}$$

The purpose of the controller synthesis is to find the nonlinear function f in such a way as to achieve possibly minimal tracking error $e_p(k)$ during the p-th trial. Taking into account the fact that the mathematical model of a nonlinear plant can be hard or even impossible to derive the ILC controller synthesis will rely on the suitably designed neural model of a plant. Then the proposed ILC scheme uses two neural networks: the first one plays the role of a model of the plant while the second is the learning controller. Let consider a neural network controller with one hidden layer:

$$u_p^{ff}(k) = \boldsymbol{W}_{2,p}\sigma(\boldsymbol{W}_{1,p}\boldsymbol{\varphi}_{p-1}(k) + \boldsymbol{b}_{1,p}) + \boldsymbol{b}_{2,p}, \tag{15}$$

where $\boldsymbol{W}_{1,p} \in \mathbb{R}^{v_c \times 2}$ and $\boldsymbol{W}_{2,p} \in \mathbb{R}^{1 \times v_c}$ are weight matrices of hidden and output layers at p-th trial, respectively, $\boldsymbol{b}_{1,p} \in \mathbb{R}^{v_c}$ and $\boldsymbol{b}_{2,p} \in \mathbb{R}^1$ are bias vectors of hidden and output units, respectively, $\sigma : \mathbb{R}^{v_c} \to \mathbb{R}^{v_c}$ is the activation function of the hidden layer and v_c stands for the number of hidden neurons. The neural network (15) can be viewed as the general representation of the ILC controller either P-type or D-type depending on the selection of the regression vector $\boldsymbol{\varphi}_{p-1}(k)$ recorded during the previous trial.

B. Update rule

The idea is to update neural network controller parameters after each trial according to the rule

$$\boldsymbol{\theta}_p = \boldsymbol{\theta}_{p-1} + \Delta\boldsymbol{\theta}_p, \tag{16}$$

where $\boldsymbol{\theta}_p$ stands for the generalized network parameter vector during p-th trial. The vector $\boldsymbol{\theta}_p \in \mathbb{R}^M$ represents all unknown network parameters, i.e. $\boldsymbol{\theta}_p = [\boldsymbol{W}_{1,p}, \boldsymbol{W}_{2,p}, \boldsymbol{b}_{1,p}, \boldsymbol{b}_{2,p}]$, and $\Delta\boldsymbol{\theta}_p$ is a correction term. The objective of the training is to determine the network parameters in such a way as to minimize a cost function

$$\boldsymbol{\theta}_p^{\star} = \arg\min J_p \tag{17}$$

where $\boldsymbol{\theta}_p^{\star} = [\boldsymbol{W}_{1,p}^{\star}, \boldsymbol{W}_{2,p}^{\star}, \boldsymbol{b}_{1,p}^{\star}, \boldsymbol{b}_{2,p}^{\star}]$ is the optimal controller parameter vector at p-th trial. Let define the following cost function

$$J_p = \frac{1}{2}\sum_{k=1}^{N}(y_r(k) - y_p(k))^2 + \frac{1}{2}\lambda\sum_{i=1}^{M}\theta_{p,i}^2, \tag{18}$$

where M is the number of the neural network controller parameters, λ is a parameter governing how strongly large weights are penalized, and $\theta_{p,i}$ is the i-th component of the vector $\boldsymbol{\theta}_p$. The second term in (18) is a weight decay which is a simple way to achieve good generalization of the neural network controller. Such a technique can prevent large changes of the weight values and thus the neural controller will not forget the training patterns presented during the previous trials.

The simplest and no time consuming solution is to apply the gradient descent to minimize the cost (18) with respect to the parameter vector $\boldsymbol{\theta}_p$. The following weight update is derived:

$$\Delta\boldsymbol{\theta}_p = -\eta\frac{\partial J_p}{\partial\boldsymbol{\theta}_p}, \tag{19}$$

where η represents the learning rate. The gradient of the cost function (18) with respect to the vector $\boldsymbol{\theta}_p$ can be derived using the chain rule of differentation as follows [15]:

$$\begin{aligned}\frac{\partial J_p}{\partial\boldsymbol{\theta}_p} &= -\sum_{k=1}^{N}\left(e_p(k)\frac{\partial y_p(k)}{\partial u_p(k-1)}\frac{\partial u_p(k-1)}{\partial\boldsymbol{\theta}_p}\right) + \lambda\boldsymbol{\theta}_p \\ &= -\sum_{k=1}^{N}\left(e_p(k)\frac{\partial \hat{y}_p(k)}{\partial u_p(k-1)}\frac{\partial u_p(k-1)}{\partial\boldsymbol{\theta}_p}\right) + \lambda\boldsymbol{\theta}_p\end{aligned}. \tag{20}$$

Here we apply the equivalence rule saying that a model represents the plant sufficiently well, i.e. $y(k) \approx \hat{y}(k)$. That is the reason to include the neural model in the scheme shown in Fig. 2. The first partial derivative in (20) can be determined using (5)-(6). We have

$$\frac{\partial\hat{y}_p(k)}{\partial u_p(k-1)} = \boldsymbol{C}\boldsymbol{V}_2\left(\sigma' \circ \boldsymbol{V}_1^u\right), \tag{21}$$

where $\boldsymbol{V}_1^u$ is the weight vector associated with the input $u_p(k-1)$ and σ' stands for the derivative of the activation function and $\circ$ represents the Hadamard product (an element-wise product). The second partial derivative in (20) can be calculated using the neural controller (12). For simplicity let consider that the feedback controller gives a response independent on the learning controller parameters ($u_p(k) = u_p^{ff}(k)$). We obtain:

- for weights of the first layer $\boldsymbol{W}_{1,p}$:

$$\frac{\partial u_p(k-1)}{\partial \boldsymbol{W}_{1,p}} = (\boldsymbol{W}_{2,p} \circ \sigma')\, \boldsymbol{\varphi}_{p-1}(k-1), \tag{22}$$

- for biases of the first layer $\boldsymbol{b}_{1,p}$:

$$\frac{\partial u_p(k-1)}{\partial \boldsymbol{b}_{1,p}} = \boldsymbol{W}_{2,p} \circ \sigma', \tag{23}$$

- for weights of the second layer $\boldsymbol{W}_{2,p}$:

$$\frac{\partial u_p(k-1)}{\partial \boldsymbol{W}_{2,p}} = \sigma(\boldsymbol{W}_{1,p}\boldsymbol{\varphi}_{p-1}(k-1) + \boldsymbol{b}_{1,p}), \tag{24}$$

- for biases of the second layer $\boldsymbol{b}_{2,p}$:

$$\frac{\partial u_p(k-1)}{\partial \boldsymbol{b}_{2,p}} = \mathbf{1}, \tag{25}$$

where $\mathbf{1} \in \mathbb{R}^{v_c}$.

IV. Convergence analysis

The ILC problem can be formulated as finding a proper controller structure such that the following conditions are satisfied:

- Convergence property (P1)

$$\lim_{p\to\infty} u_p(k) = u^*(k), \tag{26}$$

- Monotonic convergence property (P2)

$$\|e_{p+1}(k)\| \leq \|e_p(k)\|. \tag{27}$$

Having tried to keep the presentation at a reasonable level of complexity, we restricted ourselves here to the case of proportional rule (so called P-type) of the iterative learning update. Nevertheless, with a little turning up, the scheme can be analogously applied to the case of derivative type (D-type) of learning update. Introducing

$$f_u(k) = \frac{\partial f(u_p(k), e_p(k))}{\partial u_p(k)}, \;\; f_e(k) = \frac{\partial f(u_p(k), e_p(k))}{\partial e_p(k)} \tag{28}$$

we are able to formulate the following result:

Theorem 1: For the nonlinear system (1), assuming that assumptions A1–A3 hold the convergence of the control law (12) with the regession vector (13) is guaranteed if

$$\gamma_1 + \gamma_2 \cdot \frac{1-\alpha^{-(\lambda-1)N}}{1-\alpha^{-(\lambda-1)}} < 1 \tag{29}$$

where where $\gamma_1 = \sup_k \|f_u(k)\|$ and $\gamma_2 = \sup_k \|f_e(k)\boldsymbol{C}\|$. The proof of this result can be obtained as an extension of the approach in [17].

Remark 1. Theorem 1 requries that the Lipschitz constant have to be greater than 1. In the view of Lemma 1 we know that L is strongly dependent on the neural model weight matrices $\boldsymbol{V}_1$, $\boldsymbol{V}_1^u$ and $\boldsymbol{V}_1^x$. Then during the model training at each training epoch it should be checked if the following condition is satisfied:

$$\max\{|\boldsymbol{V}_2\boldsymbol{V}_1^x|, |\boldsymbol{V}_2\boldsymbol{V}_1^u|\} > 1. \tag{30}$$

Algorithm 1 Lipschitz constant checking

Step 0. Set $\mu > 1$.

Step 1. Perform the weight update.

Step 2. Check the condition (30).

Step 3. If the condition (30) is not satisfied **then**

$$\boldsymbol{V}_2^{new} = \mu \boldsymbol{V}_2^{old}$$

and go to Step 1 **else** go to Step 4.

Step 4. End of training

As the matrix $\boldsymbol{V}_2$ is the common in both arguments of the maximum operator the following procedure is proposed.

Remark 2. Let analyse the last component in (29). It is assumed that $\alpha > 1$ then

$$\lim_{\lambda\to\infty} \alpha^{-(\lambda-1)} = 0, \tag{31}$$

and as N is positive similarily

$$\lim_{\lambda\to\infty} \alpha^{-N(\lambda-1)} = 0. \tag{32}$$

Finally,

$$\lim_{\lambda\to\infty} \frac{1-\alpha^{-(\lambda-1)N}}{1-\alpha^{-(\lambda-1)}} = 1, \tag{33}$$

and the condition (29) becomes

$$\gamma_1 + \gamma_2 < 1 \tag{34}$$

From the controller representation (15) and (13) partial derivatives (28) take the form:

$$f_u(k) = (\boldsymbol{W}_{2,p} \circ \sigma')\, \boldsymbol{W}_{1,p}^u, \;\; f_e(k) = (\boldsymbol{W}_{2,p} \circ \sigma')\, \boldsymbol{W}_{1,p}^e, \tag{35}$$

where $\boldsymbol{W}_{1,p}^u \in \mathbb{R}^1$ and $\boldsymbol{W}_{1,p}^e \mathbb{R}^1$ are weights associated with the input $u_p(k)$ and tracking error $e_p(k)$, respectively. According to the condition (7iv) the supremum of activation function derivative is equal to one. Moreover, from the structure of the neural network model we know that the output matrix is in the form of the unit vector ($\boldsymbol{C} = [1, 0, \ldots, 0]$). Then

$$\gamma_1 = \|\boldsymbol{W}_{2,p}\boldsymbol{W}_{1,p}^u\|, \quad \gamma_2 = \|\boldsymbol{W}_{2,p}\boldsymbol{W}_{1,p}^e\| \tag{36}$$

and the condition (34) can be rewritten as follows:

$$\|\boldsymbol{W}_{2,p}\boldsymbol{W}_{1,p}^u\| + \|\boldsymbol{W}_{2,p}\boldsymbol{W}_{1,p}^e\| < 1, \tag{37}$$

V. Illustrative example

In order to present the underlying ideas, a pneumatic servomechanism for control the position of the mass is used as an example [11], [15]. In the pneumatic servomechanism transmission of signals is carried out through the medium of compressed air. The system consists of the double acting cylinder lifting an inertial weight. The cylinder is fed from a set of four adjustable air valves. The pneumatic servomechanism is a nonlinear system widely used in the industry. The nonlinear behaviour follows from: (i) nonlinear friction forces, (ii) deadband due to stiction, (iii) dead time due to the

compressibility of air. Technical data are provided in [11], [15]. The valves are operated in such a way that for input signal $u \geqslant 0$ the valves S_1 and S_4 are open. In turn for $u < 0$ valves S_2 and S_3 are open. All valves open proportionally to their control signals. The sampling frequency is set to 10Hz.

A. Process model

The investigated system is poorly damped and includes integration action. Thus, in order to collect data for neural network training, it is necessary to use the closed-loop scheme. The simple proportional controller has been used. The reference signal should be selected in such a way as to guarantee persistent excitation of the system. The reference was selected in the form of random steps trigerred randomly with levels covering possible piston positions from the interval $(-0.245, 0.245)$. Such a prepared reference is applied in a closed loop control with the P controller $u_p^{fb}(k) = k_p e_p(k)$ with the gain $k_p = 30$. The model input is the control signal $u_p(k)$ and the modelled output is the piston position $y_p(k)$. The training set $\{u_p(k), y_p(k)\}_{k=1}^{N}$ consists of 16000 samples. The order of the model was selected after analysing the physical properties of the process. The best settings for neural network model (15) were: $n_x = 3$ and $v_m = 5$. Such settings assures acceptable modelling results. Hidden neurons consist of hyperbolic tangent activation function while the output neuron has the linear activation function. Training was carried out off-line for 100 steps with the Levenberg-Marquardt (LM) algorithm. The obtained Sum of Squared Errors is SSE=0.0438 which guarantees pretty good modelling quality.

B. ILC synthesis

In order to satisfy the convergence condition (29) the simple heuristic can be used. If the condition is not satisfied the controller weights are not updated. Moreover, some safety margin is introduced (it was set to 0.9). Then, if the condition (29) is greater than the safety level, the weights will bring back to the best ones saved during learning. The best weights constitute the lowest value of the tracing error norm during the learning.

Initially, the parameters of the ILC controller (12)-(13) are set as random values and then the neural controller is retrained after each trial according to the algorithm presented in Section III-B. In order to avoid overfitting a small number of training epochs is carried out (only 50). Through a series of experiments the well performing value of the training rate was found to be $\eta = 0.05$. To this end the weight decay was not used ($\lambda = 0$). Figure 3(a) shows how the quality of the control (marked wit the red line) over trials. At the beginning improvement is significant but later on we observe increasing value of the norm in spite of the fact that the convergence condition is satisfied (Fig. 3(b)). This is commonly observed phenomena called learning transient. In order to reduce this undesirable effect and to achieve the monotonic convergence the favourable approach is to use the low pass Q-filter which can disable learning at high frequencies. Thus, learning transients can be eliminated and

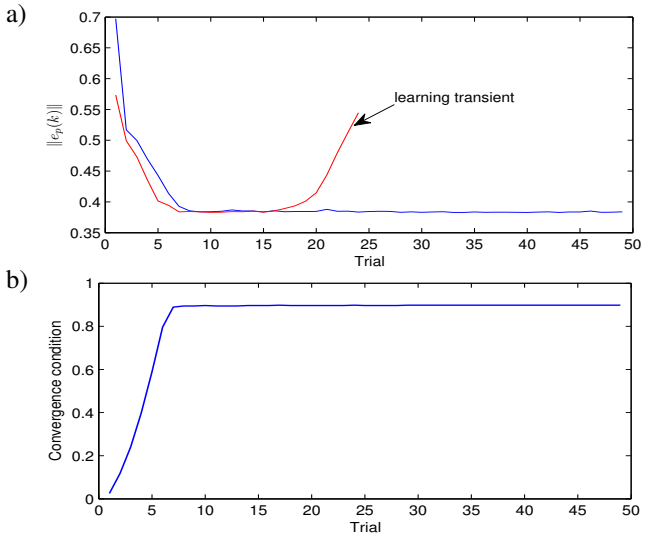

Fig. 3. Transient behaviour of ILC with (blue) and without filtering (red) (a) versus plot of convergence condition (b).

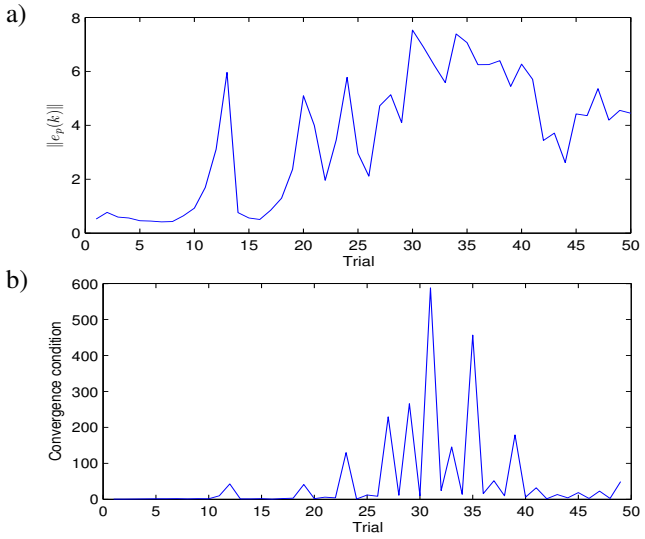

Fig. 4. Divergent ILC: performance (a), condition satisfaction (b).

simultaneously robustness of the control can be increased. The modified learning rule takes the form:

$$u_p^{ff}(k) = Q(z) f(\boldsymbol{\varphi}_{p-1}(k)), \tag{38}$$

with the Q-filter described by the transfer function: $Q(z) = 0.63/(z - 0.37)$. The cut-off frequency of the filter was set as 1.75Hz providing very good results. In Fig. 3(a) we can see the effect of introducing the control law (38) with the Q-filter (V-B) (blue line). This time the learning transients are eliminated giving nearly monotonic convergence of the ILC scheme.

The performance of the proposed ILC scheme without checking the convergence condition (29) has been also investigated. The results are shown in Fig. 4. It is evident that during the retraining of the neural controller the network weights can take large values causing that γ_1 and γ_2 represented by (36) are significantly exceeding the acceptable

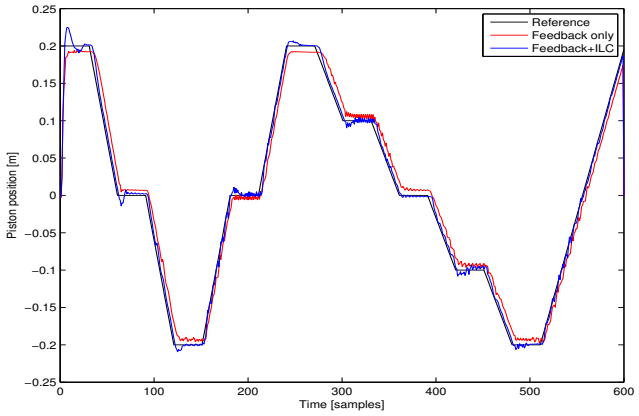

Fig. 5. Reference tracking: the reference (black), the P controller (red), ILC (blue).

level (Fig. 4(b)). Moreover, it is clearly seen that the more the convergence condition is exceeded (left hand side of (37)) the greater is the value of the tracking error norm. Even application of the Q-filter cannot prevent the divergent behaviour of the ILC scheme.

Tracking results are presented in Fig. 5. The reference trajectory is represented with the black line. Firstly, the system was investigated with the P controller only. The results are marked with the red line. In the case of the P control the tracking error norm was $\|e_p(k)\| = 0.5022$. It is clear that the P controller has problems with tracking the reference especially at the decreasing slopes. In the second experiment the classical control system with the P controller was enhanced by means of ILC according to the scheme shown in Fig. 2. The system response in this case is represented with the blue line. Contrary to the P controller the results achieved by ILC are definitely much better ($\|e_p(k)\| = 0.3876$).

VI. Concluding remarks

Clearly, the proposed approach to design of iterative learning control enhanced with neural modelling constitutes an attractive alternative to the conventional adaptive control schemes for repetitive processes. Its decided advantage is a great simplicity of the training algorithm and flexibility in applications to nonlinear systems, especially with non-invertible dynamics. We have demonstrated that the basic network structure performs very well in practice. Apart from that, as the main contribution, we have formulated the sufficient conditions for the convergence of the iterative learning scheme to the optimal solution and provided the careful analysis on how to exploit this result during the training process of a neural controller.

This work is a part of ongoing research on more general methodology for combining optimum experimental design and iterative learning control in order to increase the control quality for systems with repetitive operations. In particular, estimating the neural network response uncertainty via training data selection outlined in [14], the robust ILC scheme can be developed.

References

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic systems*, vol. 1, no. 2, pp. 123–140, 1984.

[2] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control: a learning-based method for high-performance tracking control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[3] C. T. Freeman, E. Rogers, A. Hughes, J. H. Burridge, and K. L. Meadmore, "Iterative learning control in health care: electrical stimulation and robotic-assisted upper-limb stroke rehabilitation," *Control Systems, IEEE*, vol. 32, no. 1, pp. 18–43, 2012.

[4] C. Freeman, A.-M. Hughes, J. Burridge, P. Chappell, P. Lewin, and E. Rogers, "Iterative learning control of fes applied to the upper extremity for rehabilitation," *Control Engineering Practice*, vol. 17, no. 3, pp. 368–381, 2009.

[5] F. Gao, Y. Yang, and C. Shao, "Robust iterative learning control with applications to injection molding process," *Chemical Engineering Science*, vol. 56, no. 24, pp. 7025–7034, 2001.

[6] H. Havlicsek and A. Alleyne, "Nonlinear control of an electrohydraulic injection molding machine via iterative adaptive learning," *Mechatronics, IEEE/ASME Transactions on*, vol. 4, no. 3, pp. 312–323, 1999.

[7] S. Haykin, *Neural Networks. A Comprehensive Foundation, 2nd Edition*. New Jersey: Prentice-Hall, 1999.

[8] D.-I. Kim and S. Kim, "An iterative learning control method with application for cnc machine tools," *Industry Applications, IEEE Transactions on*, vol. 32, no. 1, pp. 66–72, 1996.

[9] D. Kowalów and M. Patan, "Optimal sensor selection for model identification in iterative learning control of spatio-temporal systems," in *21st IEEE Conf. on Methods and Models in Automation and Robotics (MMAR)*, 2016, pp. 70–75.

[10] O. Nelles, *Nonlinear System Identification. From Classical Approaches to Neural Networks and Fuzzy Models*. Berlin: Springer–Verlag, 2001.

[11] M. Norgaard, O. Ravn, N. Poulsen, and L. Hansen, *Networks for Modelling and Control of Dynamic Systems*. London: Springer-Verlag, 2000.

[12] W. Paszke, E. Rogers, K. Gałkowski, and Z. Cai, "Robust finite frequency range iterative learning control design and experimental verification," *Control Engineering Practice*, vol. 21, no. 10, pp. 1310–1320, 2013.

[13] K. Patan, "Neural network based model predictive control: Fault tolerance and stability," *IEEE Transactions on System Control Technology*, vol. 23, no. 3, pp. 1147–1155, 2015.

[14] K. Patan, M. Patan, and D. Kowalów, "Optimal sensor selection for model identification in iterative learning control of spatio-temporal systems," in *55th IEEE Conference on Decision and Control (CDC)*, 2016.

[15] K. Patan, M. Patan, and D. Kowalów, "Neural networks in design of iterative learning control for nonlinear systems," in *IFAC Papers Online, 20th IFAC World Congress, Toulouse, France*, vol. 50, no. 1, July 2017, pp. 13 402–13 407.

[16] E. Rogers, K. Galkowski, and D. H. Owens, *Control systems theory and applications for linear repetitive processes*. Springer, 2007, vol. 349.

[17] D. Shen, W. Zhang, and J. Xu, "Iterative learning control for discrete nonlinear systems with randomly iteration varying lengths," *Systems & Control Letters*, vol. 96, pp. 81–87, 2016.

[18] A. Tayebi, S. Abdul, M. Zaremba, and Y. Ye, "Robust iterative learning control design: application to a robot manipulator," *Mechatronics, IEEE/ASME Transactions on*, vol. 13, no. 5, pp. 608–613, 2008.

[19] M. Uchiyama, "Formulation of high-speed motion pattern of a mechanical arm by trial," *Trans. SICE (Soc. Instrum. Contr. Eng.)*, vol. 14, no. 6, pp. 706–712, 1978.

[20] D. R. Yang, K. S. Lee, H. J. Ahn, and J. H. Lee, "Experimental application of a quadratic optimal iterative learning control method for control of wafer temperature uniformity in rapid thermal processing," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 16, no. 1, pp. 36–44, 2003.

[21] A. Yarza, V. Santibanez, and J. Moreno-Valenzuela, "An adaptive output feedback motion tracking controller for robot manipulators: uniform global asymptotic stability and experimentation," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 3, pp. 599–611, 2013.